PREMIER REFERENCE SOURCE

Pattern Recognition Technologies and Applications

Recent Advances

BRIJESH VERMA & MICHAEL BLUMENSTEIN

Pattern Recognition Technologies and Applications:
Recent Advances

Brijesh Verma
Central Queensland University, Australia

Michael Blumenstein
Griffith University, Australia

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE

Hershey · New York

Acquisitions Editor: Kristin Klinger
Development Editor: Kristin Roth
Assistant Development Editor: Jessica Thompson
Senior Managing Editor: Jennifer Neidig
Managing Editor: Jamie Snavely
Assistant Managing Editor: Carole Coulson
Copy Editor: Sue Vander Hook
Typesetter: Sean Woznicki
Cover Design: Lisa Tosheff
Printed at: Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Pattern recognition technologies and applications : recent advances / Brijesh Verma and Michael Blumenstein, editors.

p. cm.

Summary: “ This book provides cutting-edge pattern recognition techniques and applications. Written by world-renowned experts in their field, this easy to understand book is a must have for those seeking explanation in topics such as on- and offline handwriting and speech recognition, signature verification, and gender classification”--Provided by publisher.

Includes bibliographical references and index.

ISBN-13: 978-1-59904-807-9 (hardcover)

ISBN-13: 978-1-59904-809-3 (e-book)

1. Pattern perception. 2. Pattern perception--Data processing. I. Verma, Brijesh. II. Blumenstein, Michael.

Q327.P383 2008

006.4--dc22

2007037396

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Table of Contents

Detailed Table of Contents

mating the pose through a correlation model. The proposed technique is coupled with Adaptive Principal Component Analysis (APCA), which was previously shown to perform well in the presence of both lighting and expression variations. The proposed recognition techniques, although advanced, are not computationally intensive. So they are quite well suited to the embedded system environment. Indeed, the authors have implemented an early prototype of a face recognition module on a mobile camera phone so the camera could be used to identify the person holding the phone.

Chapter IX

Complex multi-object interactions result in occlusion sequences, which are a visual signature for the event. In this work, multi-object interactions are tracked using a set of qualitative occlusion primitives derived on the basis of the Persistence Hypothesis— objects continue to exist even when hidden from view. Variable length temporal sequences of occlusion primitives are shown to be well correlated with many classes of semantically significant events. In surveillance applications, determining occlusion primitives is based on foreground blob tracking and requires no prior knowledge of the domain or camera calibration. New foreground blobs are identified as putative objects, which may undergo occlusions, split into multiple objects, merge back again, and so forth. Significant activities are identified through temporal sequence mining, which bear high correlation with semantic categories (e.g., disembarking from a vehicle involves a series of splits). Thus, semantically significant event categories can be recognized without assuming camera calibration or any environment/object/action model priors.

Chapter X

Human detection is the first step for a number of applications such as smart video surveillance, driving assistance system, and intelligent digital content management. It's a challenging problem due to the variance of illumination, color, scale, pose, and so forth. This chapter reviews various aspects of human detection in static images and focuses on learning-based methods that build classifiers using training samples. There are usually three modules for these methods: feature extraction, classifier design, and merge of overlapping detections. The chapter reviews most of the existing methods for each module and analyzes their respective pros and cons. The contribution includes two aspects: first, the performance of existing feature sets on human detection are compared; second, a fast human detection system based on Histogram of Oriented Gradients features and cascaded Adaboost classifier is proposed. This chapter should be useful for both algorithm researchers and system designers in the computer vision and pattern recognition community.

With the ever-increasing utilization of imagery in scientific, industrial, civilian, and military applications, visual pattern recognition has been thriving as a research field and has become an essential enabling technology for many applications. In this chapter, we present a brain-inspired pattern recognition architecture that easily can be adapted to solve various real-world visual pattern recognition tasks. The architecture has the ability to extract visual features from images and classify them within the same network structure; in other words, it integrates the feature extraction stage with the classification stage, and both stages are optimized with respect to one another. The main processing unit for feature extraction is governed by a nonlinear biophysical mechanism known as shunting inhibition, which plays a significant role in visual information processing in the brain. Here, the proposed architecture is applied to four real-world visual pattern recognition problems; namely, handwritten digit recognition, texture segmentation, automatic face detection, and gender recognition. Experimental results demonstrate that the proposed architecture is very competitive with and sometimes outperforms existing state-of-the-art techniques for each application.

This chapter, taking FIR filters as an example, presents the discussion on efficiency of various implementation methodologies of DSP algorithms targeted at modern FPGA architectures. Nowadays, programmable technology provides the possibility to implement a digital system with the use of specialized embedded DSP blocks. In the first place, however, this technology gives the designer the possibility to increase efficiency of a designed system by exploitation of parallelisms of implemented algorithms. Moreover, it is possible to apply special techniques such as distributed arithmetic (DA). Since in this approach general-purpose multipliers are replaced by combinational LUT blocks, it is possible to construct digital filters of very high performance. Additionally, application of the functional decomposition-based method to LUT blocks optimization and mapping has been investigated. The chapter presents results of the comparison of various design approaches in these areas.

Preface

The history of automated pattern recognition can be traced back to the advent of modern computing midway through the 20th century. Since that time, the popularity and growth of the pattern recognition field has been fueled by its scientific significance and its applicability to the real world. Pattern recognition is a very challenging and multidisciplinary research area attracting researchers and practitioners from many fields, including computer science, computational intelligence, statistics, engineering, and medical sciences, to mention just a few. Pattern recognition is a process described as retrieving a pattern from a database of known patterns. It has numerous real-world applications in areas such as security, medicine, information processing, and retrieval. Some pattern recognition applications in areas such as handwriting recognition, document retrieval, speech recognition, signature verification, and face recognition are the main focus of the current research activities in the pattern recognition and computational intelligence communities around the globe. Researchers and developers are facing many challenges to applying pattern recognition techniques in many real-world applications. This book consists of 17 peer-reviewed chapters that describe theoretical and applied research work in this challenging area. The state of the art in areas such as handwriting recognition, signature verification, speech recognition, human detection, gender classification, morphological structures for image classification, logic synthesis for image and signal processing, occlusion sequence mining, probabilistic neural networks for EMG patterns, multiobjective clustering ensembles, evolutionary ensembles, support vector machines for biomedical data, and unified support vector machines is presented in various chapters of this book.

The first two chapters focus on off-line cursive handwriting recognition. In **Chapter I**, Verma and Blumenstein review existing handwriting recognition techniques and present the current state of the art in cursive handwriting recognition. Standard handwriting recognition processes are presented, and each process is described in detail. Some novel segmentation strategies and a segmentation-based approach for automated recognition of unconstrained cursive handwriting are also presented.

In **Chapter II**, Uchida investigates the theoretical and practical importance of elastic matching for handwriting recognition. He argues that the use of elastic matching techniques instead of rigid matching techniques improves the robustness of handwriting recognition systems. In addition, the optimized matching represents the deformation of handwritten characters and, thus, is useful for statistical analysis of the deformation. Elastic matching is formulated as an optimization problem of planar matching, or pixel-to-pixel correspondence, between two character images under a certain matching model such as affine and nonlinear.

The next two chapters focus on off-line signature verification. In **Chapter III**, Batista, Rivard, Sabourin, Granger, and Maupin present the current state of art in automatic signature verification. Automatic signature verification is a biometric method that can be applied in all situations where handwritten signatures are used, such as cashing a check, signing a credit card, and authenticating a document. They review existing approaches in the literature and present a survey of the most important techniques used for feature extraction and verification in this field. They also present strategies used for problems such as limited amounts of data and show important challenges and some new research directions.

In **Chapter IV**, Madasu and Lovell present an off-line signature verification and forgery detection system based on fuzzy modeling. The various handwritten signature characteristics and features are first studied and encapsulated to devise a robust verification system. The verification of genuine signatures and detection of forgeries is achieved via angle features extracted using a grid method. The derived features are fuzzified by an exponential membership function, which is modified to include two structural parameters. The structural parameters are devised to take into account the possible variations due to handwriting styles and to reflect other factors affecting the scripting of a signature. The proposed system has been tested on a large database of signatures comprising more than 1,200 signature images obtained from 40 volunteers.

Chapters V and **VI** focus on speech recognition. In **Chapter V**, Suárez-Guerra and Oropeza-Rodriguez present the state of the art in automatic speech recognition. Speech recognition is very challenging for researchers in many fields, including computer science, mathematical statistics, applied artificial intelligence, and linguistics. The unit of essential information used to characterize the speech signal in the most widely used ASR systems is the phoneme. However, several researchers recently have questioned this representation and demonstrated the limitations of the phonemes, suggesting that ASR with better performance can be developed replacing the phoneme by triphones and syllables as the unit of essential information used to characterize the speech signal. This chapter presents an overview of the most successful techniques used in ASR systems, together with some recently proposed ASR systems that intend to improve the characteristics of conventional ASR systems.

In **Chapter VI**, Leedham, Pervouchine, and Zhong investigate features of handwriting and speech and their effectiveness at determining whether the identity of a writer or speaker can be identified from handwriting or speech. For handwriting, some of the subjective and qualitative features used by document examiners are investigated in a scientific and quantitative manner based on the analysis of three characters (d, y, and f) and the grapheme th. For speech, several frequently used features are compared for their strengths and weaknesses in distinguishing speakers. The results show that some features do have good discriminative power, while others are less effective. Acceptable performance can be obtained in many situations using these features. However, the effect of handwriting forgery/disguise or conscious speech imitation/alteration on these features is not investigated. New and more powerful features are needed in the future if high accuracy person identification can be achieved in the presence of disguise or forgery.

In **Chapter VII**, Yu, Pham, and Yan present a new pattern recognition method using morphological structure. First, smooth linearization is introduced based on various chain codes. Second, morphological structural points are described in terms of smooth followed contours and linearized lines, and then the patterns of morphological structural points and their properties are given. Morphological structural points are basic tools for pattern recognition-based morphological structure. Furthermore, how the morphological structure can be used to recognize and classify images is presented. One application is document image processing and recognition, analysis, and recognition of broken handwritten digits. Another one is dynamic analysis and recognition of cell-cycle screening based on morphological structures.

In **Chapter VIII**, Shan, Bigdeli, Lovell, and Chen propose a variability compensation technique that synthesizes realistic frontal face images from nonfrontal views. It is based on modeling the face via active appearance models and estimating the pose through a correlation model. The proposed technique is coupled with adaptive principal component analysis (APCA), which was previously shown to perform well in the presence of both lighting and expression variations. The proposed recognition techniques, although advanced, are not computationally intensive. So they are quite well suited to the embedded system environment. Indeed, the authors have implemented an early prototype of a face recognition module on a mobile camera phone so the camera could be used to identify the person holding the phone.

In **Chapter IX**, Guha, Mukerjee, and Venkatesh present complex multi-object interactions resulting in occlusion sequences that are a visual signature for the event. In this chapter, multi-object interactions are tracked using a set of qualitative occlusion primitives derived on the basis of the persistence hypothesis—objects continue to exist even when hidden from view. Variable length temporal sequences of occlusion primitives are shown to be well correlated with many classes of semantically significant events. In surveillance applications, determining occlusion primitives is based on foreground blob tracking and requires no prior knowledge of the domain or camera calibration. New foreground blobs are identified as putative objects that may undergo occlusions, split into multiple objects, merged back again, and so forth. Significant activities are identified through temporal sequence mining, which bear a high correlation with semantic categories. Thus, semantically significant event categories can be recognized without assuming camera calibration or any environmental/object/action model prior.

In **Chapter X**, Jia and Zhang review human detection techniques. Human detection is the first step for a number of applications such as smart video surveillance, driving assistance systems, and intelligent digital content management. It is a challenging problem due to the variance of illumination, color, scale, pose, and so forth. This chapter reviews various aspects of human detection in static images and focuses on learning-based methods that build classifiers using training samples. There are usually three modules for these methods: feature extraction, classifier design, and merging of overlapping detections. The chapter reviews most of the existing methods for each module and analyzes their respective pros and cons. The contribution includes two aspects: first, the performance of existing feature sets on human detection are compared; second, a fast human detection system based on the histogram of oriented gradients features and a cascaded Adaboost classifier is proposed. This chapter is useful for both algorithm researchers and system designers in the computer vision and pattern recognition communities.

In **Chapter XI**, Tivive and Bouzerdoum present a brain-inspired pattern recognition architecture. With the ever-increasing utilization of imagery in scientific, industrial, civilian, and military applications, visual pattern recognition has been thriving as a research field and has become an essential enabling technology for many applications. In this chapter, a brain-inspired pattern recognition architecture that easily can be adapted to solve various real-world visual pattern recognition tasks is presented. The architecture has the ability to extract visual features from images and classify them within the same network structure; in other words, it integrates the feature extraction stage with the classification stage, and both stages are optimized with respect to one another. The main processing unit for feature extraction is governed by a nonlinear biophysical mechanism known as shunting inhibition, which plays a significant role in visual information processing in the brain. The proposed architecture is applied to four real-world visual pattern recognition problems; namely, handwritten digit recognition, texture segmentation, automatic face detection, and gender recognition. Experimental results demonstrate that the proposed architecture is very competitive with and sometimes outperforms existing state-of-the-art techniques for each application.

In **Chapter XII**, Rawski, Selvaraj, Falkowski, and Łuba present the discussion on efficiency of various implementation methodologies of DSP algorithms targeting modern FPGA architectures. Nowadays, programmable technology provides the possibility of implementing digital systems with the use of specialized embedded DSP blocks. In the first place, however, this technology gives the designer the possibility to increase the efficiency of designed systems by exploitation of parallelisms of implemented algorithms. Moreover, it is possible to apply special techniques such as distributed arithmetic (DA). Since in this approach general-purpose multipliers are replaced by combinational LUT blocks, it is possible to construct digital filters of very high performance. Additionally, application of the functional decomposition-based method to LUT block optimization and mapping has been investigated. The chapter presents results of the comparison of various design approaches in these areas.

In **Chapter XIII**, Zhou and Wang present an approach to class-dependent feature selection and a novel support vector machine (SVM). The relative background and theory are presented for describing the proposed method, and real applications of the method on several biomedical datasets are demonstrated. The authors hope that this chapter can provide readers with a different view of the feature selection method and also the classifier so as to promote more promising methods and applications.

In **Chapter XIV**, Shilton and Palaniswami present a unified introduction to support vector machine (SVM) methods for binary classification, one-class classification, and regression. The SVM method for binary classification (binary SVC) is introduced first and then extended to encompass one-class classification (clustering). Next, using the regularized risk approach as a motivation, the SVM method for regression (SVR) is described. These methods are then combined to obtain a single, unified SVM formulation that encompasses binary classification, one-class classification, and regression (as well as some extensions of these), and the dual formulation of this unified model is derived. A mechanical analogy for the binary and one-class SVCs is given to provide an intuitive explanation of the operation of these two formulations. Finally, the unified SVM is extended to implement general cost functions, and an application of SVM classifiers to the problem of spam e-mail detection is considered.

In **Chapter XV**, Faceli, Carvalho, and Souto investigate multi-objective clustering ensembles for clustering techniques. Clustering is an important tool for data exploration. Several clustering algorithms exist, and new algorithms are frequently proposed in the literature. These algorithms have been very successful in a large number of real-world problems. However, there is no clustering algorithm, optimizing only a single criterion, able to reveal all types of structures (homogeneous or heterogeneous) present in a dataset. In order to deal with this problem, several multi-objective clustering and cluster ensemble methods have been proposed in the literature, including a multi-objective clustering ensemble algorithm. In this chapter, an overview of these methods, which, to a great extent, are based on the combination of various aspects from traditional clustering algorithms, is presented.

In **Chapter XVI**, Duell and Yao present negative correlation learning in evolutionary ensembles with suitable speciation techniques. Negative correlation learning (NCL) is a technique that attempts to create an ensemble of neural networks whose outputs are accurate but negatively correlated. The motivation for such a technique can be found in the bias-variance-covariance decomposition of an ensemble of the learner's generalisation error. NCL is also increasingly used in conjunction with an evolutionary process, which gives rise to the possibility of adapting the structures of the networks at the same time as learning the weights. This chapter examines the motivation and characteristics of the NCL algorithm. Some recent work relating to the implementation of NCL in a single objective evolutionary framework for classification tasks is presented, and the authors examine the impact of two different speciation techniques: implicit fitness sharing and an island model population structure. The choice of such speciation techniques can have a detrimental effect on the ability of NCL to produce accurate and diverse ensembles and should, therefore, be chosen carefully. This chapter also provides an overview of other researchers' work with NCL and gives some promising future research directions.

In **Chapter XVII**, Tsuji, Bu, and Fukuda present a recurrent probabilistic neural network for EMG pattern recognition. In the field of pattern recognition, probabilistic neural networks (PNNs) have been proven as an important classifier. For pattern recognition of EMG signals, the characteristics usually used are amplitude, frequency, and space. However, significant temporal characteristics exist in the transient and nonstationary EMG signals, which cannot be considered by traditional PNNs. In this chapter, a recurrent PNN called recurrent log-linearized Gaussian mixture network (R-LLGMN) is introduced for EMG pattern recognition, with the emphasis on utilizing temporal characteristics. The structure of R-LLGMN is based on the algorithm of a hidden Markov model (HMM), which is a routinely used technique for modeling stochastic time series. Since R-LLGMN inherits advantages from both HMM and

neural computation, it is expected to have a higher representation ability and show better performance when dealing with time series such as EMG signals. Experimental results show that R-LLGMN can achieve high discriminant accuracy in EMG pattern recognition.

Brijesh Verma, Central Queensland University, Australia
Michael Blumenstein, Griffith University, Australia
Editors

Acknowledgment

The editors of this book wish to thank a number of individuals without whose assistance this project would not have been possible.

First, we would like to thank the contributors who have provided a number of excellent chapters that span some of the most exciting and state-of-the-art topics in the field of pattern recognition theory and applications; and the referees who devoted their time and expertise in ensuring that this book is of the highest academic standard through a rigorous peer-review process.

We would like to thank Central Queensland University and Griffith University and its staff in providing the editors with the opportunity and resources for making this project possible.

We would also like to thank the staff at IGI Global for their support and editorial assistance, which has facilitated the publication of this book.

Finally, we would like to thank our families for their encouragement and support during the editing of this book.

Brijesh Verma, Central Queensland University, Australia
Michael Blumenstein, Griffith University, Australia
Editors

Chapter I
Fusion of Segmentation Strategies for Off-Line Cursive Handwriting Recognition

Brijesh Verma
Central Queensland University, Australia

Michael Blumenstein
Griffith University, Australia

ABSTRACT

Cursive handwriting recognition is a challenging task for many real-world applications such as document authentication, form processing, postal address recognition, reading machines for the blind, bank check recognition, and interpretation of historical documents. Therefore, in the last few decades, researchers have put an enormous effort into developing various techniques for handwriting recognition. This chapter reviews existing handwriting recognition techniques and presents the current state of the art in cursive handwriting recognition. The chapter also presents segmentation strategies and a segmentation-based approach for automated recognition of unconstrained cursive handwriting. The chapter provides a comprehensive literature review with basic and advanced techniques and research results in handwriting recognition for graduate students as well as for advanced researchers.

INTRODUCTION

Cursive handwriting recognition systems are in enormous demand by law enforcement agencies, financial institutions, postal services, and a variety of other industries in addition to the general public nationally and globally. Currently, there are no commercial solutions available to deal with the problem of automated reading of *totally unconstrained* cursive handwriting from static surfaces (i.e., paper-based forms, envelopes, documents, checks, etc.). The domain of reading handwriting from static images is called off-line recognition, not to be confused with online approaches com-

monly associated with personal digital assistants (PDAs) and hand-held computers.

The research on cursive handwriting recognition has grown significantly in recent years. In the literature, many papers have been published with research detailing new techniques for the classification of handwritten numerals, characters, and words (Arica & Yarman-Vural, 2002; Blumenstein, Liu, & Verma, 2004; Blumenstein & Verma, 1999; Blumenstein & Verma, 2001; Blumenstein, Verma, & Basli, 2003; Britto, Sabourin, Bortolozzi, & Suen, 2004; Camastra & Vinciarelli, 2003; Casey & Lecolinet, 1996; Chevalier, Geoffrois, Preteux, & Lemaltre, 2005; Chiang, 1998; Cho, 1997; Dimauro, Impedovo, Pirlo, & Salzo, 1998; Dunn & Wang, 1992; Eastwood, Jennings, & Harvey, 1997; Elliman & Lancaster, 1990; Fan & Verma, 2002; Fujisawa, Nakano, & Kurino, 1992; Gader, Mohamed, & Chiang, 1997; Gang, Verma, & Kulkarni, 2002; Gatos, Pratikakis, Kesidis, & Perantonis, 2006; Gilloux, 1993; Günter & Bunke, 2004; Günter & Bunke, 2005; Hanmandlu, Murali, Chakraborty, Goyal, & Choudhury, 2003; Kapp, de Almendra Freitas, & Sabourin, 2007; Koerich, Britto, Oliveira, & Sabourin, 2006; Lee & Coelho, 2005; Liu & Fujisawa, 2005; Lu, 1995; Lu & Shridhar, 1996; Marinai, Gori, & Soda, 2005; Martin, Rashid, & Pittman, 1993; Plamondon & Srihari, 2000; Schambach, 2005; Singh & Amin, 1999; Srihari, 1993; Srihari, 2006; Suen, Legault, Nadal, Cheriet, & Lam, 1993; Suen & Tan, 2005; Verma, 2003; Verma, Blumenstein, & Ghosh, 2004; Verma, Blumenstein, & Kulkarni, 1998; Verma, Gader, & Chen, 2001; Viard-Gaudin, Lallican, & Knerr, 2005; Vinciarelli, Bengio, & Bunke, 2003; Wang, Ding, & Liu, 2005; Wen, Lu, & Shi, 2007; Xiao & Leedham, 2000; Xu, Lam, & Suen, 2003; Yanikoglu & Sandon, 1998). Some researchers have obtained very promising results for isolated/segmented numerals and characters using conventional and intelligent techniques. However, the results obtained for the segmentation and recognition of cursive handwritten words have not been satisfactory in comparison (Arica & Yarman-Vural, 2002; Blumenstein & Verma, 1999; Blumenstein & Verma, 2001; Blumenstein et al., 2003; Camastra & Vinciarelli, 2003; Chevalier et al., 2005; Chiang, 1998; Dimauro et al., 1998; Eastwood et al., 1997; Fan & Verma, 2002; Gader et al., 1997; Gang et al., 2002; Gatos et al., 2006; Gilloux, 1993; Günter & Bunke, 2004; Günter & Bunke, 2005; Hanmandlu et al., 2003; Kapp et al., 2007; Koerich et al., 2006; Lee & Coelho, 2005; Martin et al., 1993; Schambach, 2005; Srihari, 1993; Srihari, 2006; Verma, 2003; Verma et al., 1998; Verma et al., 2001; Verma et al., 2004; Viard-Gaudin et al., 2005; Vinciarelli et al., 2003; Xiao & Leedham, 2000; Yanikoglu & Sandon, 1998) The reason for not achieving satisfactory recognition rates is the difficult nature of cursive handwriting (cursive, touching, individual, etc.) and difficulties in the accurate segmentation and recognition of cursive and touching characters.

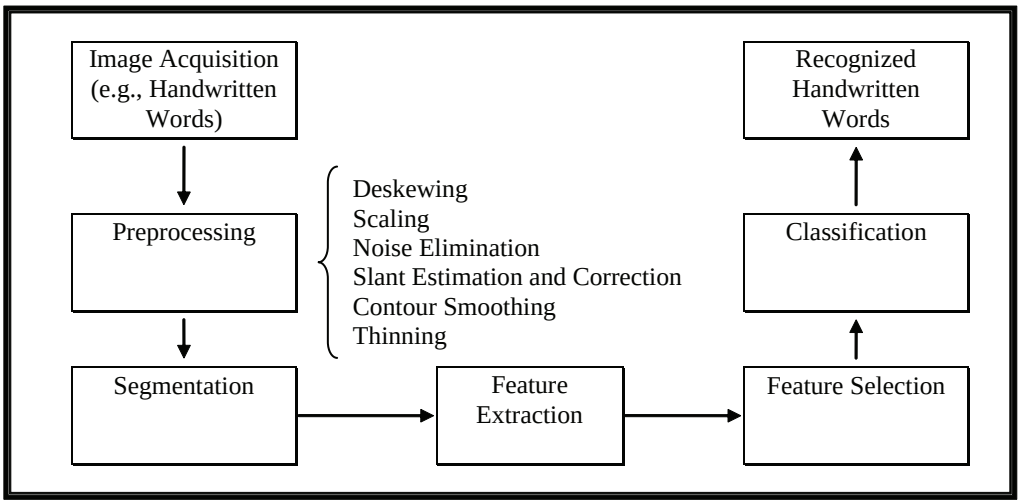
This chapter reports on the state of the art in handwriting recognition research and methods for segmentation of cursive handwriting. The remainder of this chapter is broken up into four sections. Section 2 provides an overview of handwriting recognition and methodologies used for this process. Section 3 reviews the accuracy of existing systems/techniques for handwriting recognition. Section 4 deals with fusion of segmentation strategies for cursive handwriting recognition, and Section 5 provides conclusions and future research.

TYPICAL HANDWRITING RECOGNITION SYSTEM

A typical handwriting recognition system is characterized by a number of steps, which include (1) digitization/image acquisition, (2) preprocessing, (3) segmentation (4) feature extraction, and (5) recognition/classification. Figure 1 illustrates one such system for handwritten word recognition.

The steps required for typical handwriting recognition are described next in detail.

Figure 1. Typical segmentation-based handwriting recognition system



Preprocessing

Preprocessing aims at eliminating the variability that is inherent in cursive and hand-printed words. Following is a list of preprocessing techniques that have been employed by various researchers in an attempt to increase the performance of the segmentation/recognition process:

- Deskewing
- Scaling
- Noise Elimination
- Slant Estimation and Correction
- Contour Smoothing
- Thinning

Deskewing is the process of first detecting whether the handwritten word has been written on a slope and then rotating the word if the slope's angle is too high so the baseline of the word is horizontal. Some examples of techniques for correcting slope are described in Senior (1994) and Brown and Ganapathy (1983).

Scaling sometimes may be necessary to produce words of relative size. In the case of Burges, Be, and Nohl (1992), the authors used a neural network for the segmentation stage of their system. The neural network accepted areas between the upper and lower baselines of each word as input. This area, called the core, must be of fixed height to be used in conjunction with the neural net. Therefore, it was necessary to scale the words so that all cores were of an identical height.

Noise (small dots or blobs) may be introduced easily into an image during image acquisition. Noise elimination in word images is important for further processing; therefore, these small foreground components are usually removed. Chen, Kundu, Zhou, and Srihari (1992) used morphological opening operations to remove noise in handwritten words. Kim, Govindaraju, and Srihari (1999) identified noise in a word image by comparing the sizes and shapes of connected components in an image to the average stroke width. Madhvanath, Kleinberg, and Govindaraju (1999) also analyzed the size and shape of connected components in a word image and compared them to a threshold to remove salt and pepper noise. In postal address words and other real-world applications, larger noise such as underlines is sometimes present. Therefore, some researchers have also applied some form of

underline removal to their word images (Dimauro, Impedovo, Pirlo & Salzo, 1997).

Slant estimation and correction is an integral part of any word image preprocessing. Bozinovic and Srihari (1989) employed an algorithm that estimated the slant of a word by first isolating those parts of the image that represented near vertical lines (accomplished by removing horizontal strokes through run-length analysis). Second, an average estimation of the slant given by the near-vertical lines was obtained. The word was then slant corrected by applying a transformation. In their system, the presence of a slant correction procedure was essential for segmenting their words using vertical dissection. Other estimation and correction techniques have been employed in the literature. Some have accomplished this using the chain code histogram of entire border pixels (Ding, Kimura, Miyake & Shridhar, 1999; Kimura, Shridhar & Chen, 1993), while others have estimated the slope through analysis of the slanted vertical projections at various angles (Guillevic & Suen, 1994). The process of slant correction introduces noise in the contour of the image in the form of bumps and holes. Therefore, some sort of smoothing technique is usually applied (as previously discussed for numeral recognition) to remove contour noise. As also previously described, some researchers have used the skeleton of the word image to normalize the stroke width. This operation is still a topic of debate, as there are advantages and disadvantages to using the skeleton for word recognition.

Segmentation

Segmentation of handwriting is defined as an operation that seeks to decompose a word image of a sequence of characters into subimages of individual characters. Research surveys on segmentation by Casey and Lecolinet (1996), Dunn and Wang (1992), Lu (1995), Lu and Shridhar (1996), Elliman and Lancaster (1990), Fujisawa, et al. (1992), Blumenstein and Verma (2001), Gang, et al. (2002), Verma, et al. (1998), Blumenstein, et al. (2003), Verma (2003), Blumenstein and Verma (1999), Fan and Verma (2002), and Verma, et al. (2001) confirmed that segmentation is one of the most difficult processes in cursive handwriting recognition. Some recent work by a number of researchers has demonstrated encouraging results for the segmentation of cursive handwriting. Eastwood, et al. (1997) proposed a neural-based technique for segmenting cursive script. In their research, they trained a neural network with feature vectors representing possible segmentation points as well as "negative" features that represented the absence of a segmentation point. The feature vectors were manually obtained from training and test words in the CEDAR benchmark database. The accuracy of the network on a test set of possible segmentation points was 75.9%. Yanikoglu and Sandon (1998) proposed a segmentation algorithm by evaluating a cost function to locate successive segmentation points along the baseline. They reported an accuracy of 92% for their custom database of words. Dimauro, et al. (1998) proposed an advanced technique for segmenting cursive words as part of a recognition system to read the amounts on Italian bank checks. The segmentation technique is based on a hypothesis-then-verification strategy. The authors did not report a measure of the segmentation accuracy but indicated that the new approach improved the recognition of cursive words on bank checks by 6%. Nicchiotti, Scagliola, and Rimassa (2000) presented a simple but effective segmentation algorithm. The algorithm is divided into three main steps: (1) possible segmentation points detection; (2) determining the cut direction; and (3) merging of oversegmented strokes to the main character by some heuristic rules. The authors reported results of 86.9% on a subset of words from the CEDAR database. Finally, Xiao and Leedham (2000) presented a knowledge-based technique for cursive word segmentation. They obtained segmentation results of 78.3% (correct rate) on a custom dataset collected by the authors,

and 82.9% on a subset of words from the CEDAR database.

Most work in the area of cursive handwriting recognition focuses on oversegmentation and primitive matching, which has many problems. The detailed analysis (Blumenstein & Verma, 2001; Verma et al., 2004) conducted by Blumenstein and Verma has shown that most existing segmentation algorithms have three major problems: (1) inaccurately cutting characters into parts; (2) missing many segmentation points; and (3) oversegmenting a character many times, which contributes to errors in the word recognition process. This chapter presents the solution in Section IV for the aforementioned problems.

Feature Extraction

A crucial component of the segmentation-based strategy for handwriting recognition is the development of an accurate classification system for scoring individual characters and character combinations, as identified in our preliminary work (Verma et al., 2004). The literature is replete with high accuracy recognition systems for separated handwritten numerals (Cho, 1997; Kapp et al., 2007; Plamondon & Srihari, 2000; Suen et al., 1993; Wen et al., 2007; Xu et al., 2003); however, it is clear from recent studies (Arica & Yarman-Vural, 2002; Britto et al., 2004; Camastra & Vinciarelli, 2003; Hanmandlu et al., 2003; Suen & Tan, 2005; Wang et al., 2005) that the same measure of success has not been obtained for cursive character recognition. One of the ways in which researchers have tackled the problem of cursive/segmented character recognition is through the investigation of a variety of feature extraction techniques. However, the extraction of appropriate features has proved difficult based on three factors inherent in cursive/segmented character recognition: (1) the ambiguity of characters without the context of the entire word; (2) the illegibility of certain characters due to the nature of cursive writing (e.g., ornamentation, distorted character shape, etc.) (Blumenstein et al., 2004); and (3) difficulties in character classification due to anomalies introduced during the segmentation process (i.e., dissected character components (Blumenstein & Verma, 2001).

Feature Selection

There has been a significant number of feature extraction techniques developed and employed for segmentation and overall handwriting recognition (Arica & Yarman-Vural, 2002; Blumenstein & Verma, 1999; Blumenstein & Verma, 2001; Blumenstein et al., 2003; Blumenstein et al., 2004; Britto et al., 2004; Camastra & Vinciarelli, 2003; Casey & Lecolinet, 1996; Chiang, 1998; Cho, 1997; Dimauro et al., 1998; Dunn & Wang, 1992; Eastwood et al., 1997; Elliman & Lancaster, 1990; Fan & Verma, 2002; Fujisawa et al., 1992; Gader et al., 1997; Gang et al., 2002; Gilloux, 1993; Günter & Bunke, 2004; Hanmandlu et al., 2003; Kapp et al., 2007; Lu, 1995; Lu & Shridhar, 1996; Martin et al. 1993; Plamondon & Srihari, 2000; Singh & Amin, 1999; Srihari, 1993; Suen et al., 1993; Verma, 2003; Verma et al., 1998; Verma et al., 2001; Verma et al., 2004; Vinciarelli et al., 2003; Wang et al., 2005; Wen et al., 2007; Xiao, & Leedham, 2000; Xu et al., 2003; Yanikoglu & Sandon, 1998); however, the importance of a particular feature or feature value in recognizing a character has not been fully investigated. The selection of features is very important because there might be only one or two values, which are significant to recognize a particular segmented character/primitive. The research on feature selection in other pattern recognition areas has achieved promising results. The selection can be manually determined, or a better way is to automate and optimize the process by using neural genetic algorithms. The neural genetic algorithm has great advantages over traditional techniques. Our recent research has shown that neural genetic algorithms perform better in the selection of features than traditional techniques.

Genetic algorithms are a class of search methods deeply inspired by the natural process of evolution. In each iteration of the algorithm (generation), a fixed number (population) of possible solutions (chromosomes) is generated by means of applying certain genetic operations in a stochastic process guided by a fitness measure. The most important and commonly used genetic operators are recombination, crossover, and mutation. Canonical genetic representations will be chosen for feature selection because in canonical GAs, a chromosome is represented through a binary string. If a bit is 1, it means that the corresponding feature value is selected. Otherwise, the feature value is omitted in that particular iteration. The mutation operator functions on a single string and changes a bit randomly. Crossover operates on two parent strings to produce two offspring. The fitness evaluation determines the confidence level of the optimized solution. In the feature selection process, the objective is to minimize the number of feature values. The character classification rate is used for fitness evaluation. In the selection phase, the population is initialized randomly. For each member in the population, if the bit position holds a zero value, the feature is assigned to zero, and a new dataset is created. With that dataset, the neural network is trained. So for individual members in the population, there is an individual neural network that is trained with a separate dataset. Then that trained neural network is used to calculate the fitness. The stopping condition for training the neural network is equal for all the members in the population, and it is taken as the classification error. The stopping criterion of the genetic algorithm is the number of generations.

Classification

Classification in handwriting recognition refers to one of the following processes: (1) classification of characters; (2) classification of words; and (3) classification of features. A number of classification techniques has been developed and investigated for the classification of characters, words, and features. The classification techniques have used various statistical and intelligent classifiers, including k-NN, SVMs, HMMs, and neural networks.

For the classification of numerals/characters, a profuse number of techniques have been explored in the literature. Many statistical techniques have been employed for classification, such as k-Nearest Neighbor. However, some statistical methods have been found to be impractical in real-world applications, as they require that all training samples be stored and compared for the classification process (Liu & Fujisawa, 2005). In recent times, some of the most popular, powerful, and successful methods have employed neural network classifiers (Cho, 1997; Verma et al., 2004) and HMM-based techniques (Arica & Yarman-Vural, 2002; Cai & Liu, 1999), obtaining recognition rates above 99% for off-line handwritten, isolated numerals. Recently, support vector machines have been employed for numeral/character classification, also obtaining impressive results above 99% (Liu & Fujisawa, 2005). It has also been found that the use of multistage and combined classifiers has been very successful for numeral/character classification (Camastra & Vinciarelli, 2003; Cao, Ahmadi & Shridhar, 1995).

For the word recognition problem, HMM-based techniques have been popular for holistic methods (Plamondon & Srihari, 2000). Whereas for segmentation-based word recognition, neural network classification has been commonly used in conjunction with dynamic programming (Gader et al., 1997). HMMs continue to be a popular classification method in recent times (Günter & Bunke, 2005; Schambach, 2005; Viard-Gaudin et al., 2005), as is the use of classifier combinations such as neural networks and HMMs (Koerich et al., 2006). SVMs also have been used successfully for classification of words in recent studies (Gatos et al., 2006).

REVIEW OF EXISTING HANDWRITING RECOGNITION TECHNIQUES/SYSTEMS

An enormous number of papers have been published in the handwriting recognition literature in the last few decades (Arica & Yarman-Vural, 2002; Blumenstein & Verma, 1999; Blumenstein & Verma, 2001; Blumenstein et al., 2003; Blumenstein et al., 2004; Bozinovic & Srihari, 1989; Britto et al., 2004; Brown & Ganapathy, 1983; Burges et al., 1992; Camastra & Vinciarelli, 2003; Casey & Lecolinet, 1996; Chen et al., 1992; Chevalier et al., 2005; Chiang, 1998; Cho, 1997; Davis, 2005; Dimauro et al., 1997; Dimauro et al., 1998; Ding et al., 1999; Dunn & Wang, 1992; Eastwood et al., 1997; Elliman. & Lancaster, 1990; Fan & Verma, 2002; Fujisawa et al., 1992; Gader et al., 1997; Gang et al., 2002; Gatos et al., 2006; Gilloux, 1993; Guillevic & Suen, 1994; Günter & Bunke, 2004; Günter & Bunke, 2005; Hanmandlu et al., 2003; Howe, Rath, & Manmatha, 2005; Kim et al., 1999; Kimura et al., 1993; Koerich et al., 2006; Koerich, Sabourin, & Suen, 2005; Lee & Coelho, 2005; Liu & Fujisawa, 2005; Lu, 1995; Lu & Shridhar, 1996; Madhvanath et al., 1999; Marinai et al., 2005; Martin et al., 1993; Schambach, 2005; Senior, 1994; Singh & Amin, 1999; Srihari, 1993; Srihari, 2006; Suen & Tan, 2005; Suen et al., 1993; Verma, 2003; Verma et al., 1998; Verma et al., 2001; Verma et al., 2004; Viard-Gaudin et al., 2005; Vinciarelli et al., 2003; Wang et al., 2005; Xiao, & Leedham, 2000; Yanikoglu & Sandon, 1998). A number of review papers on off-line handwriting recognition have been published (Koerich, Sabourin, & Suen, 2003; Plamondon. & Srihari, 2000; Steinherz, Rivlin, & Intrator, 1999; Verma et al., 1998; Vinciarelli, 2002). In their review, Steinherz, et al. (1999) categorize off-line handwriting recognition systems into three categories: segmentation-free methods, segmentation-based methods, and perception-oriented approaches, which the authors include as methods that perform similarly to human-reading machines using features located throughout the word. The authors did not compare the experimental results of approaches reviewed, as it was felt that the field was not sufficiently mature for this. However, the authors commented that one of the most integral components of a handwriting recognition system related to the features used.

The review of Vinciarelli (2002) focused on a general discussion of off-line cursive word recognition and subsequently the pertinent applications relating to cursive word recognition (i.e., bank check recognition) (highest recognition rate reported: 89.2%), postal applications (highest recognition rate reported: 96.3%), and finally, generic recognition (highest recognition rate reported: 99.3%). The main approaches that Vinciarelli identified in his review are explicit segmentation-based approaches, implicit segmentation-based approaches, and human-reading-inspired approaches. The latter is similar to Steinherz's perception-oriented approaches. Vinciarelli points out that these approaches are limited to the application of bank check recognition, as they can only cope well with a small lexical. Although some high recognition rates were detailed in the review, most approaches dealt with were used on small vocabularies (lexical) for experimentation. The new frontier has been the exploration of large vocabulary off-line handwriting recognition.

The final review to be described was presented by Koerich, et al. (2003), which concentrated on the discussion of large vocabulary-based handwriting recognition systems. The authors stressed that in large vocabulary applications, segmentation-based approaches are recommended due to the large amount of training data required for use with holistic approaches. The review discussed methods for handling large vocabulary recognition such as lexicon reduction. The research of some authors was compared in this area. A case study was also included in the review featuring the authors' system based on HMMs. For the largest lexicon (30,000 words), a top recognition accuracy of 73.3% was achieved. The authors commented

on the number of applications available for large vocabulary systems such as postal applications, reading handwritten notes, information retrieval, and reading fields in handwritten forms. Overall, it was concluded that large vocabulary recognition systems were still immature, and accurate recognition (with a reasonable speed) was still an open-ended problem.

State of the Art in Cursive Word Recognition

In the current section, a number of very recent systems are presented, and some future directions are discussed in the field. Günter and Bunke's recent research (2004, 2005) has focused on the use of ensemble methods and HMMs. On a medium-sized vocabulary, their results (Table 1) achieved 70% to 75% accuracy. The HMM-based technique proposed by Schambach (2006) on a large vocabulary has shown reduced recognition accuracy at 60%. Meanwhile, Koerich, et al. (2005) and Koerich, et al. (2006) obtained results close to 78% on a relatively large vocabulary problem combining neural networks and HMMs. These results are in contrast to Viard-Gaudin, et al.'s work (2005) and that of Gatos, et al. (2006) on a smaller vocabulary problem, obtaining results, respectively, above 90% using HMMs and just below 90% with SVMs. Finally, the boosted tree approach proposed by Howe, et al. (2005) obtained results between 50% and 60%.

Based on the results presented, a significant difference may be noted between small-medium vocabulary research presented and those using large vocabularies. Many researchers have employed HMM-based approaches; however, some have presented hybrids using neural networks (segmental neural networks). In the hybrid approaches, the use of supporting classifiers and segmentation-based methods has assisted the recognition accuracy for unconstrained, large vocabulary word recognition problems. It is this fusion/combination and the potential for improving segmentation-based techniques that will continue to be promising for future work in unconstrained cursive word recognition.

Cursive word segmentation poses a number of the following problems:

- Algorithms to tackle the variety of writing styles as well as appropriate features to describe the suitable segmentation points of interest and for subsequently determining correct/incorrect segmentations are lacking.

Table 1. Accuracy for word recognition

Authors	Accuracy (%)	Technique	Database
Koerich et al. (2006)	78%	SNN & HMM combining low-level and high-level features	SRTP
Gatos et al. (2006)	87.68%	SVM	IAM
Howe et al. (2005)	51.1% - 63.5%	Boosted Trees	GW20
Günter and Bunke (2005)	75.61% - 82.28%	HMMs and Ensemble Methods	IAM
Viard-Gaudin et al. (2005)	92.4%	HMMs	IRONOFF
Koerich et al. (2005)	77.62% (large lexicon) - 99.29%	SNN and HMMs	SRTP
Schambach (2005)	60% (large lexicon)	HMMs	Siemens DB
Günter and Bunke (2004)	71.58%	HMMs and Classifier Ensembles	IAM

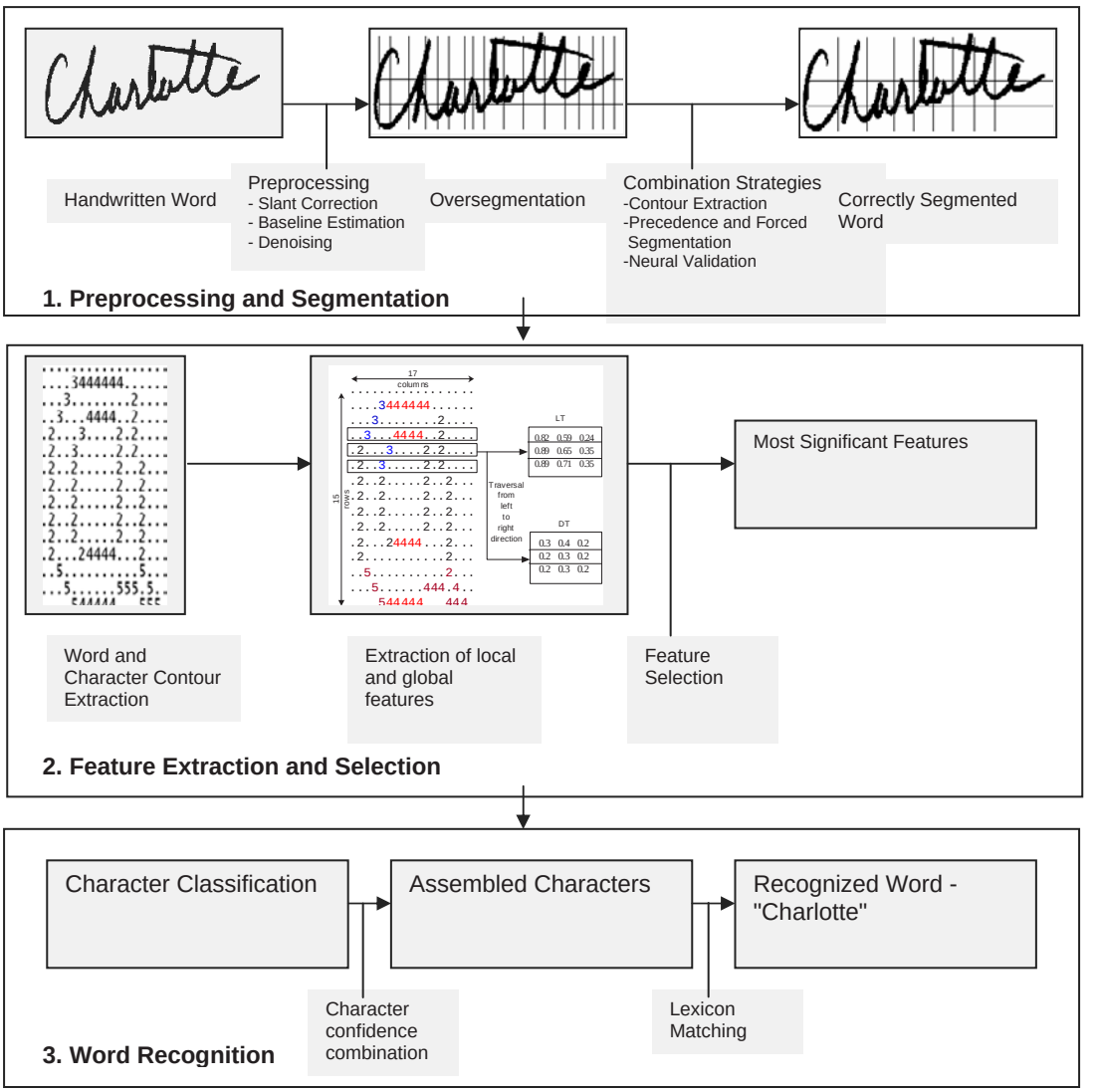
- In addition, the problem of cursive character recognition remains very much an open problem despite the success in the area of numeral recognition, as cursive characters appear ambiguous and in some cases incomplete.
- Salient features have still not been determined to adequately distinguish difficult/ambiguous, segmented/cursive characters.

In the next section, we try to tackle and solve some of the aforementioned problems by introducing combined strategies for segmentation of handwritten words.

PROPOSED STRATEGIES FOR SEGMENTATION-BASED HANDWRITING RECOGNITION

As can be seen in previous sections, the segmentation and feature extraction processes create major problems in achieving good classification

Figure 2. Proposed strategies for segmentation-based handwriting recognition



accuracy. In this section, we propose various strategies for improving the segmentation-based handwriting recognition. An overview of the proposed combination strategies for segmentation-based cursive handwriting recognition is shown in Figure 2.

Most work in the area of segmentation focuses on oversegmentation and primitive matching, which have many problems. The detailed analysis conducted by Blumenstein and Verma (2001) and Verma, et al. (2004) has shown that most existing segmentation algorithms have three major problems: (1) inaccurately cutting characters into parts; (2) missing many segmentation points; and (3) oversegmenting a character many times, which contributes to errors in the word recognition process.

First, we propose a contour-based segmentation method, which should solve the first problem. A contour extraction approach for the extraction of the character's contour between two segmentation points is very significant and useful. Contour extraction is very important because an extraction based on a vertical dissection may cut a character in half or in an inappropriate manner (missing important character components). The contour between two consecutive segmentation points is extracted using the following few steps. In the first step, disconnect the pixels near the first segmentation point, and disconnect the pixels near the second segmentation point. Find the nearest distance of the first black pixel from the first segmentation point and the baselines. Follow the contour path across that baseline having minimum distance. Find the connecting contour. Mark it as "visited" once it is located. If the contour has already been visited, then discard that and take the other part, if any.

Second, we propose a "precedence" and "forced" segmentation-based approach, which should solve the second problem. Here, the main aim is to develop an approach, which is based on evaluation of precedence and a rule to force a segmentation point. During oversegmentation, we detect the human-recognized features in handwriting, such as loops, hat shape, valleys, and so forth, which are used to determine real segmentation points. The problem here is that we miss some segmentation points because of errors in feature detection. A method that sets a precedence to various features such as to set highest priority for a blank vertical line (space between two characters) with the next priority given to average character width (to assist in accurate segmentation point placement), and so forth, is developed. Based on the aforementioned precedences, the method is forced to segment. In this way, we do not neglect any suspected points, which are "real" segmentation points.

Finally, we propose a neural validation approach to remove incorrect segmentation points (third problem). This approach is based on three classifiers utilizing both multilayer perceptrons (MLPs) and support vector machines (SVMs). The success of neural-based techniques for numeral and character recognition (Chiang, 1998; Gader et al., 1997; Marinai et al., 2005; Verma et al., 2001) has provided the motivation for their use in the current context. The recent success in applying SVMs in the area of handwriting recognition justifies their use alongside neural-based techniques, in some cases outperforming neural networks (Liu & Fujisawa, 2005). The first classifier is trained with information from left and right strokes of a character. The second classifier is trained with descriptive information from the segmentation points themselves. The third classifier is trained with the compatibility of adjacent characters. The final scores are fused, and the segmentation points are removed or retained based on the final score (confidence of the fused network output).

In order to contend with the difficult problems inherent in accurately representing cursive character patterns, we propose a methodology to (1) simplify a character's contour or thinned representation, (2) allow the extraction of local features determined from the directions of identified strokes/line segments, and (3) global features

obtained through the analysis of a character's entire contour and dimensions (e.g., the width-to-height ratio).

It is our contention that the key to effectively extracting the most meaningful features from segmented/cursive characters is through the local and global analysis of a character's contour. Hence, in order to obtain these local and global features, we require that the image is preprocessed (Blumenstein et al., 2004) and a binary boundary retrieved. In the next step, it is necessary to trace the boundary, appropriately distinguishing individual strokes and determining appropriate direction values. This can be achieved by locating appropriate starting points and then investigating rules for determining the beginning and end of individual strokes. In this process, individual pixel directions are defined, and subsequently, a single value defining an individual stroke's direction is recorded.

The goal of simplifying a character's representation is to dispense with the problem of illegibility based on the difficult nature of cursive handwriting. The local information is extracted from the character's simplified representation to assist in the effective description of the character, to compress this information, and to facilitate the creation of a feature vector. It is proposed that this local information is extracted by zoning the character, processing the stroke data (i.e., encoding it from each zone) and subsequently storing it for later processing. Once the local features are obtained, complimentary global information is extracted.

The measurement of the physical location of each pixel in the simplified character boundary (obtained as mentioned in the previous paragraph) is obtained, which is then processed and recorded. In addition to this and in order to dispel the problem of ambiguity between character classes, the width-to-height ratio of each character is determined and stored. Other aspects of the character pattern also can be studied, such as the surface area and relative size. Hence, the output includes a global feature representation of the character's boundary along with additional information such as its width-to-height ratio, surface area, and relative size.

Once these subtasks are completed, an investigation of the local and global features on their own and as a single vector is required. A classifier based on MLP and SVM is used.

CONCLUSION AND FUTURE RESEARCH

In this chapter, a state of the art in handwriting recognition has been presented. A segmentation-based handwriting recognition technique and its components are described in detail, which will help graduate students, researchers, and technologists understand the handwriting recognition processes. A critical literature review of existing techniques and challenges in the area of handwriting recognition has been presented. A comparative performance of recent developments in the area, including accuracies on benchmark databases, is presented. Some novel strategies to improve segmentation-based handwriting recognition have also been presented. Future research will focus on the investigation and development of the presented strategies to improve segmentation accuracy and overall accuracies for general handwriting recognition systems.

REFERENCES

Arica, N., & Yarman-Vural, F.T. (2002). Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(6), 801–813.

Blumenstein, M., Liu, X.Y., & Verma, B. (2004). A modified direction feature for cursive character recognition. *Proceedings of the International Joint Conference on Neural Networks* (pp.

2983–2987). Budapest, Hungary: IEEE Computer Society Press.

Blumenstein, M., & Verma, B. (1999). Neural-based solutions for the segmentation and recognition of difficult handwritten words from a benchmark database. *Proceedings of the Fifth International Conference on Document Analysis and Recognition* (pp. 281–284). Bangalore, India: IEEE Computer Society Press.

Blumenstein, M., & Verma, B. (2001). Analysis of segmentation performance on the CEDAR benchmark database. *Proceedings of the Sixth International Conference on Document Analysis and Recognition* (pp. 1142–1146). Seattle: IEEE Computer Society Press.

Blumenstein, M., Verma, B., & Basli, H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 137–141). Edinburgh, UK: IEEE Computer Society Press.

Bozinovic, R.M., & Srihari, S.N. (1989). Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(1), 68–83.

Britto Jr., A. Sabourin, R., Bortolozzi, F., & Suen, C.Y. (2004). Foreground and background information in an HMM-based method for recognition of isolated characters and numeral strings. *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition* (pp. 371–376). Tokyo, Japan: IEEE Computer Society Press.

Brown, M.K., & Ganapathy, S. (1983). Preprocessing techniques for cursive script word recognition. *Pattern Recognition, 16*(5), 447–458.

Burges, C.J.C., Be, J.I., & Nohl, C.R. (1992). Recognition of handwritten cursive postal words using neural networks. *Proceedings of the 5th USPS Advanced Technology Conference* (pp. 117–124).

Cai, J., & Liu, Z.-Q. (1999). Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(3), 263–270.

Camastra, F., & Vinciarelli, A. (2003). Combining neural gas and learning vector quantization for cursive character recognition. *Neurocomputing, 51*, 147–159.

Cao, J., Ahmadi, M., & Shridhar, M. (1995). Recognition of handwritten numerals with multiple feature and multistage classifier. *Pattern Recognition, 28*(3), 153–159.

Casey, R.G., & Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(7), 690–706.

Chen, M.-Y., Kundu, A., Zhou, J., & Srihari, S.N. (1992). Off-line handwritten word recognition using hidden Markov model. *Proceedings of the 5th USPS Advanced Technology Conference* (pp. 563–579).

Chevalier, S., Geoffrois, E., Preteux, F., & Lemaltre, M. (2005). A generic 2D approach of handwriting recognition. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 489–493). Seoul, Korea: IEEE Computer Society Press.

Chiang, J.-H. (1998). A hybrid neural model in handwritten word recognition. *Neural Networks, 11*(2), 337–346.

Cho, S.-B. (1997). Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Transactions on Neural Networks, 8*(1), 43–53.

Davis, M. (2005). *Waiting in vain for the paperless office.* Butler Group, ZDNet UK posting.

Retrieved July 14, 2007, from http://news.zdnet.co.uk/itmanagement/0,1000000308,39223857,00.htm

Dimauro, G., Impedovo, S., Pirlo, G., & Salzo, A. (1997). Removing underlines from handwritten text: An experimental investigation. In A.C. Downton, & S. Impedovo (Eds.), *Progress in handwriting recognition* (pp. 497–501). World Scientific Publishing.

Dimauro, G., Impedovo, S., Pirlo, G., & Salzo, A. (1998). An advanced segmentation technique for cursive word recognition. In S.W. Lee (Ed.), *Advances in handwriting recognition* (pp. 255–264). World Scientific Publishing.

Ding, Y., Kimura, F., Miyake, Y., & Shridhar, M. (1999). Evaluation and improvement of slant estimation for handwritten words. *Proceedings of the 5th International Conference on Document Analysis and Recognition* (pp. 753–756). Bangalore, India: IEEE Computer Society Press.

Dunn, C.E., & Wang, P.S.P. (1992). Character segmentation techniques for handwritten text—A survey. *Proceedings of the 11th International Conference on Pattern Recognition* (pp. 577–580). The Hague, The Netherlands.

Eastwood, B., Jennings, A., & Harvey, A. (1997). A feature based neural network segmenter for handwritten words. In B. Verma, & X. Yao (Eds.). *Proceedings of the 1st International Conference on Computational Intelligence and Multimedia Applications* (pp. 286–290). Gold Coast, Australia.

Elliman, D.G., & Lancaster, I.T. (1990). A review of segmentation and contextual analysis techniques for text recognition. *Pattern Recognition*, *23*(3-4), 337–346.

Fan, X., & Verma, B. (2002). Segmentation vs. non-segmentation based neural techniques for cursive word recognition. *International Journal of Computational Intelligence and Applications*, *2*(4), 1–8.

Fujisawa, H., Nakano, Y., & Kurino, K. (1992). Segmentation methods for character recognition: From segmentation to document structure analysis. *Proceedings of the IEEE*, *80*(7), 1079–1092.

Gader, P.D., Mohamed, M., & Chiang, J.-H. (1997). Handwritten word recognition with character and inter-character neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, *27*(1), 158–164.

Gang, L., Verma, B., & Kulkarni, S. (2002). Experimental analysis of neural network based feature extractors for cursive handwriting recognition. *Proceedings of the IEEE World Congress on Computational Intelligence* (pp. 2837–2841). Hawaii: IEEE Computer Society Press.

Gatos, B., Pratikakis, I., Kesidis, A.L., & Perantonis, S.J. (2006). Efficient off-line cursive handwriting word recognition. *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition*.

Gilloux, M. (1993). Research into the new generation of character and mailing address recognition systems at the French post office research centre. *Pattern Recognition Letters*, *14*(4), 267–276.

Guillevic, D., & Suen, C.Y. (1994). Cursive script recognition: A sentence level recognition scheme. *Proceedings of the 4th International Workshop on the Frontiers of Handwriting Recognition* (pp. 216–223).

Günter, S., & Bunke, H. (2004). Feature selection algorithms for the generation of multiple classier systems and their application to handwritten word recognition. *Pattern Recognition Letters*, *25*(11), 1323–1336.

Günter, S., & Bunke, H. (2005). Off-line cursive handwriting recognition using multiple classifier systems—On the influence of vocabulary, ensemble, and training set size. *Optics and Lasers in Engineering*, *43*(3-5), 437–454.

Hanmandlu, M., Murali, K.R.M., Chakraborty, S., Goyal, S., & Choudhury, D.R. (2003). Unconstrained handwritten character recognition based on fuzzy logic. *Pattern Recognition*, *36*(3), 603–623.

Howe, N.R., Rath, T.M., & Manmatha, R. (2005). Boosted decision trees for word recognition in handwritten document retrieval. *Proceedings of the 28th Annual SIGIR Conference on Research and Development in Information Retrieval* (pp. 377–383).

Kapp, M.N., de Almendra Freitas, C., & Sabourin, R. (2007). Methodology for the design of NN-based month-word recognizers written on Brazilian bank checks. *Image Vision Computing*, *25*(1), 40–49.

Kim, G., Govindaraju, V., & Srihari, S.N. (1999). Architecture for handwritten text recognition systems. In S.W. Lee (Ed.), *Advances in handwriting recognition* (pp. 163–182). World Scientific Publishing.

Kimura, F., Shridhar, M., & Chen, Z. (1993). Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. *Proceedings of the 2nd International Conference on Document Analysis and Recognition* (pp. 18–22). Tsukuba, Japan: IEEE Computer Society Press.

Koerich, A.L., Britto, A., Oliveira, L.E.S., & Sabourin, R. (2006). Fusing high- and low-level features for handwritten word recognition. *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition*.

Koerich, A.L., Sabourin, R., & Suen, C.Y. (2003). Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis and Applications*, *6*(2), 97–121.

Koerich, A.L., Sabourin, R., & Suen, C.Y. (2005). Recognition and verification of unconstrained handwritten words. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1509–1522.

Lee, L., & Coelho, S. (2005). A simple and efficient method for global handwritten word recognition applied to Brazilian bankchecks. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 950–955). Seoul, Korea: IEEE Computer Society Press.

Liu, C.-L., & Fujisawa, H. (2005). Classification and learning for character recognition: Comparison of methods and remaining problems. *Proceedings of the International Workshop on Neural Networks and Learning in Document Analysis and Recognition* (pp. 5–7). Seoul, Korea: IEEE Computer Society Press.

Lu, Y. (1995). Machine printed character segmentation—An overview. *Pattern Recognition*, *28*(1), 67–80.

Lu, Y., & Shridhar, M. (1996). Character segmentation in handwritten words—An overview. *Pattern Recognition*, *29*(1), 77–96.

Madhvanath, S., Kleinberg, E., & Govindaraju, V. (1999). Holistic verification of handwritten phrases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(12), 1344–1356.

Marinai, S., Gori, M., & Soda, G. (2005). Artificial neural networks for document analysis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(1), 23–35.

Martin, G.L., Rashid, M., & Pittman, J.A. (1993). Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps. *International Journal on Pattern Recognition and Artificial Intelligence*, *7*(4), 831–847.

Nicchiotti, G., Scagliola, C., & Rimassa, S. (2000). A simple and effective cursive word segmentation method. *Proceedings of the 7th International Workshop on the Frontiers of Handwriting Recognition* (pp. 499–504).

Plamondon, R., & Srihari, S.N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(1), 63–84.

Schambach, M.-P. (2005). Fast script word recognition with very large vocabulary. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 9–13). Seoul, Korea: IEEE Computer Society Press.

Senior, A.W. (1994). *Off-line cursive handwriting recognition using recurrent neural networks* [unpublished doctoral dissertation]. Cambridge, England: University of Cambridge.

Singh, S., & Amin, A. (1999). Neural network recognition of hand printed characters. *Neural Computing & Applications, 8*(1), 67–76.

Srihari, S.N. (1993). Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern Recognition Letters, 14*(4), 291–302.

Srihari, S.N. (2006). Automatic handwriting recognition. *Encyclopedia of Language & Linguistics, 2nd Edition*. Elsevier.

Steinherz, T., Rivlin, E., & Intrator, N. (1999). Offline cursive script word recognition—A survey. *International Journal of Document Analysis and Recognition, 2*, 90–110.

Suen, C.Y., Legault, R., Nadal, C., Cheriet, M., & Lam, L. (1993). Building a new generation of handwriting recognition systems. *Pattern Recognition Letters, 14*(4), 305–315.

Suen, C.Y., & Tan, J. (2005). Analysis of errors of handwritten digits made by a multitude of classifiers. *Pattern Recognition Letters, 26*(3), 369–379.

Verma, B. (2003). A contour code feature based segmentation for handwriting recognition. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 1203–1207). Edinburgh, UK: IEEE Computer Society Press.

Verma, B., Blumenstein, M., & Ghosh, M. (2004). A novel approach for structural feature extraction: Contour vs. direction. *Pattern Recognition Letters, 25*(9), 975–988.

Verma, B., Blumenstein, M., & Kulkarni, S. (1998). Recent achievements in offline handwriting recognition. *Proceedings of the 2nd International Conference on Computational Intelligence and Multimedia Applications* (pp. 27–33). Melbourne, Australia: World Scientific Publishing Company.

Verma, B., Gader, P., & Chen, W. (2001). Fusion of multiple handwritten word recognition techniques. *Pattern Recognition Letters, 22*(9), 991–998.

Viard-Gaudin, C., Lallican, P.-M., & Knerr, S. (2005). Recognition-directed recovering of temporal information from handwriting images. *Pattern Recognition Letters, 26*(16), 2537–2548.

Vinciarelli, A. (2002). A survey on off-line cursive word recognition. *Pattern Recognition, 35*(7), 1433–1446.

Vinciarelli, A., Bengio, S., & Bunke, H. (2003). Offline recognition of large vocabulary cursive handwritten text. *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 1101–1107). Edinburgh, UK: IEEE Computer Society Press.

Wang, X., Ding, X., & Liu, C. (2005). Gabor filters-based feature extraction for character recognition. *Pattern Recognition, 38*(3), 369–379.

Wen, Y., Lu, Y., & Shi, P. (2007). Handwritten bangla numeral recognition system and its application to postal automation. *Pattern Recognition, 40*(1), 99–107.

Xiao, X., & Leedham, G. (2000). Knowledge-based cursive script segmentation. *Pattern Recognition Letters, 21*(10), 945–954.

Xu, Q., Lam, L., & Suen, C.Y. (2003). Automatic segmentation and recognition system for handwritten dates on Canadian bank cheques. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 704–709). Edinburgh, UK: IEEE Computer Society Press.

Yanikoglu, B., & Sandon, P.A. (1998). Segmentation of off-line cursive handwriting using linear programming. *Pattern Recognition*, *31*(12), 1825–1833.

Chapter II
Elastic Matching Techniques for Handwritten Character Recognition

Seiichi Uchida
Kyushu University, Japan

ABSTRACT

This chapter reviews various elastic matching techniques for handwritten character recognition. Elastic matching is formulated as an optimization problem of planar matching, or pixel-to-pixel correspondence, between two character images under a certain matching model, such as affine and nonlinear. Use of elastic matching instead of rigid matching improves the robustness of recognition systems against geometric deformations in handwritten character images. In addition, the optimized matching represents the deformation of handwritten characters and thus is useful for statistical analysis of the deformation. This chapter argues the general property of elastic matching techniques and their classification by matching models and optimization strategies. It also argues various topics and future work related to elastic matching for emphasizing theoretical and practical importance of elastic matching.

INTRODUCTION

In handwritten character recognition, it is important to tackle geometric deformations of characters. The geometric deformations are classified into the following four types: fluctuation of stroke thickness, linear deformations (e.g., translation, scaling, shear, and rotation), nonlinear and topology-preserving deformations, and deformations changing topology. Those deformations will be caused by many factors; for example, writing material, writer's habit, writing speed, writing style (especially cursive style), character size, inherent character shape, and noise and geometric transformation at character image acquisition.

The purpose of this chapter is to overview various elastic matching (EM) techniques for handwritten character recognition. EM is also called deformable template (Trier, Jain & Taxt, 1996), flexible matching (Mori, Yamamoto &

Yasuda, 1984), or nonlinear template matching (Mori, Suen & Yamamoto, 1992), and has been developed by many researchers to tackle the geometric deformations. EM has been employed in not only handwritten character recognition but also many other image pattern matching problems, such as face recognition, fingerprint recognition, gesture recognition, medical image analysis, automatic image morphing, computer vision (e.g., stereo), and motion analysis. For more general surveys, see Glasbey and Mardia (1998), Jain, Zhong, and Dubuisson-Jolly (1998), Lester and Arridge (1999), Redert, Hendriks, and Biemond (1999), and Zitová and Flusser (2003).

EM is formulated as an optimization problem of planar matching, or 2D-2D mapping between two character images, **A** and **B**. From another viewpoint, EM treats a character image **A** like a "rubber sheet"' and fits it to another character image **B** as closely as possible. Hereafter, this 2D-2D mapping from **A** to **B** is called 2D warping (2DW). Note that we can consider EM based on 1D-2D mapping, where a 1D-stroke model is fitted to input image, although this chapter mainly concerns EM techniques based on 2DW. In a later section, we will briefly review these EM techniques based on 1D-2D mapping.

For handwritten character recognition, EM possesses two merits. The first merit is that the distance evaluated under the optimized 2DW is deformation-invariant. Thus, by using the EM distance as a discriminant function, we can realize character recognition systems robust to the geometric deformations. The range of the invariance depends on the definition of 2DW; that is, the more flexible 2DW becomes, the more invariant the EM distance becomes. Thus, EM has a potential to provide more intuitive and robust recognition frameworks than other deformation-invariant techniques, such as invariant feature (e.g., the horizontal projection profile by Nakata, Nakano, and Uchikura (1972)) and shape normalization (Lee & Park, 1994; Liu, Nakashima, Sako, & Fujisawa, 2004).

The second merit is that the optimized 2DW itself describes the deformation of subjected characters. This fact shows that EM possesses a useful property of structural analysis techniques. Furthermore, EM can be linked to statistical and stochastic frameworks by the merit. Active shape models (Cootes, Taylor, Cooper & Graham, 1995; Shi, Gunn, & Damper, 2003; Uchida & Sakoe, 2003a) and (pseudo-) 2D HMMs (Agazzi, Kuo, Levin, & Pieraccini, 1993; Kuo & Agazzi, 1994; Levin & Pieraccini, 1992; Park & Lee, 1998) are two good examples.

The remaining part of this chapter is organized as follows. First, EM is formulated as an optimization problem of 2D-2D mapping (i.e., 2DW). General properties of EM and the EM distance are also described. Second, EM techniques are classified according to their specific formulations of 2DW and optimization strategies. It will be emphasized that there is a strong relation between the formulation and the optimization strategy. Third, several related topics are discussed, such as incorporation of category-dependent deformation tendency. Fourth, EM techniques based on 1D-2D mapping are briefly reviewed, which is another type of EM used in handwritten character recognition. Finally, conclusions are presented after listing various future tasks for EM.

OUTLINE OF ELASTIC MATCHING

Formulation of EM

As described before, EM is formulated as an optimization problem of 2D-2D mapping (i.e., 2DW) between two character images, **A** and **B**. Let $\mathbf{a}_{i,j}$ and $\mathbf{b}_{x,y}$ denote pixel values (e.g., intensity values) or pixel feature vectors (e.g., RGB vectors) at pixel (i,j) on **A** and (x,y) on **B**, respectively. While we can deal with the matching between two images of arbitrary sizes, we hereafter assume $N \times N$ images for simplicity.

Let **F** denote 2DW from **A** to **B** (i.e., $\mathbf{F}:(i,j)\mapsto(x,y)$). The 2DW **F** represents the pixel-to-pixel correspondence between **A** and **B** as shown in Figure 1. Using **F**, we can consider $\mathbf{B}_\mathbf{F}=\{\mathbf{b}_{\mathbf{F}(i,j)}\}$ which is the deformed image of **B** by the 2DW **F**. If **F** is a topology-preserving 2DW, $\mathbf{B}_\mathbf{F}$ undergoes rubber-sheetlike deformations.

EM is formulated as the minimization problem of the following objective function with respect to **F**:

$$J_{\mathbf{A},\mathbf{B}}(\mathbf{F})=D(\mathbf{A},\mathbf{B}_\mathbf{F})=\sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\mathbf{a}_{i,j}-\mathbf{b}_{\mathbf{F}(i,j)}\right\|,$$

where $D(\mathbf{A}, \mathbf{B}_\mathbf{F})$ is a simple "rigid"' distance between **A** and $\mathbf{B}_\mathbf{F}$, and $\|\cdot\|$ denotes a distance metric between two pixel feature vectors, such as Euclidean distance and absolute distance.

The *EM distance* $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$ is obtained as the solution of the previous minimization problem; that is,

$$D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})=\min_{\mathbf{F}} J_{\mathbf{A},\mathbf{B}}(\mathbf{F})=J_{\mathbf{A},\mathbf{B}}(\tilde{\mathbf{F}}),$$

where $\tilde{\mathbf{F}}$ denotes the optimal **F**, which minimizes $J_{\mathbf{A},\mathbf{B}}(\mathbf{F})$. Clearly, $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})=D(\mathbf{A},\mathbf{B}_{\tilde{\mathbf{F}}})$. That is, EM distance is the rigid distance between **A** and **B** after fitting **B** to **A** as closely as possible.

Figure 2 shows two examples of EM results (Uchida & Sakoe, 1998), where the 2DW $\tilde{\mathbf{F}}$ is represented as a deformed mesh. If the optimal 2DW $\tilde{\mathbf{F}}$ shows correct correspondence, it represents the deformation of **B** relative to **A**.

Recognizers based on the EM distance $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$ are expected to be robust to geometric deformations. This is because $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$ is

Figure 1. 2DW defined between two handwritten character images (© copyright 2005IEICE)

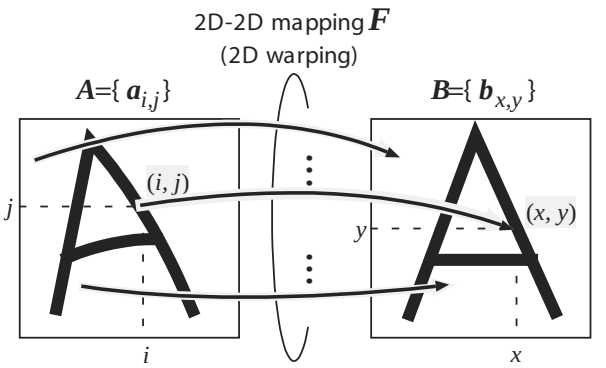
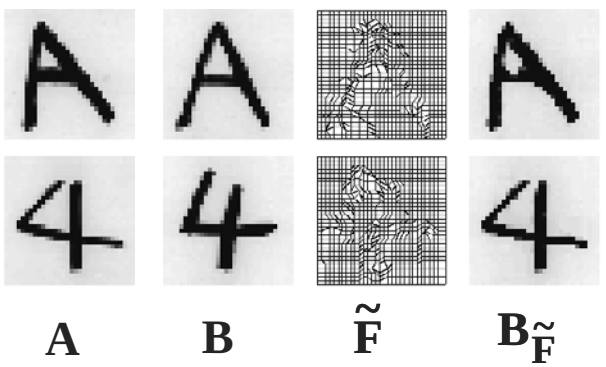


Figure 2. Examples of EM results



invariant to the deformations compensable by **F**. For example, if the 2DW **F** is defined as an affine transformation, $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$ is theoretically invariant to any affine transformed version of **B**.

Although past literatures have reported the usefulness of the EM distance on its robustness to the deformations, the EM-based recognizers often suffer from *overfitting,* which is the phenomenon that the distance between similar-shaped characters of different categories is underestimated. For example, if **F** is an affine transformation, $D_{\mathrm{EM}}(\text{"9"},\text{"6"})$ becomes a very small value because the affine transformation can compensate 180° rotation. Thus, the input "9" may be misrecognized to the different category "6". Generally, there is a trade-off between the ability of compensating deformations (i.e., the flexibility of **F**) and the risk of overfitting.

Anisotropy is another important aspect of the EM distance $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$. Unlike Euclidean distance, the set of patterns equidistant from a certain pattern do not form a hypersphere. Consequently, the centroid (the center of gravity) of a set of patterns may not be placed around the center of their distribution in Euclidean space. Figure 3 (Matsumoto, Uchida, & Sakoe, 2004) shows an experimental result of deriving a centroid for a set of patterns. The small dots are actual handwritten numeral patterns of a category, and the black triangle is their centroid. They are displayed in the 2D subspace spanned by their first two principal axes. When the Euclidean distance is used, the centroid is placed around the center of pattern distribution (Figure 3(a)). In contrast, when the EM distance is used, the centroid is not placed around the center (Figure 3(b)).

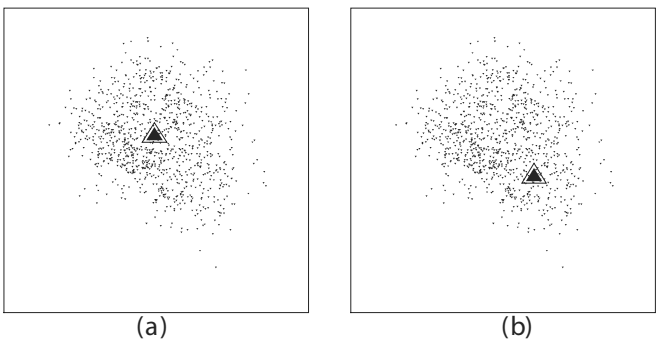
The EM distance is often asymmetric; that is, $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B}) \neq D_{\mathrm{EM}}(\mathbf{B},\mathbf{A})$, and therefore just a pseudodistance metric. This asymmetric property comes from the fact that the 2DW defined previously is not a bijective mapping. If symmetric property is necessary, one may simply use $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B}) + D_{\mathrm{EM}}(\mathbf{B},\mathbf{A})$ instead of $D_{\mathrm{EM}}(\mathbf{A},\mathbf{B})$. A more plausible solution is the use of a bijective 2DW; in this case, not only **B** but also **A** are deformed by the 2DW, and the optimization of the bijective 2DW tends to be a complicated one. As reported in past literatures, this asymmetric property is not crucial for the recognition performance.

CLASSIFICATION OF ELASTIC MATCHING TECHNIQUES

In this section, EM techniques are classified into several types. The classification can be done by two factors: the specific formulation of 2DW and the optimization strategy of 2DW. Figure 4 (Uchida & Sakoe, 2005) shows a classification tree based on those factors.

The first factor affects the range of compensable deformations. While 2DW **F** was formulated

Figure 3. The centroids under (a) Euclidean distance and (b) EM distance (© copyright 2005IEICE)



as a general 2D-2D mapping in the previous section, specific formulations are necessary for individual tasks. As shown in Figure 4, the EM techniques can be divided roughly into two classes by the 2DW formulation: parametric 2DW-based EM and nonparametric 2DW-based EM. In parametric 2DW, each variable does not represent pixel correspondence but represents a parameter that controls 2DW indirectly. In nonparametric 2DW, each variable that controls **F** directly represents pixel correspondence.

On the formulation of 2DW, it is important to consider the deformation characteristics of handwritten characters. For example, when we can assume that handwritten characters mainly undergo rubber-sheetlike deformations, topology-preserving 2DW is a natural choice. Both parametric 2DW and nonparametric 2DW are further classified into several classes, depending on this consideration.

The second factor, the optimization strategy of 2DW, affects the accuracies of the results of EM; namely, the accuracies of the minimized distance and the optimized 2DW. Generally, optimization strategies for globally optimal solutions will provide more accurate results than those for suboptimal solutions.

As shown in Figure 4, each class by the formulation of 2DW is closely related to several optimization strategies. In other words, possible optimization strategies are almost determined according to the formulation.

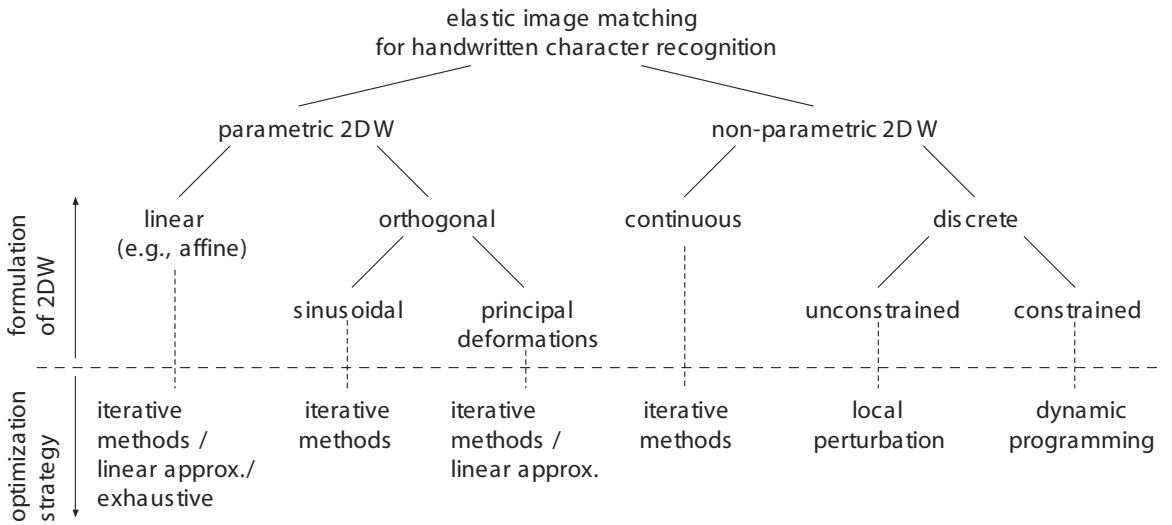
We should note that those two factors mutually affect the computational complexity of EM. For example, strategies for globally optimal solutions will require more computations than those for suboptimal solutions in general. The effect of the formulation of 2DW on computational complexity is more complicated. In the following, the computational complexity of each EM technique is discussed carefully.

Parametric 2DW

Linear 2DW

Among parametric 2DWs, linear 2DW is the simplest and the most common one. Linear 2DW $\mathbf{F}:(i, j)\mapsto(x, y)$ is generally formulated as:

Figure 4. Classification of planar EM techniques for handwritten character recognition (© copyright 2005IEICE)



$$(x, y) = (\alpha_1 i + \alpha_2 j + \alpha_3, \alpha_4 i + \alpha_5 j + \alpha_6),$$

where α_1, α_2, …, and α_6 are real-valued parameters that control **F**. If $\alpha_1 = \alpha_5 = 1$ and $\alpha_2 = \alpha_4 = 0$, this linear 2DW will compensate translation only. If $\alpha_2 = \alpha_3 = \alpha_4 = \alpha_6 = 0$, the 2DW will compensate horizontal and vertical scaling. If all of the six parameters are controllable, **F** becomes the affine transformation.

Although the formulation of linear 2DW is very simple, its optimization problem becomes a complicated one. In fact, since the parameters to be optimized are involved in a nonlinear image pattern function **B**, the optimization problem also becomes nonlinear. Thus, EM techniques based on linear 2DW have employed iterative solutions or approximate solutions.

An example of the iterative solution of linear 2DW is done by Wakahara and colleagues (Wakahara, Kimura, & Tomono, 2001; Wakahara, & Odaka, 1998). They have employed affine transformation as 2DW. The optimization problem of the parameters α_1, α_2, …, and α_6 is approximated as a linear problem by a successive iteration method.

The tangent distance method (Keysers, Dahmen, Theiner, & Ney, 2000; Simard, Le Cun, & Denker, 1993) has been proposed as another linear 2DW-based EM where a nonlinear optimization problem of linear 2DW is approximated as a linear problem by Taylor series expansion. Specifically, the tangent distance method assumes a nonlinear manifold that contains all deformed character patterns of a certain category. Then, this manifold is approximated to be its tangent plane. Consequently, the EM problem is reduced to the minimum distance problem between the tangent plane and an input pattern; that is, a linear problem. Recently, this idea was successfully linked with statistic framework (Keysers, Macherey, Ney, & Dahmen, 2004).

A more straightforward solution is also possible by using an exhaustive search strategy. Yasuda, Yamamoto, and Yamada (1997) proposed the perturbed correlation method, where a 2D reference pattern is "perturbed" by a huge number of affine transformations. Each perturbed pattern is rigidly matched with a 2D input pattern, and the best perturbed pattern is searched for. Since the number of possible parameter values becomes very large, this method requires numerous and repetitive 2D-2D rigid matching. Recent hardware, however, makes the method a computationally tractable one. A similar method can be found in Ha and Bunke (1997).

Perspective transformation is another important class of linear transformation. Nowadays, camera-based character recognition is widely investigated (Liang, Doermann & Li, 2005), and one of its main hurdles is how to compensate the perspective transformation due to an oblique camera angle. In past researches, perspective transformation has not been compensated by 2DWs at individual characters of a document; instead, it has been compensated by a preprocessing technique, called dewarping, for the entire document image. When camera-based character recognition tackles "rumpled" documents, EM should be applied to individual characters for compensating perspective transformation.

Orthogonal 2DW

In several EM techniques, 2DW **F** is represented as a linear combination of orthogonal functions:

$$(x, y) = \sum_{k=1}^{K} \alpha_k \varphi_k (x, y), \qquad (1)$$

where φ_1, ..., φ_k, ..., φ_K are orthogonal 2D-2D functions; that is, $\langle \varphi_k, \varphi_l \rangle = 0$ for $k \neq l$, and $\alpha_1, \ldots, \alpha_k, \ldots, \alpha_K$ are parameters to be optimized.

One of the most reasonable choices of $\{\varphi_k\}$ will be orthogonal sinusoids. Jain and Zongker (1997) have proposed a sinusoid-based 2DW and optimized its gain parameters $\{\alpha_k\}$ by a natural coarse-to-fine strategy; the parameters of low-frequency sinusoids are first determined

and fixed by the gradient descent method (i.e., an iterative method), and then the parameters of high-frequency sinusoids are determined in a similar way.

The active shape model (ASM) proposed by Cootes, et al. (1995) is a statistical deformation model defined as a linear combination of orthogonal deformations that are provided by applying principal component analysis (PCA) to actual deformations collected from training patterns. Inspired by ASM, Shi et al. (2003) have proposed a 1D-2D EM technique for the character recognition task. In this work, a linear (e.g., 1D) reference pattern is fitted to a 2D input pattern. The fitting is governed by the principal deformations of the 1D reference pattern and evaluated by the chamfer distance. The optimal fitting that minimizes the chamfer distance is searched for by a gradient descent method. A related idea can be found in Kimura, Yoshimura, Miyake, and Ichikawa (1970), where the displacement between the strokes of two skeletonized handwritten characters is evaluated by the Maharanobis distance.

Uchida and Sakoe (2003a) have extended ASM to fully 2D-2D EM (i.e., planar EM) and applied to handwritten character recognition. Figure 5 shows the reference pattern of "A" deformed by applying the principal deformations $\varphi_1,\varphi_2,\varphi_3$ (called the eigen-deformations in Uchida and Sakoe, 2003a) of "A" positively or negatively. The first eigen-deformation represents the slant deformation, and the second represents the vertical shift of the horizontal stroke. It is interesting to note that those eigen-deformations were estimated from the deformations collected as a result of nonparametric EM, as noted later.

Uchida and Sakoe (2003b) have combined the previously mentioned ASM technique with the tangent distance method. In this strategy, the eigen-deformations of Figure 5 are converted into the tangent vectors of by the Taylor series expansion. Those tangent vectors will span the tangent plane of the manifold, which contains all the patterns realized by the ASM. Those tangent vectors are then used as $\{\varphi_k\}$ in (1), while they are not orthogonal.

Figure 5. Top three eigen-deformations of "A"

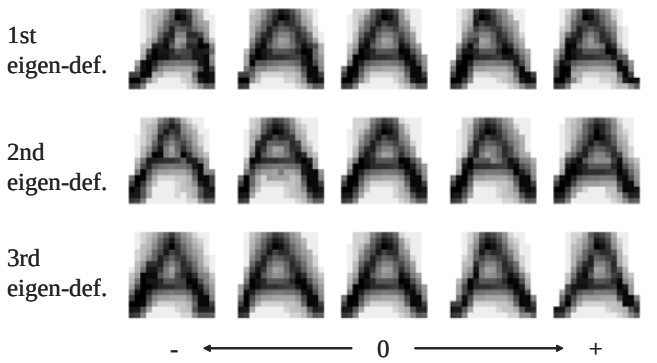


Figure 6. Tangent vectors for the top three eigen-deformations of Figure 5.

Nonparametric 2DW

Nonparametric 2DW is treated as a set of individual pixel-to-pixel correspondences between two images. This implies that nonparametric 2DW is controlled more directly than parametric 2DW and therefore more flexible than parametric 2DW. The nonparametric 2DWs can be divided into two classes: continuous 2DW and discrete 2DW.

Continuous 2DW

Continuous (and nonparametric) 2DW is formulated as a mapping $\mathbf{F}:(i,j)\in\Re^2\mapsto(x,y)\in\Re^2$; that is, a 2D-2D function. Consequently, the optimization problem of the continuous 2DW is often formulated as a variational problem; a continuous 2DW **F** is considered an argument function of the underlying variational problem, and $J_{\mathbf{A},\mathbf{B}}(\mathbf{F})$ is the functional to be minimized.

The variational problem often is solved by the deterministic relaxation (Sakaue, Amano & Yokoya, 1999). Specifically, the Euler-Lagrange equation is first derived and discretized to obtain a system of nonlinear equations. Then a suboptimal 2DW is obtained by solving the equations by some iterative method, such as the Gauss-Seidel method. Mizukami (1998) has successfully applied the deterministic relaxation to handwritten character recognition.

Regularization techniques and/or coarse-to-fine strategies are necessary for solving the variational problem based on the deterministic relaxation. This is because the Euler-Lagrange equation is only a necessary condition for the optimal solution of the variational problem. In addition, since the system of equations is nonlinear, the Gauss-Seidel method cannot guarantee even its convergence. Mizukami (1998) has employed a regularization technique and a careful coarse-to-fine strategy.

Another interesting approach is to relate continuous 2DW to physical phenomena. Webster and Nakagawa (1997) and Nakagawa, Yanagida, and Nagasaki (1999) have proposed a motion equation-based EM technique. In their techniques, an elastic membrane created from **B** is falling into a potential field created from **A**. The membrane showing $\mathbf{B}_\mathbf{F}$ is updated iteratively by calculating its motion equation until an equilibrium state. For EM of medical images, we can find different physical formulations (Bajscy & Broit, 1982, Christensen, Rabbitt & Miller, 1996).

As already seen, continuous 2DW has often been assumed as a differentiable function and optimized by some iterative optimization strategy. In this sense, continuous 2DW is similar to several parametric 2DWs.

Discrete and Unconstrained 2DW

The discrete 2DW is formulated as a set of $2N^2$ variables:

$$\left((x_{1,1},y_{1,1}),\ldots,(x_{i,j},y_{i,j}),\ldots,(x_{N,N},y_{N,N})\right)$$

where $(x_{i,j},y_{i,j})$ denotes the pixel on **B** corresponding to the pixel (i,j) on **A**. The discrete 2DW is further divided into two classes: unconstrained 2DW and constrained 2DW. In unconstrained 2DW, the mapping of the pixel (i,j); that is, $(x_{i,j},y_{i,j})$, is independent of the mapping of other pixels.

Since there is no constraint among pixels, it is possible to determine the optimal $(x_{i,j},y_{i,j})$ for each pixel (i,j) independently. This pixel-independent optimization strategy is called local perturbation (Burr, 1981, Hattori, Watanabe, Sanada & Tezuka, 1983; Izui, Harashima & Miyagawa, 1985; Liolios, Kavallieratou, Fakotakis & Kokkinakis, 2002; Meguro & Umeda, 1978; Saito, Yamada & Yamamoto, 1982; Yamada, Saito & Mori, 1981), or image distortion model (Keysers et al., 2000; Keysers, Gollan & Ney, 2004). For each pixel (i,j) on **A**, its best corresponding pixel $(x_{i,j},y_{i,j})$ on **B** is searched for locally and independently. The great merit of this simplest optimization strategy is its far less complexity than other optimization strategies.

Local perturbation, however, possesses a weak point: the resulting 2DW becomes jaggy due to the noise and the ambiguity in pixel features. Thus, careful coarse-to-fine strategies (Izui et al., 1978), smoothing of local displacements (Burr, 1981; Hattori et al., 1983), sequential (outside-to-inside) optimization with mild constraints (Hattori et al., 1983), and/or sophisticated pixel features (Keysers et al., 2004; Saito et al., 1982) will be indispensable to expect sufficient performance.

Discrete and Constrained 2DW

In discrete and constrained 2DW, the mapping of the pixel (*i,j*) is constrained by the mapping of adjacent pixels of (*i,j*) for regulating flexibility. For example, continuity constraints, such as $|x_{i,j} - x_{i-1,j}| \leq \Delta xi$ where Δxi is a positive small constant, are often imposed on 2DW to exclude large gaps from 2DW. The four parameters Δxi, Δxj, Δyi, and Δyj of Figure 7 are often used for specifying the constraints; that is, for specifying the flexibility of the 2DW. Monotonicity constraints such as $x_{i,j} - x_{i-1,j} \geq 0$ are also popular constraints to exclude foldover from 2DW.

According to the constraints imposed on the adjacent pixels, previous discrete and constrained 2DWs can be classified into DP1-DP5 of Figure 8. Each node of the graph on **B** represents the pixel $(x_{i,j}, y_{i,j})$ corresponding to a pixel (*i,j*) on **A**. Each link represents the dependency between the mappings of adjacent pixels on **A** (e.g., the dependency between $(x_{i,j}, y_{i,j})$ and $(x_{i-1,j}, y_{i-1,j})$). As shown in the figure, it is often assumed that all 2DWs are restricted by boundary constraints that any boundary pixel of **A** corresponds to a boundary pixel of **B**.

Note that a discrete and constrained 2DW other than DP1-DP5 has been proposed in Moore (1979), Tanaka (1985), and Wu, Liu, and Chang (1995). This 2DW is optimized diagonally from one corner (1,1) to its opposite corner (*N,N*) while extending a rectangular region being processed. Despite its computational feasibility, its flexibility

Figure 7. Parameters for specifying the constraints of 2DW

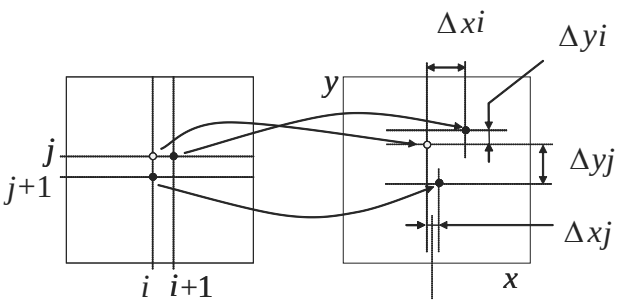


Figure 8. Several types of discrete and constrained EM

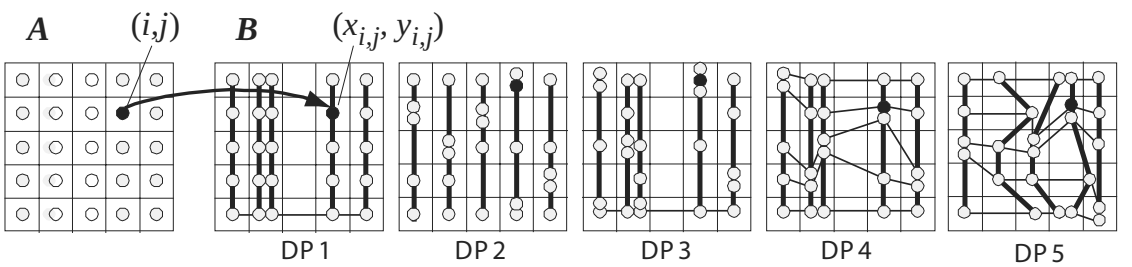


Table 1. Property of discrete and constrained EMs, DP1-DP5

	Constraints	**Computational Complexity by DP**	**Note**
DP1	$\begin{cases} 0 \le x_{i,j} - x_{i-1,j} \le \Delta_{xi}, \\ x_{i,j} = x_{i,j-1}, \\ y_{i,j} = j. \end{cases}$	Polynomial $O\left(N^2\left(N+\Delta_{xi}\right)\right)$ $\approx O\left(N^3\right)$	
DP2	$\begin{cases} x_{i,j} = i, \\ 0 \le y_{i,j} - y_{i,j-1} \le \Delta yj. \end{cases}$	Polynomial $O\left(N^3\Delta_{yj}\right)$	
DP3	$\begin{cases} 0 \le x_{i,j} - x_{i-1,j} \le \Delta_{xi}, \\ x_{i,j} = x_{i,j-1}, \\ 0 \le y_{i,j} - y_{i,j-1} \le \Delta_{yj}. \end{cases}$	Polynomial $O\left(N^2\left(N^2\Delta_{yj}+\Delta_{xi}\right)\right)$ $\approx O\left(N^4\Delta_{yj}\right)$	Often referred to as "pseudo-2D"
DP4	$\begin{cases} 0 \le x_{i,j} - x_{i-1,j} \le \Delta_{xi}, \\ x_{i,j} = x_{i,j-1}, \\ \left\lvert y_{i,j} - y_{i-1,j} \right\rvert \le \Delta_{yi}, \\ 0 \le y_{i,j} - y_{i,j-1} \le \Delta_{yj}. \end{cases}$	Exponential $O\left(N^2\Delta_{yj}^N\left(N+\Delta_{xi}\Delta_{yi}^N\right)\right)$ $\approx O\left(N^2\Delta_{xi}\left(\Delta_{yi}\Delta_{yj}\right)^N\right)$	
DP5	$\begin{cases} 0 \le x_{i,j} - x_{i-1,j} \le \Delta_{xi}, \\ \left\lvert x_{i,j} - x_{i,j-1} \right\rvert \le \Delta_{xj}, \\ \left\lvert y_{i,j} - y_{i-1,j} \right\rvert \le \Delta_{yi}, \\ 0 \le y_{i,j} - y_{i,j-1} \le \Delta_{yj}. \end{cases}$	Exponential $O\left(N^2\left(\Delta_{xj}\Delta_{yj}\right)^N\left(N+\left(\Delta_{xi}\Delta_{yi}\right)^N\right)\right)$ $\approx O\left(N^2\left(\Delta_{xi}\Delta_{xj}\Delta_{yi}\Delta_{yj}\right)^N\right)$	Truly 2D

does not seem to match actual deformations of handwritten characters.

Table 1 summarizes the constraints specifying the 2DWs, DP1-DP5. DP1-DP4 are specified by asymmetric constraints; the constraints in vertical direction are different from those in horizontal direction. Thus, DP1-DP4 are not fully two-dimensional 2DWs and cannot compensate truly 2D deformations such as rotation and slant. This fact can be understood from the fact that all the pixels on each column of **A** are mapped together to the same column of **B**. In contrast, DP5 is specified by symmetric constraints and thus fully two-dimensional. Note that every type except DP5 has its transposed (i.e., rotated 90 degrees) version.

Table 1 also summarizes the computational complexity to obtain the optimal 2DW when dynamic programming (DP) is used as the optimization strategy. DP has been employed for many discrete and constrained 2DWs because of its useful properties. DP can guarantee globally optimal 2DW and be free from numerical errors. Those properties imply high accuracy of the 2DW optimized by DP. In addition, DP accepts undifferentiable objective functions, position-dependent constraints, and various pixel features. Furthermore, DP framework can be readily extended to be HMM. (Note that HMM and DP are not distinguished in this section unless otherwise mentioned.) Those versatility and extension abilities are other useful properties of DP.

DP1 is the simplest 2DW and can compensate simple (and intrinsically 1D) deformations where all the pixels of each column are shifted equally and horizontally. DP1 itself has been employed in word recognition (Cho, Lee & Kim, 1995; Makhoul, Schwaltz, Lapre & Bazzi, 1998; Mohamed & Gader, 1996; Shiku, Nakamura, Kuroda & Miyahara, 2000) rather than isolated handwritten character recognition.

While the optimization of DP1 requires fewer computations than others, DP1 cannot compensate for any vertical deformation. Thus, DP1 is often repeated to compensate for deformations that are more complex. Nakano, Nakata, Uchikura, and Nakajima (1973) have proposed a DP-based EM technique where the vertical version of DP1 is optimized after the horizontal version is optimized. Hallouli, Likforman-Sulem, and Sigelle (2002) have compared several combinations of vertical and horizontal versions of DP1 in the framework of HMM. In Nishimura, Tsutsumi, Maruyama, Miyao, and Nakano (2001), DP1 is repeated four times under different feature vectors having different roles on representing spatial distribution of strokes. Wang, Brakensiek, Kosmala, and Rigoll (2001) first use a horizontal DP1 for segmenting a handwritten word into its component characters, and second use a vertical DP1 for compensating the vertical deformation of each of those component characters.

DP2 is comprised of independent one-dimensional vertical warpings. DP2 cannot compensate for any horizontal deformation. Thus, DP2 is often repeated like DP1. In Isomichi and Ogawa (1975), the vertical version of DP2 is optimized after the horizontal version is optimized. Tsukumo (1992) has proposed a smart technique where a blurring operation is employed to complement the compensation ability of DP2.

DP3 is the most popular 2DW among discrete and constrained 2DWs. DP3 can compensate both vertical and horizontal deformations simultaneously with polynomial-order computations. The HMM version of DP3 is so-called Pseudo-2D HMM (Agazzi et al., 1993; Kuo & Agazzi, 1994; Levin & Pieraccini, 1992) and widely used in recognizing handwritten characters (Levin & Pieraccini, 1992), machine-printed words (Agazzi et al., 1993; Kuo & Agazzi, 1994; Yen, Kuo, & Lee, 1999), and handwritten words (Bippus & Märgner, 1999).

DP3 has been extended by Keysers, et al. (2004). Their 2DW allows columnwise local perturbation on the 2DW given by DP3. This extended DP3 can provide truly 2D warping with a feasible amount of computations. Large perturbation should be used carefully because it may lose continuity and monotonicity.

DP4 can realize a topology-preserving 2DW since it is constrained in both vertical and horizontal directions at each pixel. Thus, DP4 can avoid the overfitting of "P"' to "b"' by the discontinuity between two strokes. (This overfitting can happen in DP2 and DP3.)

The computational amount of DP4 is an exponential order of N and thus far larger than DP1-DP3. This is because it is impossible to optimize the mapping of each column independently; that is, the mutual relation between the mappings of adjacent columns should be considered during the optimization. DP4 is a restricted version of DP5, and therefore, its algorithm can be derived easily from DP5 (Uchida & Sakoe, 1998, 1999).

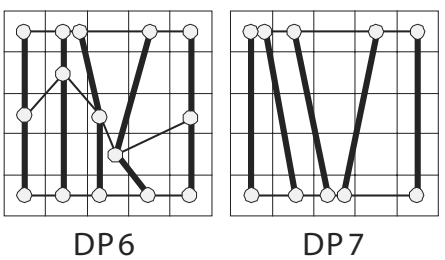
DP5, which is a truly 2D 2DW, was originally proposed by Levin and Pieraccini (1992). In their technique, Δxi, Δxj, Δyi, and Δyj are set at their maximum value, N. Thus, their 2DW can preserve upper/lower and left/right relationships (i.e., vertical and horizontal monotonicity) and does not care about continuity of character patterns. In other words, their 2DW allows large discontinuities and thus is not a topology-preserving 2DW. (Theoretically, their 2DW can map "A" to "H" by losing the continuity around the top of "A.") Inspired by Levin and Pieraccini (1992), Uchida and Sakoe (1998, 1999) have proposed a topology-preserving and truly 2D 2DW (i.e., truly rubber-sheet EM), where $\Delta xi = \Delta yj = 2$ and $\Delta xj = \Delta yi = 1$.

Although DP5 has a good potential to realize truly rubber-sheet EM, its optimization of DP5 is an NP-hard problem (Keysers & Unger, 2003) and thus requires exponential-order computations. Even if character images are small ($N \approx 20$), it is impossible to obtain the globally optimal 2DW. Thus, some approximation should be introduced for the practical use of DP5. In Uchida and Sakoe (1998, 1999), beam search is incorporated into the DP optimization process to obtain a suboptimal 2DW with fewer computations. One can employ other local search-based approximation algorithms for DP-based EM. In Sugimura, Iiguni, and Adachi (1997), the optimization of DP5 is performed in a sequential (i.e., column-by-column) and greedy manner. In Chen and Willson (2000), this sequential and greedy optimization is iterated to refine the result. In Quénot (1992), the iteration proceeds alternately in horizontal and vertical directions. Uchida and Sakoe (2000a) have proposed an approximation algorithm that exploits the fact that the global optimization by DP can be done very fast if an image pattern is an elongated one.

Hybrid Between Parametric 2DW and Nonparametric 2DW

The boundary between parametric 2DW and nonparametric 2DW is not strict, and in fact, their hybrids have been proposed. A popular way to realize the hybrids is to employ piecewise linearity, or local linearity. DP6 and DP7 (Ronee, Uchida & Sakoe, 2001; Uchida & Sakoe, 2000b) of Figure 9 is a piecewise linear version of DP5. In DP6, the mapping of each column (i.e., the mapping of N pixels) is represented as line segments whose control points are fewer than N. DP7 employs more drastic linearization that the column is mapped as a line. The merit of the DP6 and DP7 is that they can compensate 2D deformations (unlike DP1-DP4) with polynomial-order computations. (For DP6 and DP7, the optimization by DP requires $O(N^6)$ and $O(N^4)$ computations.)

Figure 9. Piecewise linear 2DW



Local affine transformation (LAT) proposed by Wakahara (1994) is also a hybrid based on local linearization. In LAT, 2DW is described by a set of many locally effective affine transformations. Thus, LAT is a parametric 2DW in a microscopic sense and simultaneously a nonparametric 2DW in a macroscopic sense.

Uchida and Sakoe (2003b) also present a hybrid 2DW, different from local/piecewise linearization. This 2DW is a parametric and orthogonal 2DW where the principal components of within-category deformations, called eigen-deformations, are used as orthogonal functions. The eigen-deformations themselves, however, are estimated from the results of some nonparametric 2DW. Thus, this technique can be considered a parametric 2DW on its optimization and a nonparametric 2DW on the ability of compensating deformations.

The deformations of handwritten characters can be decomposed into global deformations and local deformations. Scaling, rotation, translation, and projective transformation of an entire character image are examples of the global deformations. Independent and partial changes of stroke direction, curvature, and length are examples of local deformations. EM should compensate both deformations. Since the parametric 2DW and the nonparametric 2DW are suitable for compensating the global deformations and the local deformations, respectively, their cooperative combination will be promising.

RELATED TOPICS

Comparison with Shape Normalization

Shape normalization (Lee & Park, 1994; Liu et al., 2004) is another strategy to provide a deformation-invariant distance. For example, linear scaling of the bounding box of a character is the simplest normalization technique and enough to realize scale-invariant recognition. Line density equalization (Yamada, Yamamoto & Saito, 1990) is a nonlinear shape normalization technique and can adjust nonlinear deformations.

Figure 10 illustrates the difference between shape normalization and EM. As shown in Figure 10(a), EM shifts **B** to a pattern close to **A**. In contrast, as shown in Figure 10(b), shape normalization shifts **A** to a pattern having some ideal property and shifts **B** independently to another pattern having the same property. In general, EM has a greater ability to compensate deformations. This fact also indicates that EM has more risks of providing underestimated distance between two character patterns of different categories; namely, EM has more risks of overfitting.

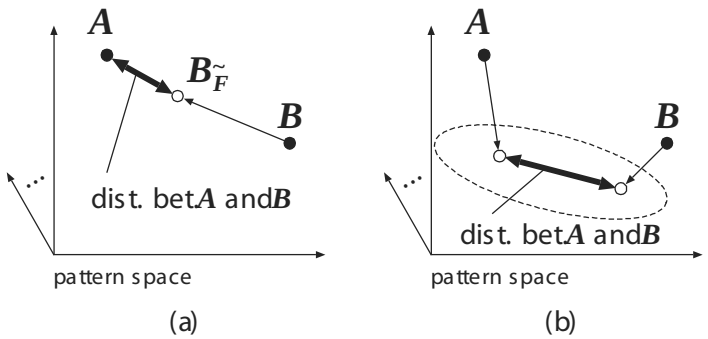
EM and shape normalization can be utilized in a collaborative manner like the collaboration of parametric 2DW and nonparametric 2DW discussed previously. For example, shape normalization is applied first to remove global and simple deformations, and then EM is applied to remove local and complicated deformations. This collaboration may suppress the overfitting by EM. Tsukumo's (1992) EM technique based on DP2 is another good example where blurring normalization is employed to complement the lesser flexibility of DP2.

Reference Patterns for EM

EM-based recognition can be considered multiple reference-based recognition since EM generates a reference pattern $\mathbf{B}_F$ adaptively to the input pattern **A**. Thus, the seed of the adaptive generation (i.e., the original reference pattern **B**) should be carefully designed. Most of EM-based recognizers, however, do not pay much attention to this point; several patterns manually designed/selected are often used as **B**, or all of training patterns are directly used as a set of **B** (Jain & Zongker, 1997; Simard et al., 1993).

Like other recognizers, clustering will be promising to set reference patterns in a sophisticated manner. Actually, any clustering technique (e.g., k-means, ISODATA, LVQ, and GLVQ) can be used for EM. Matsumoto, et al. (2004) have pointed out that the objective function of clustering should be designed using the EM distance instead of the conventional Euclidean distance. This is because the reference patterns optimized under a Euclidean distance-based clustering are

Figure 10. Distance between **A** *and* **B** *given by EM (a) and normalization (b)*



not optimal for EM distance-based discrimination (see Figure 3). Accordingly, Matsumoto, et al. (2004) have proposed a k-means algorithm based on the EM distance. Another trial can be found in Hastie, Simard, and Säckinger (1995), where clustering based on the tangent distance has been proposed.

Pixel Feature

In general, the less ambiguous a pixel feature vector becomes, the more accurate the optimized 2DW becomes. In the ultimate case that only a pair of pixels corresponding truly has the same pixel feature vector, local perturbation (the most naïve optimization strategy) is sufficient to obtain optimal 2DW. Conversely, the more ambiguous a pixel feature vector becomes, the less accurate the optimized 2DW becomes. For example, if we try to find 2DW with binary pixel feature (black/white), it is very hard to obtain a reasonable 2DW.

Local context (Keysers et al., 2004) and directional feature (Mizukami, 1998; Uchida & Sakoe, 1999) are simple and reasonable choices as a less ambiguous pixel feature. Those shape-sensitive features, however, face a problem of inconsistency. As shown in Figure 11, those feature vectors are often not the same at corresponding pixel pairs. Local contexts, each of which is a subimage, are different at the pixels corresponding truly.

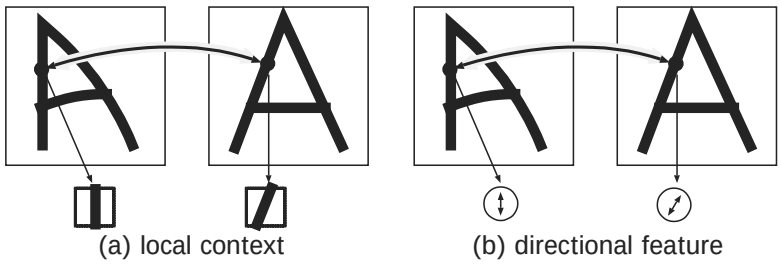
There are two possible remedies for this problem of inconsistency. One is the adaptation of the feature vectors according to 2DW. For example, under the 2DW showing the rotation of angle θ, the directional feature becomes consistent by shifting the original directional feature by θ. The other is the use of invariant features, such as moments.

Category-Dependent Deformation Tendency

Each category has its own deformation tendency. Figure 12 is a simple example that proves its existence. In category "M," two parallel vertical strokes are often slanted to be closer (Figure 12(a)). The same deformation, however, is rarely observed in "H" (Figure 12(b)). Consequently, if an EM technique based on the assumption can compensate the deformations of "M," it may suffer from the overfitting of "H" to "A."

The necessity of the category-dependent EM techniques indicated by this simple example will be confirmed by the result of a character recognition experiment. Six hundred handwritten character samples of ETL6 (http://www.is.aist.go.jp/etlcdb/) were prepared for each of 26 categories of English alphabets. Four-dimensional directional features (0, 45, 90, and -45 degrees) were extracted at each pixel. Then they are combined with a gray-level feature to be a five-dimensional pixel feature vector. Each sample was linearly scaled to 20 x 20. The first 100 samples of each category were simply averaged to create one reference pattern **B**. The remaining 500 samples were used as test samples **A**. The EM techniques based on DP1 ($\Delta xi = 2$),

Figure 11. Inconsistency in pixel features



DP3 ($\Delta xi = \Delta yj = 2$), DP5 ($\Delta xi = \Delta yj = 2$, $\Delta xj = \Delta yi = 1$), and DP6 were chosen, and DP was used as their optimization strategy.

The necessity of the category-dependent EM techniques is proved by the category-wise result of this experiment shown in Figure 13; that is, the most appropriate flexibility is different in each category. The most flexible 2DW (DP5) could not provide the best result in many categories; on the contrary, the most rigid 2DW (DP1) could provide the best result for several categories. That is, each category has its own range of deformations, and excessive/insufficient flexibility often degrades the recognition performance. In this sense, category-dependent EM techniques, such as HMM and ASM-based EM, are more promising than category-independent ones.

EM TECHNIQUES BASED ON 1D-2D MAPPING

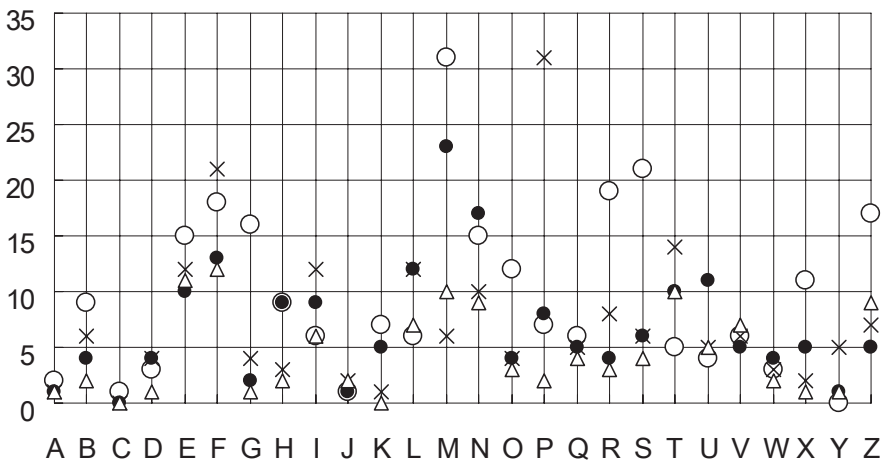
Since any character is a "linear" pattern, it is also natural to use a one-dimensional stroke model instead of a two-dimensional image model. When we use these 1D-stroke models, our task becomes the fitting of the 1D model onto a 2D input pattern. This task is an optimization problem of a 1D-2D mapping function between the 1D model and the 2D input and therefore also a kind of EM.

The 1D-2D EM techniques can be classified by their stroke models. One stroke model will be defined as a sequence of x-y coordinates, and another will be defined as a sequence of line segments. More generally, the model will be defined as a sequence of states, each of which represents a local shape of a stroke.

Rubber-string (RS) matching (Sakoe, 1974) is one of the most classical 1D-2D EM techniques.

Figure 12. Example of category-dependent deformation tendency (© copyright 2005IEICE)

Figure 13. The number of misrecognized samples by DP1 (○), DP3 (●), DP5 (×), and DP6 (Δ)



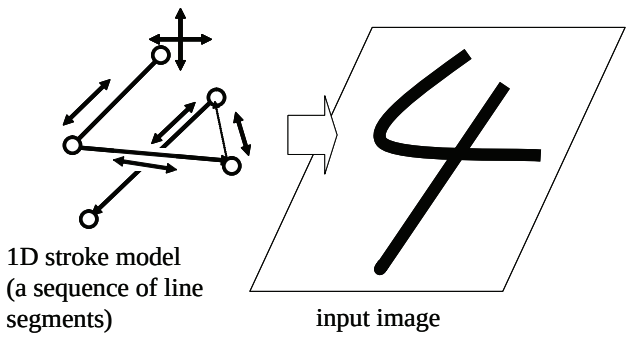
As shown in Figure 14, its stroke model is a sequence of line segments. The direction of the line segment is fixed, but the length is flexible. Thus, the optimal fitting problem is reduced to the optimization problem of the lengths of the line segments (and the initial point). In Sakoe (1974), a DP-based algorithm has been proposed for solving the optimization problem. In Sakoe, Ali, and Katayama (1994), RS matching has been extended to increase its ability to represent fine deformations within line segments.

The stroke model by Revow, Williams, and Hinton (1996) is a more elaborated one. It is a stochastic model and represents stroke shape variations by a sequence of Gaussian ink generators. The ink generator outputs ink dots according to a probabilistic distribution. The similarity between the model and the input image is evaluated as a likelihood of the input image by the model and calculated through an expectation-maximization estimation of the location and the variance of the ink generators. This approach has been rearranged in a Bayesian framework in Cheung, Yeung, and Chin (1998). Kato, Omachi, and Aso (2000) have introduced a multiresolution framework for dealing with characters with heavier deformations.

A weak point of the 1D-2D EM is its embedment problem; the stroke model of the digit "1" may be fitted (or embedded) perfectly to character images with a vertical line (e.g., "4," "5," "7," and "9"). Once the stroke model is embedded into an input character image, the fitting is evaluated only around the stroke model, and the input character image may be considered a similar image to the model. Hastie and Tibshirani (1994) called this situation *uncovered*. One possible remedy is a posteriori evaluation of a residual part on the input image. For example, we will have a "c"-shaped residual part on fitting the model "1" to an input image "9," and therefore, we can avoid the misrecognition by penalizing the residual part. Another remedy is the regulation of global deformation of the stroke model. Revow, et al. (1996) employ affine transformation to fit their model roughly to the input image.

We can find 1D-2D EM techniques in other formulations, such as contour matching techniques by Yamada (1984) and Yamamoto and Rosenfeld (1982) and thinned pattern matching proposed by Fujimoto, Kadota, Hayashi, Yamamoto, Yajima, and Yasuda (1976). The ASM-based method (Shi et al., 2003) is also a 1D-2D EM technique, where a chamfer matching distance is employed to evaluate the fitting between a stroke model and a binary character image. Kobayashi, Nakamura, Muramatsu, Sugiyama, and Abe (2001) have proposed a SNAKES-like deformable model where optimal fitting, which gives minimum energy, is searched for in an iterative manner.

Figure 14. 1D-2D EM technique with a 1D stroke model



FUTURE TASKS

We still have many tasks to utilize EM techniques in practical problems such as commercial OCR for handwritten characters. The following is a list of those tasks.

Reduction of Computational Complexity

Generally speaking, EM requires a fair amount of computations. This fact is crucial, especially for a character set comprised of many categories such as Chinese characters. Thus, acceleration of the EM algorithm is necessary. Coarse classification based on a rigid distance is a possible remedy.

Design of 2DW by Category-Dependent Characteristics

As proved by the experiment, category-dependent EM is very promising. Statistic and/or stochastic frameworks will help realize it. Discriminative learning of 2DW will be useful to suppress the misrecognitions due to overfitting. Kernel machines are also promising partners.

Multistep Deformation Compensation

As discussed, we can apply parametric 2DW to compensate global deformations and then nonparametric 2DW to compensate local and complicated deformations. This two-step framework may reduce the degradation by overfitting. Shape normalization also compensates for global deformations, and therefore, it is useful to narrow the range of 2DW.

Feature Extraction

Less ambiguous pixel features are required for accurate 2DW. If such a desirable pixel feature is available, we will be able to use unconstrained 2DW instead of costly constrained 2DW. Adaptation of pixel features for consistent correspondence is still an open problem.

EM for Handwritten Word Recognition

EM techniques employed in word recognition are rather simple like DP1. In handwritten words, not only deformations within individual characters but also deformations between adjacent characters are observed. The compensation of such complex deformations is challenging.

Utilization of Optimized 2DW

It is important to note that the optimized 2DW represents the deformation of **B** relative to **A**. This fact implies that EM is one of structural analysis techniques for image patterns. The utilization of the optimized 2DW is very promising to extract various properties of handwritten characters.

EM for Camera-Based Character Recognition

Camera-based character recognition is a recent research trend with many open problems (Liang et al., 2005). Due to an oblique camera angle, each character undergoes perspective transformation. Due to a nonflat document surface, each character undergoes nonlinear geometric transformation as well as photometric deformation. Past trials to tackle those problems are called *dewarping* and treated as preprocessing techniques. Although they are superficially different from EM techniques reviewed in this chapter, the various techniques to optimize 2DW will be applicable to the dewarping problem. In fact, the document dewarping method by Ezaki, Uchida, and Sakoe (2005) relies on a DP-based 2DW optimization technique.

EM for Kernel Machines

Recently, EM techniques have been linked to kernel machines, since the distance by EM can be considered a nonlinear kernel between two patterns. Especially the DP-based EM techniques are strongly related to string kernels based on the edit distance, or Levenshtein distance, which is often provided by DP. The use of EM will extend the horizon of kernel machines so they can deal with a set of patterns with different dimensionalities.

CONCLUSION

This chapter reviewed elastic matching (EM) techniques. Since the distance provided by EM is invariant to a certain range of geometric deformations, EM has been employed in handwritten character recognition tasks. EM is defined as the optimization problem of two-dimensional warping (2DW), which specifies 2D-2D mapping between two image patterns. In a discrete case, 2DW specifies pixel-to-pixel correspondence between them.

This chapter also showed that EM techniques can classify by two factors: the formulation of 2DW and the optimization strategy of 2DW. Those factors actually are strongly related to each other; that is, each kind of 2DW has its appropriate optimization strategy.

As noted in the last section, there remain many open problems and future work in the application of EM to handwritten character/word recognition. Further researches tackling these problems will be very meaningful from not only theoretical but also practical viewpoints.

ACKNOWLEDGMENT

The author greatly thanks Prof. Hiroaki Sakoe for his thoughtful, sincere, and valuable advice throughout my research career.

REFERENCES

Agazzi, O., Kuo, S., Levin, E., & Pieraccini, R. (1993). Connected and degraded text recognition using planar hidden Markov models. Proc. ICASSP, V113–116.

Bajscy, R., & Broit, C. (1982). Matching of deformed images. *Proc. ICPR*, 351–353.

Bippus, R., & Märgner, V. (1999). Script recognition using inhomogeneous P2DHMM and hierarchical search space reduction. *Proc. ICDAR*, 773–776.

Burr, D.J. (1981). A dynamic model for image registration. *Computer Graphics, and Image Proc., 15*, 102–112.

Chen, M.C., & Willson Jr., A.N. (2000). Motion-vector optimization of control grid interpolation and overlapped block motion compensation using iterated dynamic programming. *IEEE Trans. Image Proc., 9*(7), 1145–1157.

Cheung, K.-W., Yeung, D.-T., & Chin, R.T. (1998). A Bayesian framework deformable pattern recognition with application to handwritten character recognition. *IEEE Trans Pattern Anal Mach Intell, 20*(12), 1382–1388.

Cho, W., Lee, S.-W., & Kim, J.H. (1995). Modeling and recognition of cursive words with hidden Markov models. *Pattern Recognit, 28*(12), 1941–1953.

Christensen, G.E., Rabbitt, R.D., & Miller, M.I. (1996). Deformable templates using large deformation kinematics. *IEEE Trans Image Processing, 5*(10), 1435–1447.

Cootes, T.F., Taylor, C.J., Cooper, D.H., & Graham, J. (1995). Active shape models—Their training and application. *Comput Vis Image Und, 61*(1), 38–59.

Ezaki, H., Uchida, S., & Sakoe, H. (2005). Dewarping of document image by global optimization. *Proc. ICDAR*, 1, 302–306.

Fujimoto, Y., Kadota, S., Hayashi, S., Yamamoto, M., Yajima, S., & Yasuda, M. (1976). Recognition of handprinted characters by nonlinear elastic matching. *Proc. ICPR*, 113–118.

Glasbey, C.A., & Mardia, K.V. (1998). A review of image-warping methods. *J Applied Statistics, 25*(2), 155–171.

Ha, T.M., & Bunke, H. (1997). Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans Pattern Anal Mach Intell, 19*(5), 535–539.

Halloui, K., Likforman-Sulem, L., & Sigelle, M. (2002). A comparative study between decision fusion and data fusion in Markovian printed character recognition. *Proc. ICPR*, 3 of 4, 147–150.

Hastie, T., Simard, P.Y., & Säckinger, E. (1995). Learning prototype models for tangent distance. *Advances in Neural Information Processing Systems, 7*, 999–1006.

Hastie, T.J., & Tibshirani, R. (1994). *Handwritten digit recognition via deformable prototypes.* AT&T Bell Laboratories Technical Report.

Hattori, T., Watanabe, Y., Sanada, H., & Tezuka, Y. (1983). Absorption of local variations in handwritten character by an elastic transformation using vector field [in Japanese]. *IEICE Trans, J66-D*(6), 645–652.

Isomichi, Y., & Ogawa, T. (1975). Pattern matching by using dynamic programming [in Japanese]. *J Inf Process Soc Japan, 16*(1), 15–22.

Izui, Y., Harashima, H., & Miyagawa, H. (1985). Handprinted Chinese characters recognition by hierarchical modification of dictionary [in Japanese]. *IEICE Trans, J68-D*(3), 361–368.

Jain, A.K., Zhong, Y., & Dubuisson-Jolly, M.-P. (1998). Deformable template models: A review. *Signal Processing, 71*(2), 109–129.

Jain, A.K., & Zongker, D. (1997). Representation and recognition of handwritten digits using deformable templates. *IEEE Trans Pattern Anal Mach Intell, 19*(12), 1386–1391.

Kato, T., Omachi, S., & Aso, H. (2000). Precise hand-printed character recognition using elastic models vianonlinear transformation. *Proc. ICPR*, 2, 364–367.

Keysers, D., Dahmen, J., Theiner, T., & Ney, H. (2000). Experiments with an extended tangent distance. *Proc. ICPR*, 2 of 4, 38–42.

Keysers, D., Gollan, C., & Ney, H. (2004). Local context in non-linear deformation models for handwritten character recognition. *Proc. ICPR*, 4 of 4, 511–514.

Keysers, D., Macherey, W., Ney, H., & Dahmen, J. (2004). Adaptation in statistical pattern recognition using tangent vectors. *IEEE Trans Pattern Anal Mach Intell, 26*(2), 269–274.

Keysers, D., & Unger, W. (2003). Elastic image matching is NP-complete. *Pattern Recog Lett, 24*(1)-3, 445–453.

Kimura, F., Yoshimura, M., Miyake, Y., & Ichikawa, M. (1970). Unconstrainedly handprinted "KATAKANA"' character recognition by a stroke structure analysis method [in Japanese]. *Trans IEICE, J62-D*(1), 16–23.

Kobayashi, T., Nakamura, K., Muramatsu, H., Sugiyama, T., & Abe, K. (2001). Handwritten numeral recognition using flexible matching based on learning of stroke statistics. Proc. ICDAR, 612–616.

Kuo, S., & Agazzi, O.E. (1994). Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *IEEE Trans Pattern Anal Mach Intell, 16*(8), 842–848.

Lee, S.-W., & Park, J.-S. (1994). Nonlinear shape normalization methods for the recognition of large-set handwritten characters. *Pattern Recognit, 27*(7), 895–902.

Lester, H., & Arridge, S.R. (1999). A survey of hierarchical non-linear medical image registration. *Pattern Recognit, 32*(1), 129–149.

Levin, E., & Pieraccini, R. (1992). Dynamic planar warping for optical character recognition. Proc. ICASSP, III 149–152.

Liang, J., Doermann, D., & Li, H. (2005). Camera-based analysis of text and documents: A survey. *Int J Doc Anal Recog, 7*, 84–104.

Liolios, N., Kavallieratou, E., Fakotakis, N., & Kokkinakis, G. (2002). A new shape transformation approach to handwritten character recognition. *Proc. ICPR*, 1 of 4, 584–587.

Liu, C.L., Nakashima, K., Sako, H., & Fujisawa, H. (2004). Handwritten digit recognition: Investigation of normalization and feature extraction techniques. *Pattern Recognit, 37*(2), 265–279.

Makhoul, J., Schwaltz, R., Lapre, C., & Bazzi, I. (1998). A script-independent methodology for optical character recognition. *Pattern Recognit, 31*(9), 1285–1294.

Matsumoto, N., Uchida, S., & Sakoe, H. (2004). Prototype setting for elastic matching-based image pattern recognition. *Proc. ICPR*, 1 of 4, 224–227.

Meguro, S., & Umeda, M. (1978). *An extraction of shape derivations in handwritten characters by hierarchical pattern matching* [in Japanese]. Tech. Rep. of IEICE Japan, PRL77-70.

Mizukami, Y. (1998). A handwritten Chinese character recognition system using hierarchical displacement extraction based on directional features. *Pattern Recog Lett, 19*(7), 595–604.

Mohamed, M., & Gader, P. (1996). Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IE*EE Trans Pattern Anal Mach Intell, 18*(5), 548–554.

Moore, R.K. (1979). A dynamic programming algorithm for the distance between two finite areas. *IEEE Trans Pattern Anal Mach Intell, PAMI-1*(1), 86–88.

Mori, S., Suen, C.Y., & Yamamoto, K. (1992). Historical review of OCR research and development. *Proc. IEEE, 80*(7), 1029–1058.

Mori, S., Yamamoto, K., & Yasuda, M. (1984). Research on machine recognition of handprinted characters. *IEEE Trans Pattern Anal Mach Intell, PAMI-6*(4), 386–405.

Nakagawa, M., Yanagida, T., & Nagasaki, T. (1999). An off-line character recognition method employing model-dependent pattern normalization by an elastic membrane model. *Proc. ICDAR*, 495–498.

Nakano, Y., Nakata, K., Uchikura, Y., & Nakajima, A. (1973). Improvement of Chinese character recognition using projection profiles. *Proceedings of the International Joint Conference on Pattern Recognition*, 172–178.

Nakata, K., Nakano, Y., & Uchikura, Y. (1972). Recognition of Chinese characters. *Proceedings of the Conference on Machine Perception of Patterns and Pictures*, 45–52.

Nishimura, H., Tsutsumi, M., Maruyama, M., Miyao, M., & Nakano, Y., (2001). Off-line handwritten character recognition using integrated 1D HMMs based on feature extraction filters. *Proc. ICDAR*, 417–421.

Park, H.-S. & Lee, S.W. (1998). A truly 2-D hidden Markov model for off-line handwritten character recognition. *Pattern Recognit, 31*(12), 1849–1864.

Quénot, G.M. (1992). The orthogonal algorithm for optical flow detection using dynamic programming. *Proc. ICASSP*, III-249–252.

Redert, A., Hendriks, E., & Biemond, J. (1999). Correspondence estimation in image pairs. *IEEE SP Mag, 16*(3), 29–46.

Revow, M., Williams, C.K.I., & Hinton, G.E. (1996). Using generative models for handwritten digit recognition. *IEEE Trans Pattern Anal Mach Intell, 18*(6), 592–606.

Ronee, M.A., Uchida, S., & Sakoe, H. (2001). Handwritten character recognition using piecewise linear two-dimensional warping. *Proc. ICDAR*, 39–43.

Saito, T., Yamada, H., & Yamamoto, K. (1982). An analysis of handprinted Chinese characters by directional pattern matching approach [in Japanese]. *IEICE Trans, J65-D*(5), 550–557.

Sakaue, K., Amano, A., & Yokoya, N. (1999). Optimization approaches in computer vision and image processing. *IEICE Trans Inf & Syst, E82-D*(3), 534–547.

Sakoe, H. (1974). *Handwritten character recognition by rubber string matching method* [in Japanese]. Tech. Rep. of IEICE Japan, PRL74-20.

Sakoe, H., Ali, M.M., & Katayama, Y. (1994). One dimensional-two dimensional dynamic programming algorithm for character recognition. *IEICE Trans Information & Systems, E77-D*(9), 1047–1054.

Shi, D., Gunn, S.R., & Damper, R.I. (2003). Handwritten Chinese radical recognition using nonlinear active shape models. *IEEE Trans Pattern Anal Mach Intell, 25*(2), 277–280.

Shiku, O., Nakamura, A., Kuroda, H., & Miyahara, S. (2000). A method for handwritten Japanese word recognition based on holistic strategy [in Japanese]. *Trans Inf Process Soc Japan, 41*(4), 1086–1095.

Simard, P., Le Cun, Y., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems, 5*, 50–58.

Sugimura, M., Iiguni, Y., & Adachi, N. (1997). A 2-dimensional dynamic programming for image matching with Zernike moments [in Japanese]. *Trans IEICE, J80-D-II*(1), 101–108.

Tanaka, E. (1985). A two dimensional context-dependent similarity measure. *Trans IECE Japan, E-68*(10), 667–673.

Trier, Ø.D., Jain, A.K., & Taxt, T. (1996). Feature extraction methods for character recognition—A survey. *Pattern Recognit, 29*(4), 641–662.

Tsukumo, J. (1992). Handprinted Kanji character recognition based on flexible template matching. *Proc. ICPR*, 483–486.

Uchida, S., & Sakoe, H. (1998). A monotonic and continuous two-dimensional warping based on dynamic programming. *Proc. ICPR*, 1 of 2, 521–524.

Uchida, S., & Sakoe, H. (1999). Handwritten character recognition using monotonic and continuous two-dimensional warping. *Proc. ICDAR*, 499–502.

Uchida, S., & Sakoe, H. (2000a). An approximation algorithm for two-dimensional warping. *IEICE Trans Info & Syst, E83-D*(1), 109–111.

Uchida, S., & Sakoe, H. (2000b). Piecewise linear two-dimensional warping. *Proc ICPR, 3*, 538–541.

Uchida, S., & Sakoe, H. (2003a). Eigen-deformations for elastic matching based handwritten character recognition. *Pattern Recognit, 36*(9), 2031–2040.

Uchida, S., & Sakoe, H. (2003b). Handwritten character recognition using elastic matching based on a class-dependent deformation model. *Proc. ICDAR, 1*, 163–167.

Uchida, S., & Sakoe, H. (2005). A survey of elastic matching techniques for handwritten character recognition. *IEICE Transactions on Information & Systems, E88-D*(8), 1781–1790.

Wakahara, T. (1994). Shape matching using LAT and its application to handwritten numeral recognition. *IEEE Trans Pattern Anal Mach Intell, 16*(6), 618–629.

Wakahara, T., Kimura, Y., & Tomono, A. (2001). Affine-invariant recognition of gray-scale characters using global affine transformation correlation. *IEEE Trans Pattern Anal Mach Intell, 23*(4), 384–395.

Wakahara, T., & Odaka, K. (1998). Adaptive normalization of handwritten characters using global/local affine transformation. *IEEE Trans Pattern Anal Mach Intell, 20*(12), 1332–1341.

Wang, W., Brakensiek, A., Kosmala, A., & Rigoll, G. (2001). Multi-branch and two-pass HMM modeling approaches for off-line cursive handwriting recognition. *Proc. ICDAR*, 231–235.

Webster, R., & Nakagawa, M. (1997). An on-line/off-line compatible character recognition method based on a dynamic model. *IEICE Trans Inf & Syst, E80-D*(6), 672–683.

Wu, C.-M., Liu, P., & Chang, W.-C. (1995). Unconstrained-endpoint dynamic space-warping algorithm with experiments in binary English character images. *Int J Electronics, 78*(1), 55–66.

Yamada, H. (1984). Contour DP matching method and its application to handprinted Chinese character recognition. *Proc. ICPR*, 389–392, 1 of 2.

Yamada, H., Saito, T., & Mori, S. (1981). An improvement of correlation method—Locally maximized correlation [in Japanese]. *Trans IEICE, J64-D*(10), 970–976.

Yamada, H., Yamamoto, K., & Saito, T. (1990). A nonlinear normalization method for handprinted Kanji character recognition—Line density equalization. *Pattern Recognit, 23*(9), 1023–1029.

Yamamoto, K., & Rosenfeld, A. (1982). Recognition of hand-printed KANJI characters by a relaxation method. *Proc. ICPR*, 1 of 2, 395–398.

Yasuda, M., Yamamoto, K., & Yamada, H. (1997). Effect of the perturbed correlation method for optical character recognition. *Pattern Recognit, 30*(8), 1315–1320.

Yen, C., Kuo, S.S., & Lee, C.-H. (1999). Minimum error rate training for PHMM-based text recognition. *IEEE Trans Image Proc, 8*(8), 1120–1124.

Zitová, B., & Flusser, J. (2003). Image registration methods: A survey. *Image and Vision Computing, 21*, 977–1000.

Chapter III
State of the Art in Off-Line Signature Verification

Luana Batista
École de technologie supérieure, Canada

Dominique Rivard
École de technologie supérieure, Canada

Robert Sabourin
École de technologie supérieure, Canada

Eric Granger
École de technologie supérieure, Canada

Patrick Maupin
Defence Research and Development Canada (DRDC), Canada

ABSTRACT

Automatic signature verification is a biometric method that can be applied in all situations where handwritten signatures are used, such as cashing a check, signing a credit card, authenticating a document, and others. Over the last two decades, several innovative approaches for off-line signature verification have been introduced in literature. Therefore, this chapter presents a survey of the most important techniques used for feature extraction and verification in this field. The chapter also presents strategies used to face the problem of a limited amount of data, as well as important challenges and research directions.

INTRODUCTION

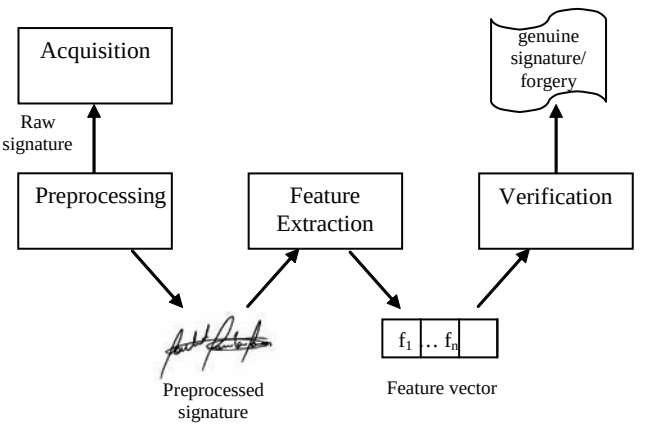
Biometrics refers to automated methods used to verify or recognize the identity of a person. In contrast to the conventional identification systems whose features such as ID cards or passwords can be forgotten, lost, or stolen, biometric systems are based on physiological or behavioral features that are difficult for another individual to reproduce, thereby reducing the possibility of forgery (Kung, Mak & Lin, 2004). Fingerprints, voice, iris, retina, hand, face, handwriting, keystroke, and finger shape are examples of popular features used in biometrics. The use of other biometric measures such as gait, ear shape, head resonance, optical skin reflectance, and body odor is still in an initial research phase (Wayman, Jain, Maltoni & Maio, 2005).

The handwritten signature has always been one of the most simple and accepted ways to authenticate an official document. It is easy to obtain, results from a spontaneous gesture, and is unique to each individual (Abdelghani & Amara, 2006). Automatic signature verification, therefore, can be applied in all situations where handwritten signatures are currently used, such as cashing a check, signing a credit card transaction, and authenticating a document (Griess & Jain, 2002).

The goal of a signature verification system is to verify the identity of an individual based on an analysis of his or her signature through a process that discriminates a genuine signature from a forgery (Plamondon, 1994). Figure 1 shows an example of a signature verification system. The process follows the classical pattern recognition model steps; that is, data acquisition, preprocessing, feature extraction, classification (generally called "verification" in the signature verification field), and performance evaluation.

Depending on the data acquisition mechanism, the process of signature verification can be classified *online* and *off-line*. In the online (or dynamic) approach, specialized hardware such as a digitizing tablet or a pressure-sensitive pen is used in order to capture the pen movements over the paper at the same time of the writing. In this case, a signature can be viewed as a space-time variant curve $O(t) = [\sum(t), \Theta(t), \Gamma(t)]$, where $\sum(t)$ is the curvilinear displacement, $\Theta(t)$ is the angular displacement, and $\Gamma(t)$ is the torsion of its trajectory (Plamondon & Lorette, 1989). On the other hand, in the off-line (or static) approach, the signature is available on a sheet of paper, which is later scanned in order to obtain a digital representation composed of M x N pixels. Hence, the signature image is considered as a discrete 2D function *f(x,y)*, where $x = 0, 1, 2, \ldots, M$ and $y = 0, 1, 2, \ldots,$

Figure 1. Block diagram of a generic signature verification system



N denote the spatial coordinates. The value of *f* in any *(x,y)* corresponds to the grey level (generally a value from 0 to 255) in that point (Gonzalez & Woods, 2002).

Over the last two decades, and with renewed interest in biometrics caused by the tragic events of 9/11, several innovative approaches for off-line signature verification have been introduced in literature. Therefore, this chapter presents a survey of off-line signature verification techniques, focusing on the feature extraction and verification strategies. The goal is to present the most important advances as well as the current challenges in this field. Of particular interest are the techniques that allow for designing a signature verification system based on a limited amount of data.

The chapter is organized as follows. In the next section, the types of signatures and forgeries are defined. Next, a literature review in feature extraction techniques and verification strategies proposed in this field is presented. Some strategies used to face the problem of a limited amount of data are then discussed. Finally, the conclusions of the chapter are presented.

SIGNATURE AND FORGERY TYPES

Signature verification is directly related to the alphabet (e.g., Roman, Chinese, Arabic, etc.) and the form of writing of each region (Justino, 2001). The occidental signatures can be classified in two main styles: *cursive* and *graphical* (Figure 2). With *cursive* signatures, the author writes his or her name in a legible way, while the graphical signatures contain complex patterns that are very difficult to interpret as a set of characters.

According to Coetzer, Herbst, and du Preez (2004), forged signatures can be classified in three basic types:

1. *Random forgery.* The forger has no access to the genuine signature (not even the author's name) and reproduces a random one. A random forgery may also include the forger's own signature.
2. *Simple forgery.* The forger knows the author's name but has no access to a sample of the signature. Thus, the forger reproduces the signature in his or her own style.
3. *Skilled forgery.* The forger has access to one or more samples of the genuine signature and is able to reproduce it. Skilled forgeries even can be subdivided according to the level of the forger's skill. Figure 3 presents examples of the mentioned types of forgeries.

Generally, only random forgeries are used to train the classifier(s) of a signature verification

Figure 2. Examples of cursive (a) (b) and graphical (c) signatures

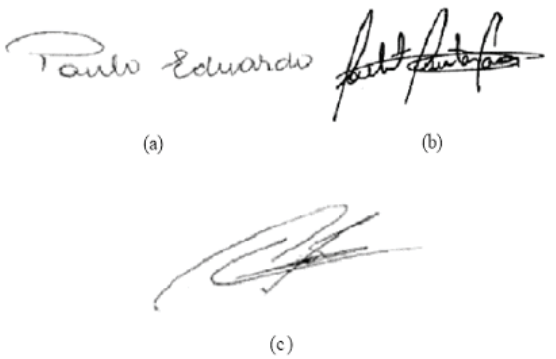


Figure 3. Examples of (a) genuine signature, (b) random forgery, (c) simple forgery, and (d) skilled forgery

system. The reason is that, in practice, it is rarely possible to obtain samples of forgeries; and, when dealing with banking applications, for example, it becomes impracticable (Murshed, Bortolozzi & Sabourin, 1995). On the other hand, all types of forgeries are used to evaluate the system's performance.

FEATURE EXTRACTION TECHNIQUES

Feature extraction is essential to the success of a signature verification system. In an off-line environment, signatures are acquired from a medium, usually paper, and preprocessed before the feature extraction begins. Off-line feature extraction is a fundamental problem because of a handwritten signature's variability and the lack of dynamic information about the signing process. An ideal feature extraction technique extracts a minimal feature set that maximizes interpersonal distance between signature examples of various persons while minimizing intrapersonal distance for those belonging to the same person.

There are two classes of features used in off-line signature verification: (1) *static*, related to the signature shape, and (2) *pseudo-dynamic*, related to the dynamics of the writing. These features can be extracted locally if the signature is viewed as a set of segmented regions, or globally if the signature is viewed as a whole. It is important to note that techniques used to extract global features also can be applied to specific regions of the signature in order to produce local features. In the same way, a local technique can be applied to the whole image to produce global features. Figure 4 presents a way to visualize the categories of features used in signature verification.

Moreover, local features can be classified as *contextual* and *noncontextual*. If the signature segmentation is performed in order to interpret the text (e.g., bars of "t" and dots of "i"), the analysis is considered contextual (Chuang, 1977). This type of analysis is not popular for two reasons: (1) it requires a complex segmentation process, and (2) it is not suitable to deal with graphical signatures. On the other hand, if the signature is viewed as a drawing composed by line segments (as it occurs in the majority of the works), the analysis is considered noncontextual.

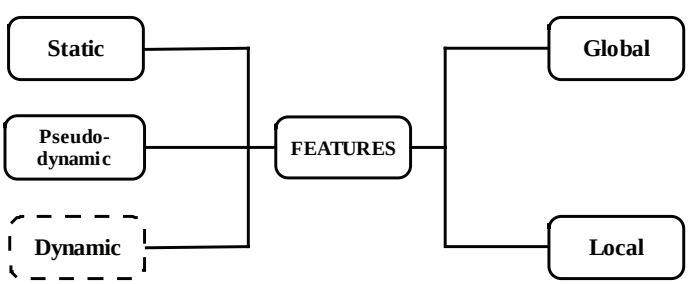
Before describing some of these features, the many ways to represent a signature image are discussed.

Representations of a Signature

Some extraction techniques transform the signature image into another representation before extracting the features. The literature is quite extensive in signature representations.

Box and convex hull representations have been used to represent signatures (Frias-Martinez, Sanchez & Velez, 2006). The box representation

Figure 4. Features used in signature verification; the dynamic features are used only in online approaches



is composed of the smallest rectangle fitting the signature. Its perimeter, area, and perimeter/area ratio can be used as features. The convex hull representation is composed of the smallest convex hull fitting the signature. Its area, roundness, compactness, and length and orientation of its maximum axis can be used as features.

The skeleton of the signature, its outline, directional frontiers, and ink distributions also have been used as signature representations (Huang & Yan, 1997). The skeleton (or core) representation is the pixel wide strokes resulting from the application of a thinning algorithm to a signature image. The skeleton can be used to identify the signature edge points (1-neighbor pixels) that mark the beginning and ending of strokes (Ozgunduz, Senturk & Karsligil, 2005). Further, pseudo-Zernike moments also have been extracted from this kind of representation (Wen-Ming, Shao-Fa & Xian-Gui, 2004).

The outline representation is composed of every black pixel adjacent to at least one white pixel. Directional frontiers (also called shadow images) are obtained when keeping only the black pixels touching a white pixel in a given direction (there are 8 possible directions). To perform ink distribution representations, a virtual grid is posed on the signature image. The cells containing more than 50% of black pixels are completely filled while the others are emptied. Depending on the grid scale, the ink distributions can be coarser or more detailed. The number of filled cells can also be used as a global feature.

Upper and lower envelopes (or profiles) are also found in the literature. The upper envelope is obtained by selecting column-wise the upper pixels of a signature image, while the lower envelope is achieved by selecting the lower pixels. As global features, the number of turns and gaps in these representations have been extracted (Ramesh & Murty, 1999).

Mathematic transforms have been used to represent signature images. Nemcek and Lin (1974) chose the fast Hadamard transform in their feature extraction process as a tradeoff between computational complexity and representation accuracy, when compared to other transforms. Discrete Radon transform is used to extract an observation sequence of the signature, which is used as a feature set (Coetzer et al., 2004).

Finally, signature images also can undergo a series of transformations before feature extraction. For example, Tang, Tao, and Lam (2002) used a central projection to reduce the signature image to a 1-D signal that is in turn transformed by a wavelet before fractal features are extracted from its fractal dimension.

Geometrical Features

Global geometric features measure the shape of a signature. The height, width (Armand, Blumenstein & Muthukkumarasamy, 2006) and area (or pixel density) (Abdelghani & Amara, 2006) of the signature are basic features pertaining to this category. The height and width can be combined to form the aspect ratio (or caliber) (Oliveira, Justino, Freitas & Sabourin, 2005).

More elaborate geometric features consist of proportion, spacing, and alignment to baseline. Proportion measures the height variations of the signature, while spacing describes the gaps in the signature (Oliveira et al., 2005). Alignment to baseline extracts the general orientation of the signature according to a baseline reference (Abdelghani & Amara, 2006; Armand et al., 2006; Frias-Martinez et al., 2006; Oliveira et al., 2005; Senol & Yildirim, 2005).

Connected components also can be extracted as global features, such as the number of 4-neighbors and 8-neighbors pixels in the signature image (Frias-Martinez et al., 2006).

Statistical Features

Many authors use projection representation. It consists of projecting every pixel on a given axis (usually horizontal or vertical), resulting in

a pixel density distribution. Statistical features, such as the mean (or center of gravity), global, and local maximums, can be extracted from this distribution (Frias-Martinez et al., 2006; Ozgunduz et al., 2005; Senol & Yildirim, 2005). Moments, which can include central moments (i.e., skewness and kurtosis) (Bajaj & Chaudhury, 1997; Frias-Martinez et al., 2006) and moment invariants (Al-Shoshan, 2006; Lv, Wang, Wang, & Zhuo, 2005; Oz, 2005), are also extracted from the pixel distributions.

Moreover, other types of distributions can be extracted from a signature. Sabourin and Drouhard (1992) extracted directional PDF from the gradient intensity representation of the silhouette of a signature. Stroke direction distributions have been extracted using structural elements and morphologic operators (Frias-Martinez et al., 2006; Lv et al., 2005; Madasu, 2006; Ozgunduz et al., 2005). A similar technique is used to extract edge-hinge (strokes changing direction) distributions (Madasu, 2006). Based on an envelope representation of the signature, slope distributions are also extracted in this way (Fierrez-Aguilar, Alonso-Hermira, Moreno-Marquez & Ortega-Garcia, 2004; Lee, Lizárraga, Gomes & Koerich, 1997), whereas Madasu, Hanmandlu, and Madasu (2003) extracted distributions of angles with respect to a reference point from a skeleton representation.

Similarity Features

Similarity features differ from other kinds of features in the sense that they are extracted from a set of signatures. Thus, in order to extract these features, one signature is the questioned signature, while the others are used as references.

In literature, dynamic time warping (DTW) seems to be the matching algorithm of choice. However, since it works with 1D signals, the 2D signature image must be reduced to one dimension. To that effect, projection and envelope representations (Fang, Leung, Tang, Tse, Kwok& Wong, 2003; Kholmatov, 2003) have been used. A weakness of the DTW is that it cumulates errors, and for this reason, the sequences to match must be the shortest possible. To solve this problem, a wavelet transform can be used to extract inflection points from the 1D signal. Then DTW matches this shorter sequence of points (Deng, Jaw, Wang & Tung, 2003). The inflection points can also be used to segment the wavelet signal into shorter sequences to be matched by the DTW algorithm (Ye, Hou & Feng, 2005).

Among other methods, a local elastic algorithm has been used to match the skeleton representations of two signatures (Fang et al., 2003; You, Fang, He & Tang, 2005), and cross-correlation has been used to extract correlation peak features from multiple signature representations obtained from identity and Gabor filters (Fasquel & Bruynooghe, 2004).

Fixed Zoning

Fixed zoning defines arbitrary regions and uses them for all signatures. To perform fixed zoning based on the pixels, all the pixels of a signature are sent to the classifier after a normalization of the signature image to a given size (Frias-Martinez et al., 2006; Martinez, Travieso, Alonso & Ferrer, 2004; Mighell, Wilkinson & Goodman, 1989). Otherwise, numerous fixed zoning methods are described in the literature. Usually, the signature is divided into strips (vertical or horizontal) or uses a layout like a grid or angular partitioning. Then geometric features (Abdelghani & Amara, 2006; Armand et al., 2006; Ferrer, Alonso, & Travieso, 2005; Huang & Yan, 1997; Justino, Bortolozzi, & Sabourin, 2005; Martinez et al., 2004; Ozgunduz et al., 2005; Qi & Hunt, 1994; Santos, Justino, Bortolozzi, & Sabourin, 2004; Senol & Yildirim, 2005), wavelet transform features (Abdelghani & Amara, 2006), and statistical features (Fierrez-Aguilar et al., 2004; Frias-Martinez et al., 2006; Hanmandlu, Yusof & Madasu, 2005; Justino et al., 2005; Madasu, 2006) can be extracted.

Other techniques are specially designed for extracting local features. Strip-based methods include peripheral features extraction from horizontal and vertical strips of a signature edge representation. Peripheral features measure the distance between two edges and the area between the virtual frame of the strip and the edge of the signature (Fang & Tang, 2005; Fang, Leung, Tang, Tse, & Wong, 2002).

Most fixed zoning techniques use a grid layout. For example, the modified direction feature (MDF) technique (Armand et al., 2006) extracts the location of the transitions from the background to the signature and their corresponding direction values for each cell of grid superposed on the signature image. The Gradient, Structural, and Concavity (GSC) technique (Kalera, Srihari & Xu, 2004; Srihari, Xu & Kalera, 2004) extracts gradient features from edge curvature, structural features from short strokes, and concavity features from certain hole types independently for each cell of a grid covering the signature image. The Extended Shadow Code (ESC) technique, proposed by Sabourin and his colleagues (Sabourin, Cheriet & Genest, 1993; Sabourin & Genest, 1994, 1995), centers the signature image on a grid layout. Each rectangular cell of the grid is composed of six bars: one bar for each side of the cell plus two diagonal bars stretching from a corner of the cell to the other in an "X" fashion. The pixels of the signature are projected perpendicularly on the nearest horizontal bars, on the nearest vertical bars, and on both diagonal bars. The features are extracted from the normalized area of each bar that is covered by the projected pixels. The envelope-based technique (Ramesh & Murty, 1999; Bajaj & Chaudhury, 1997) describes for each grid cell the comportment of the upper and lower envelope of the signature. The *pecstrum* technique (Sabourin, Genest & Prêteux, 1996; Sabourin, Genest & Prêteux, 1997b) centers the signature image on a grid of overlapping retinas and then uses successive morphological openings to extract local granulometric size distributions.

Signal-Dependent Zoning

Signal-dependent zoning generates different regions adapted to individual signature. When signal-dependent zoning is performed using the pixels of the signature as local regions, position features are extracted from each pixel with respect to a coordinate system. Ferrer, et al. (2005) and, previously, Martinez, et al. (2004) extracted position features from a contour representation in polar coordinates. Using this same coordinate system, signal-dependent, angular-radial partitioning techniques have been developed. These techniques adjust themselves to the circumscribing circle of the signature to achieve scale invariance. Shape matrices have been defined this way to sample the silhouette of two signatures and extract similarity features (Sabourin, Drouhard, & Wah, 1997a). This technique achieves rotation invariance by synchronizing the sampling with the baseline of the signature. A similar method is used by Chalechale, Naghdy, Premaratne, and Mertins (2004), although edge pixel area features are extracted from each sector and rotation invariance is obtained by applying a 1-D discrete Fourier transform to the extracted feature vector.

In the Cartesian coordinate system, signal-dependent retinas have been used to define local regions capturing best the intrapersonal similarities from the reference signatures of individual writers (Ando & Nakajima, 2003). A genetic algorithm is used to optimize the location and size of these retinas before similarity features are extracted from the questioned signature and its reference set.

Connectivity analysis has been performed on a signature image to generate local regions before extracting geometric and position features from each region (Igarza, Hernaez, & Goirizelaia, 2005). Even more localized regions (signal-dependent) are achieved using stroke segmentation. Perez-Hernandez, Sanchez, and Velez (2004) achieved stroke segmentation by first finding the direction of each pixel of the skeleton of the

signature and then using a pixel tracking process. Then the orientation and endpoints of the strokes are extracted as features. Another technique is to erode the stroke segments into regions before extracting similarity features (Franke, Zhang & Koppen, 2002) from these bloated regions. Instead of focusing on the strokes, the segmentation can be done in other signature representations. Chen and Srihari (2006) matched two signature contours using DTW before segmenting and extracting Zernike moments from the segments. Xiao and Leedham (2002) segmented upper and lower envelopes where their orientation changes sharply. After that, they extracted length, orientation, position, and pointers to the left and right neighbors of each segment.

Pseudo-Dynamic Features

The lack of dynamic information is a serious constrain for off-line signature verification systems. The knowledge of the pen trajectory, along with speed and pressure, gives an edge to online systems. To overcome this difficulty, some approaches use dynamic signature references to develop individual stroke models that can be applied to off-line questioned signatures. For instance, Guo, Doermann, and Rosenfeld (2001) used stroke-level models and heuristic methods to locally compare dynamic and static pen positions and stroke directions. Lau, Yuen, and Tang (2005) developed the universal writing model (UWM), which consists of a set of distribution functions constructed using the attributes extracted from online signature samples, whereas Nel, du Preez, and Herbst (2005) used a probabilistic model of the static signatures based on hidden Markov models (HMM). The HMM restricts the choice of possible of pen trajectories describing the morphology of the signature. Then the optimal pen trajectory is calculated using a dynamic sample of the signature.

However, without resorting to online examples, it is possible to extract pseudo-dynamic features from static signatures. Pressure features can be extracted from pixel intensity (i.e., grey levels) (Huang & Yan, 1997; Lv et al., 2005; Santos et al., 2004; Wen-Ming et al., 2004) and stroke width (Lv et al., 2005; Oliveira et al., 2005), whereas speed information can be extrapolated from stroke curvature (Justino et al., 2005; Santos et al., 2004), stroke slant (Justino et al., 2005; Oliveira et al., 2005; Senol & Yildirim, 2005), progression (Oliveira et al., 2005; Santos et al., 2004), and form (Oliveira et al., 2005).

Discussion

This section presented important feature extraction techniques used to extract global and local information from signatures. The choice of using global or local features will depend mainly on the types of forgeries to be detected by the system. The global features are extracted at a low computational cost, and they have good noise resilience. However, they have less capacity to discriminate between genuine signatures and skilled forgeries. On the other hand, local features are more suitable to identify imitations, despite their dependence on the zoning process.

An issue that has received little attention in literature is the generally low quantity of available signature samples vs. the generally high number of extracted features. This issue may be solved by the following:

1. *Selecting the most discriminating features*: In the work of Xuhua, Furuhashi, Obata, and Uchikawa (1996), for example, genetic algorithms were used to select the optimal set of partial curves from an online signature and the best features of each partial curve.
2. *Using regularization techniques to obtain a stable estimation of the covariance matrix* (Fang & Tang, 2005).
3. *Generating synthetic samples*: This can be done by adding noise or applying transformations to the real signatures (Fang et al.,

2002; Fang & Tang, 2005; Huang & Yan, 1997; Vélez, Sánchez & Moreno, 2003).

4. *Using dissimilarity representation*: this technique makes it possible to reduce the number of classes as well as increase the quantity of feature vectors (Santos et al., 2004).

The two last points are discussed in more detail in the section entitled "Dealing with a Limited Amount of Data."

VERIFICATION STRATEGIES AND EXPERIMENTAL RESULTS

This section categorizes some work in off-line signature verification according to the technique used to perform verification; that is, distance classifiers, artificial neural networks, hidden Markov models, dynamic time warping, support vector machines, structural techniques, and Bayesian networks.

In signature verification, the verification strategy can also be categorized as writer-independent or writer-dependent (Srihari et al., 2004). With writer-independent verification, an n-class classifier deals with the whole population of writers. In contrast, with the writer-dependent verification, a one-class or a two-class classifier is employed per writer. As the majority of the work presented in literature is designed to perform writer-dependent verification, this aspect is mentioned only when writer-independent verification is considered. The same procedure is taken regarding the type of forgeries used for training, since only random forgeries are generally used in this phase.

Before describing the work, some measures used to evaluate the performance of signature verification systems are presented. The section concludes with a discussion of the main challenges to be faced in this field.

Performance Evaluation Measures

In signature verification systems, the simplest way to report their performances is in terms of error rates. The false rejection rate (FRR) is related to genuine signatures that were rejected by the system; that is, classified as forgeries, whereas the false acceptance rate (FAR) is related to forgeries that were misclassified as genuine signatures. FRR and FAR are also known as *type 1* and *type 2* errors, respectively. Finally, the average error rate (AER) is related to the total error of the system; that is, type 1 and type 2 errors together.

On the other hand, if the decision threshold of a system is set to have the percentage of false rejections approximately equal to the percentage of false acceptances, the equal error rate (EER) is calculated.

Distance Classifiers

A simple distance classifier is a statistical technique that usually represents a pattern class with a Gaussian probability density function (PDF). Each PDF is uniquely defined by the mean vector and covariance matrix of the feature vectors belonging to a particular class. When the full covariance matrix is estimated for each class, the classification is based on Mahalanobis distance. On the other hand, when only the mean vector is estimated, classification is based on Euclidean distance (Coetzer, 2005).

Approaches based on distance classifiers are traditionally writer-dependent. The reference samples of a given author are used to compose the class of genuine signatures (w_1), and a subset of samples from each other writer is chosen randomly to compose the class of forgeries (w_2). Once the smallest distance between a reference signature and a questioned signature is found, the latter is classified according to the label of the reference signature (w_1 or w_2). If the classifier is designed to find a number of k nearest reference

signatures, a voting scheme is used to take the final decision.

Distance classifiers were one of the first classification techniques used in off-line signature verification. One of the earliest reported works was by Nemcek and Lin (1974). By using a fast Hadamard transform extraction technique on genuine signatures and simple forgeries, and maximum likelihood classifiers, they obtained an FRR of 11% and an FAR of 41%.

After that, Nagel and Rosenfeld (1977) proposed a system to discriminate between genuine signatures and simple forgeries using images obtained from real bank checks. A number of global and local features were extracted considering only the North American signature style. Using weighted distance classifiers, they obtained an FRR ranging from 8% to 12% and an FAR of 0%.

Some years later, skilled forgeries have begun to be considered in off-line signature verification. Besides proposing a method to separate the signatures from noisy backgrounds and to extract pseudo-dynamic features from static images, Ammar and colleagues (Ammar, 1991; Ammar, Yoshida, & Fukumura, 1986, 1999) were the first to try to detect skilled forgeries in an off-line signature verification system. In their work, distance classifiers were used, combined with the leave-one-out, cross-validation method since the number of signatures examples were small.

Qi and Hunt (1994) presented a signature verification system based on global geometric features and local grid-based features. Different types of similarity measures such as Euclidean distance were used to discriminate between genuine signatures and forgeries (including simple and skilled). They achieved an FRR ranging from 3% to 11.3%, and an FAR ranging from 0% to 15%.

Sabourin and colleagues (Sabourin & Plamondon, 1986; Sabourin et al., 1993) have done extensive research in off-line signature verification since the mid-1980s. In one of their works (Sabourin et al., 1993), the Extended Shadow Code was used in order to extract local features from genuine signatures and random forgeries. The first experiment used a k-nearest neighbors (k-NN) classifier with voting schema, obtaining an AER of 0.01% when k = 1. The second experiment used a minimum distance classifier, obtaining an AER of 0.77% when 10 training signatures were used for each writer. In another relevant work, Sabourin et al. (1997b) used granulometric size distributions as local features, also in order to eliminate random forgeries. By using k-nearest neighbors and threshold classifiers, they obtained an AER around 0.02% and 1.0%, respectively.

Fang, Wang, Leung, and Tse (2001) developed a system based on the assumption that the cursive segments of skilled forgeries are generally less smooth than those of genuine signatures. Besides the utilization of global shape features, a crossing and fractal dimension methods were proposed to extract the smoothness features from the signature segments. Using a simple distance classifier and the leave-one-out, cross-validation method, an FRR of 18.1% and an FAR of 16.4% were obtained. More recently, Fang, et al. (2002) extracted a set of peripheral features in order to describe internal and the external structures of the signatures. To discriminate between genuine signatures and skilled forgeries, they used a Mahalanobis distance classifier together with the leave-one-out, cross-validation method. The obtained AERs were in the range of 15.6% (without artificially generated samples) and 11.4% (with artificially generated samples).

Artificial Neural Networks

An artificial neural network (ANN) is a massively parallel distributed system composed of processing units capable of storing knowledge learned from experience (examples) and using it to solve complex problems (Haykin, 1998). Multilayer perceptron (MLP) trained with the error back propagation algorithm (Rumelhart, Hinton & Wil-

liam, 1986) has so far the most frequently ANN architecture used in pattern recognition.

Particularly in off-line signature verification, ANNs have been used extensively both in writer-independent and writer-dependent approaches. To perform writer-independent verification, the network is generally trained by using one class per writer. On the other hand, writer-dependent verification is generally performed by using two classes: one for the genuine signatures and another for the forgeries.

Mighell, et al. (1989) were the first ones to apply ANNs for off-line signature verification. In order to eliminate simple forgeries, they used the raw images as input to an MLP. In the experiments, by using a training set composed by genuine signatures and forgeries, they achieved an EER of 2%.

Sabourin and Drouhard (1992) used directional PDFs as global feature vectors and MLP as classifier in order to eliminate random forgeries. Since their database was composed of few data, some signature samples were generated by rotating the directional PDFs. In the experiments, they obtained an FRR of 1.75% and an FAR of 9%.

Cardot, Revenu, Victorri, and Revillet (1994) used outline and geometric measures of the signature images to compose two types of feature vectors. The most important contribution of their work was the proposal of a multilevel neural network architecture to eliminate random forgeries. The first level is composed of two Kohonen maps (one for each set of features) in order to perform an initial classification and to choose the random forgeries to train the second level networks. As the number of writers was very large (more than 300), they had to limit the number of classes to fewer than 50. In the second level, two MLPs for each writer are used to perform writer-dependent verification. Finally, in the last level, an MLP accepts or rejects the signature. By using a dataset of signatures extracted from real postal checks, they achieved an FRR of 4% and an FAR of 2%.

Murshed, et al. (1995) proposed a verification strategy based on fuzzy ARTMAPs in the context of random forgeries. Different from other neural networks types, the fuzzy ARTMAPs allow training by using examples of only one class. Therefore, in this approach, the genuine signatures are used for training and the random forgeries (as well as some unseen genuine signatures samples) for testing. In order to simulate different experts examining different regions of the signature, the image is divided in a number of overlapping squares according to the writer signature shape. After that, each signature region is reduced by applying an MLP network, and verified by a specialized fuzzy ARTMAP. Finally, based on the results given by each fuzzy ARTMAP, the final decision is taken. In the experiments, they obtained an AER of 9.14%.

Bajaj and Chaudhury (1997) used an ensemble of MLPs to perform writer-independent verification. In order to discriminate between genuine signatures and random forgeries, one MLP per feature vector (moments, upper envelope, and lower envelop) was trained. Moreover, each MLP was composed of 10 outputs (one for each writer). In the verification phase, the output of the three classifiers was combined to obtain a final decision. In the experiments, a substantial reduction of the error rate was obtained when using the three classifiers together (FRR=1%; FAR=3%).

Fadhel and Bhattacharyya (1999) proposed a signature verification system based on Steerable Wavelets as feature extraction technique and MLP as classifier. In the first experiment, by selecting only the first two of the 16 coefficients, which represent each signature image, they obtained a classification rate of 85.4%, whereas in a second experiment, by using all 16 coefficients, the classification rate was improved to 93.8%.

Sansone and Vento (2000) proposed a three-stage, multi-expert system in order to deal with all types of forgeries. The first stage was designed to eliminate random and simple forgeries by using only the signature's *outline* as a feature. The

second stage receives the signatures accepted by the previous stage, which can be classified as genuine or as skilled forgery. The features used in this stage are the high-pressure regions. Finally, a third stage takes the final decision. Using MLP as classifiers, they obtained an FRR of 2.04% and FARs of 0.01%, 4.29%, and 19.80% with respect to random, simple, and skilled forgeries, respectively.

Blatzakis and Papamarkos (2001) used global geometric features, grid features, and texture features to represent the signatures. They proposed a two-stage system in order to eliminate random forgeries. In the first stage, three MLPs (one for each feature set) and the Euclidean distance metric perform a coarse classification. After that, an RBF (radial basis function) neural network, trained with samples that were not used in the first stage, takes the final decision. An FRR of 3% and an FAR of 9.8% were obtained in the experiments.

Quek and Zhou (2002) proposed a system based on fuzzy neural networks in order to eliminate skilled forgeries. To represent the signatures, they used reference pattern-based features, global baseline features, pressure features, and slant features. In the first set of experiments, using both genuine signatures and skilled forgeries to train the network, an average EER of 22.4% was obtained. Comparable results were obtained in the second set of experiments, in which only genuine signatures were used as training data.

Vélez, et al. (2003) performed signature verification by comparing subimages or positional cuttings of a test signature to the representations stored in compression neural networks. In this approach, neither image preprocessing nor feature extraction is performed. By using one signature per writer, together with a set of artificially generated samples, they obtained a classification rate of 97.8%.

In a recent work, Armand, et al. (2006) proposed the combination of the modified direction feature (MDF), extracted from the signature's contour, with a set of geometric features. In the experiments, they compared RBF and resilient backpropagation (RBP) neural network performances. Both networks performed writer-independent verification and contained 40 classes—39 corresponding to each writer and one corresponding to the forgeries. In this case, skilled forgeries were used in the training phase. The best classification rates obtained were 91.21% and 88.0%, using RBF and RBP, respectively.

Hidden Markov Models

Hidden Markov models (Rabiner, 1989) are finite stochastic automata used to model sequences of observations. Although this technique is more suitable to model dynamic data (e.g., as speech and online signatures), it has also been applied in segmented off-line signatures. Generally, HMMs are used to perform writer-dependent verification by modeling only the genuine signatures of a writer. In this case, the forgeries are detected by thresholding.

Rigoll and Kosmala (1998) presented a comparison between online and off-line signature verification using discrete HMMs. To represent the signatures in the online model, they used both static and pseudo-dynamic features. In the first set of experiments, in which each feature was investigated separately, surprising results were obtained. The bitmap feature was the most important one, achieving a classification rate of 92.2%. The Fourier feature also supplied a high classification rate. Finally, another surprise was the low importance of the acceleration. As expected, good results were obtained using the velocity feature. Other experiments using several features together were performed in order to obtain high classification rates. The best result (99%) was obtained when only four features (bitmap, velocity, pressure, and Fourier feature) where combined.

To represent the signatures in the off-line model, they subdivided the signature image into several squares of 10x10 pixels. After that, the

grey value of each square was computed and used as a feature. In the experiments, a classification rate of 98.1% was achieved. The small difference between the online and off-line classification rates is an important practical result, since off-line verification is simpler to implement.

El-Yacoubi, Justino, Sabourin, and Bortolozzi (2000) proposed an approach based on HMM and pixel density features in order to eliminate random forgeries. To perform training while choosing the optimal HMMs' parameters, the Baum-Welch algorithm and the cross-validation method were used. In the experiments, each signature was analyzed under three resolutions (100x100, 40x40, and 16x16 pixels) by applying the Forward algorithm. Finally, a majority-vote rule took the final decision. An AER of 0.46% was obtained when both genuine and impostor spaces were modeled, and AER of 0.91% was obtained when only the genuine signatures were modeled.

Justino, Bortolozzi, and Sabourin (2001) used HMMs to detect random, simple, and skilled forgeries. Also using a grid-segmentation scheme, three features were extracted from the signatures: pixel density feature, Extended Shadow Code, and axial slant feature. They applied the cross-validation method in order to define the number of states for each HMM writer model. Using the Bakis model topology and the forward algorithm, they obtained an FRR of 2.83% and FARs of 1.44%, 2.50%, and 22.67% for random, simple, and skilled forgeries, respectively.

Coetzer, et al. (2004) used HMMs and discrete random transforms to detect simple and skilled forgeries. In this work, some strategies were proposed in order to obtain noise, shift, rotation, and scale invariances. By using a left-to-right ring model and the Viterbi algorithm, EERs of 4.5% and 18% were achieved for simple and skilled forgeries, respectively.

Dynamic Time Warping

Widely applied in speech recognition, dynamic time warping (DTW) is a template matching technique used for measuring similarity between two sequences of observations. The primary objective of DTW is to nonlinearly align the sequences before they are compared (matched) (Coetzer, 2005). Despite being more suitable to model data that may vary in time or speed, dynamic time warping has been used in off-line signature verification. As usually occurs in HMM-based approaches, a test signature is compared to the genuine ones of a writer (writer-dependent verification), and a forgery is detected by thresholding.

Wilkinson and Goodman (1990) used DTW to discriminate between genuine signatures and simple forgeries. Assuming that curvature, total length, and slant angle are constant among different signatures of a same writer, they used a slope histogram to represent each sample. In the experiments, they obtained an EER of 7%. Increases in the error rates were observed when the forgers had some a priori knowledge about the signatures.

Deng, Liao, Ho, and Tyan (1999) proposed a Wavelet-based approach to eliminate simple and skilled forgeries. After applying a closed-contour tracing algorithm to the signatures, the curvature data obtained were decomposed into multiresolutional signals using wavelets. Then DTW was used to match the corresponding zero-crossings. Experiments were performed using English and Chinese signature datasets. For the English dataset, an FRR of 5.6% and FARs of 21.2% (skilled forgeries) and 0% (simple forgeries) were obtained, whereas using the Chinese dataset, an FRR of 6.0% and FARs of 13.5% (skilled forgeries) and 0% (simple forgeries) were achieved.

Fang, et al. (2003) proposed a method based on DTW and one-dimensional projection profiles in order to deal with intrapersonal signature variations. To achieve discrimination between genuine signatures and skilled forgeries, nonlinear DTW was used in a different way. Instead of using the distance between a test signature and a reference sample to take a decision, the positional distortion at each point of the projection profile was

incorporated into a distance measure. Using the leave-one-out, cross-validation method and the Mahalanobis distance, they obtained AERs of 20.8% and 18.1%, when binary and grey level signatures were considered, respectively.

Support Vector Machines

Support vector machines (SVMs) (Vapnik, 1999) is a kernel-based learning technique that has shown successful results in applications of various domains (e.g., pattern recognition, regression estimation, density estimation, novelty detection, etc.).

Signature verification systems that use SVMs as classifiers are designed in a similar way to those that use neural networks. That is, in a writer-dependent approach, there is one class for the genuine signatures and another class for the forgeries. In addiction, by using one-class SVMs (Scholkopf, Platt, Taylor, Smola & Williamson, 2001), it is possible to perform training by using only genuine signatures. In the work of Srihari, et al. (2004), the authors tried to use it in the context of skilled forgeries. However, by using the traditional two-class approach, the AER decreased from 46.0% to 9.3%.

Martinez, et al. (2004) used SVM with RBF kernel in order to detect skilled forgeries. In the experiments, different types of geometrical features as well as raw signatures were tested. The best result—an FAR of 18.85%—was obtained when raw images with a scale of 0.4 were used.

Justino, et al. (2005) performed a comparison between SVM and HMM classifiers in the detection of random, simple, and skilled forgeries. By using a grid-segmentation scheme, they extracted a set of static and pseudo-dynamic features. Under different experimental conditions (i.e., varying the size of the training set and the types of forgeries), the SVM with a linear kernel performed better than the HMM.

Ozgunduz, et al. (2005) used support vector machines in order to detect random and skilled forgeries. To represent the signatures, they extracted global geometric features, direction features, and grid features. In the experiments, a comparison between SVM and ANN was performed. Using an SVM with RBF kernel, an FRR of 0.02% and an FAR of 0.11% were obtained, whereas the ANN, trained with the backpropagation algorithm, provided an FRR of 0.22% and an FAR of %0.16. In both experiments, skilled forgeries were used to train the classifier.

Structural Techniques

In structural techniques, the patterns are organized hierarchically in a way that in each level, they are viewed as being composed of simpler subpatterns. By using a small number of primitives (the most elementary subpatterns) and grammatical rules, it is possible to describe a large collection of complex patterns (Coetzer, 2005). Therefore, it is possible to interpret the scene both globally and locally.

Sabourin, Plamondon, and Beaumier (1994) were the first ones to propose a structural representation of handwritten signatures images. In their approach, a segmentation process breaks up the signature into a set of primitives. From these shape primitives, both static and pseudo-dynamic features are extracted. The comparison process is composed of two stages: Local interpretation of primitives (LIP) and global interpretation of the scene (GIS). In the LIP stage, a template match process is performed in which each primitive of the test signature image is labeled, taking into account the reference set. Finally, the GIS stage takes the final decision by computing a similarity measure between the test primitive set and the reference primitive set. The experiments were performed in order to eliminate random forgeries. By using a minimum distance classifier with two reference signatures, they obtained an AER of 1.43%.

Bastos, Bortolozzi, Sabourin, and Kaestner (1997) proposed a mathematical signature representation in terms of ellipses, parabolas, and

hyperboles. The goal of this approach is to allow a simplification of the signature tracing when detecting random forgeries. After performing a thinning process, junction and endpoints are found in the signatures. Next, an algorithm is applied in each part of the signature tracing (between an endpoint and a junction point) in order to obtain all necessary points for modeling a mathematical equation. Finally, the least square method of curve adjusting is applied to each set of tracing points, resulting in a number of equations of ellipses, parabolas, and hyperboles. Performing superpositions between the found equations and the respective signatures, they obtained similarity indices ranging from 86.3% and 97.3% with respect to different writers.

Huang and Yan (2002) proposed a two-stage signature verification system based on ANN and a structural approach. To represent the signatures, they used geometric and directional frontier features. In the first stage of the system, a neural network attributes to the signature three possible labels: pass (genuine signature), fail (random or less skilled forgery), and questionable (skilled forgery). For the questionable signatures, the second stage uses a structural feature verification algorithm to compare the detailed structural correlation between the test signature and the reference samples. In the experiments, the first classifier rejected 2.2% of the genuine signatures, accepted 3.6% of the forgeries, and was undecided on 32.7% of the signatures. The second classifier rejected 31.2% of the questionable genuine signatures and accepted 23.2% of the questionable forgeries. Therefore, for the combined classifier, an FRR of 6.3% and an FAR of 8.2% were obtained.

Discussion

This section presented some verification strategies proposed in the field of off-line signature verification. Even though the error rates are reported, it is very difficult to compare the performances of different verification strategies, since each work uses different feature extraction techniques, experimentation protocols, and signature databases (see Tables 1 and 2).

Despite the great recognition rates obtained by using classifiers, which learn from examples (e.g., ANNs, HMMs, and SVMs), there are some difficulties to be faced. The first one is the large number of examples required to ensure that the classifier will, in fact, learn (Leclerc & Plamondon, 1994). On the other hand, classifiers that do not require many reference samples, since there is no explicit training phase (e.g., distance classifiers), have a low generalization capability.

Another difficulty, which occurs mainly with ANNs and SVMs, is the necessity of using forgeries in the training phase in order to allow class separation by the classifier. However, the authors have already been dealing with this problem by using one-class classifiers (Murshed et al., 1995; Srihari et al., 2004), computer-generated forgeries (Mighell et al., 1989), and a subset of genuine signatures from other writers (random forgeries).

The utilization of multiple classifiers has improved the recognition performance in difficult classification problems. Using just one optimal classifier, it is possible to lose valuable information contained in the other suboptimal classifiers. It has been show that when a set of R classifiers is averaged, the variance contribution in the bias-variance decomposition decreases by 1/R, resulting in a smaller expected error (Tax, 2001).

In a multistage approach, each classification level receives the results of the previous one, reducing the complexity of the problem (Blatzakis & Papamarkos, 2001; Huang & Yan, 2002; Sansone & Vento, 2000). Particularly in signature verification, the complexity can be reduced even further if the system combines both writer-independent verification (to eliminate random and simple forgeries) and writer-dependent verification (to eliminate the skilled forgeries). However, few systems (Cardot et al., 1994) use this strategy to improve their performances.

Moreover, ensemble of classifiers has been less explored in this field (Bajaj & Chaudhury,

1997; Cardot et al., 1994; Sabourin et al., 1997b). In this approach, classifiers may be combined in parallel by changing (1) the training set (Breiman, 1996), (2) the input features (Alkoot & Kittler, 2000; Günter & Bunke, 2002), and (3) the parameters/architecture of the classifier. It is, therefore, possible to have an ensemble of classifiers in each level of a multistage system.

Another question is related to the number of classes used to perform writer-independent verification. Usually, this task has been done by using one class per writer (Armand et al., 2006; Bajaj & Chaudhury, 1997), a strategy not suitable to be applied in real-world applications, in which the number of writers to be verified is generally high.

Finally, one aspect that has not been discussed in the signature verification literature is the use of incremental learning. As a signature varies according to psychological and physical state, it is very difficult to get all possible variations during the training phase. Besides, the signature may change over time. Thus, updating the classifier as new examples are available may be useful in real signature verification systems.

DEALING WITH A LIMITED AMOUNT OF DATA

Mainly for practical reasons, a limited number of signatures by writers is available to train a

Table 1. Signature verification databases (I = Individual; G = Genuine; F = forgeries; S = Samples)

References	Images	Signatures	Forgery Types
(Nemcek & Lin, 1974)	128x256 pixels binary	600G / 15I 120F / 4I	Simple
(Nagel & Rosenfeld, 1977)	500 ppi 64 grey levels	11G / 2I 14F / 2I	Simple
(Ammar et al., 1986)	256x1024 pixels 256 grey levels	200G / 10I 200F / 10I	Skilled
(Qi & Hunt, 1994)	300 dpi 256 grey levels	300G / 15I 150F / 10I	Simple and Skilled
(Sabourin et al., 1993) (Sabourin et al., 1997b) (Sabourin & Drouhard, 1992) (Sabourin et al., 1994)	128x512 pixels 256 grey levels	800G / 20I	Random
(Fang et al., 2002) (Fang et al., 2003)	300 dpi 256 grey levels	1320G / 55I 1320F / 12I	Skilled
(Mighell et al., 1989)	128x64 pixels binary	80G / 1I 66F	Skilled
(Cardot et al., 1994)	1024x512 pixels 256 grey levels	6000G/ 300I	Random
(Murshed et al., 1995)	128x512 pixels 256 grey levels	200G / 5I	Random
(Bajaj & Chaudhury, 1997)	200 dpi binary	150G / 10I	Random
(Fadhel & Bhattacharyya, 1999)	340 dpi 256 grey levels	300S / 30I	Skilled
(Sansone & Vento, 2000)	300 dpi 256 grey levels	980G / 49I 980F / 49I	Simple and Skilled

classifier for signature verification.[1] However, by using a small training set, the class statistics estimation errors may be significant, resulting in unsatisfactory verification performance (Fang & Tang, 2005).

Huang and Yan (1997) applied slight transformations to the genuine signatures in order to generate additional training samples, and heavy transformations, also to the genuine signatures, in order to generate forgeries. In the two cases, the transformations were slant distortions, scalings in horizontal and vertical directions, rotations, and perspective view distortions; whereas Vélez et al. (2003) tried to reproduce intrapersonal variability by using only one signature per writer. To generate additional training samples, they applied rotations (in the range of ±15°), scalings (in the range of ±20%), horizontal and vertical displacements (in the range of ±20%), and various types of noise for each original signature.

By using a different approach, Fang and Tang (2005) proposed the generation of additional samples in the following way:

1. Two samples are selected from the set of genuine signatures.
2. Then an elastic matching algorithm is applied to the pair of signatures in order to estab-

Table 2. Signature verification databases (continuation) (I = Individual; G = Genuine; F = forgeries; S = Samples)

References	Images	Signatures	Forgery Types
(Blatzakis & Papamarkos, 2001)	binary	2000G / 115I	Random
(Quek & Zhou, 2002)	516x184 pixels 256 grey levels	535G / 24I 15-20F x 5I	Skilled
(Vélez et al., 2003)	300 dpi 256 grey levels	112S / 28I	not specified
(Armand et al., 2006)	not specified	936G / 39I 1170F / 39I	Skilled
(Rigoll & Kosmala, 1998)	not specified	280G / 14I 60F	Simple and Skilled
(El-Yacoubi et al., 2000)	300 dpi binary	4000G / 100I	Random
(Justino et al., 2001) (Justino et al., 2005)	300 dpi 256 grey levels	4000G / 100I 1200F / 10I	Simple and Skilled
(Coetzer, 2005)	300 dpi binary	660G / 22I 264F / 6I	Simple and Skilled
(Deng et al., 1999)	600 ppi 256 grey levels	1000G / 50I 2500G / 50I	Simple and Skilled
(Srihari et al., 2004)	300 dpi 256 grey levels	1320G/ 55I 1320F / 55I	Skilled
(Martinez et al., 2004)	not specified	3840G / 160I 4800F / 160I	Skilled
(Ozgunduz et al., 2005)	256 grey levels	1320S / 70I	Skilled
(Bastos et al., 1997)	not specified	120G / 6I	Random
(Huang & Yan, 2002)	100 dpi 256 grey levels	1272G / 53I 7632F / 53I	Skilled

lish correspondences between individual strokes.
3. Next, corresponding stroke segments are linked up by displacement vectors.
4. Finally, these displacement vectors are used to perform an interpolation between the two signatures, thus to produce a new training sample.

Based on the dissimilarity representation approach (Bicego, Murino & Figueiredo, 2004; Cha, 2001; Pekalska & Duin, 2000) in which an object is described by its distances with respect to a predetermined set of prototypes, Santos, et al. (2004) solved the problem of having a limited number of samples to perform signature verification. Instead of using one class per writer to train a global classifier, only two classes are used: genuine and forgery. After the usual feature extraction phase, new feature vectors are generated in the following way:

1. Compute the Euclidean distance vector between each pair of signatures.
2. If the pair of signatures belongs to the same writer, set the feature vector to 1; otherwise, set the feature vector to 0.
3. Finally, train the classifier by using these vectors.

In the verification phase, the distance vectors are computed between the input signature and the reference vectors of its probable class and sent as input to the classifier. The final decision is taken by combining all classifier outputs in voting schema. Similar signature verification approaches have also been developed by Srihari and colleagues (Kalera et al., 2004; Srihari et al., 2004).

CONCLUSION

This chapter presented a survey of techniques developed in the field of off-line signature verification over the last 20 years. As we could observe, despite the vast amount of work performed in order to solve this problem, there are still many challenges to be faced due to the investigation of the trade-off between the quantity of available training samples and the number of extracted features to proposals of powerful verification strategies to deal with all the types of forgeries and dynamic environments. Moreover, for security reasons, it is not easy to make a signature dataset available in the signature verification community, mainly if the signatures come from a real situation (e.g., banking documents). However, the availability of datasets could make it possible to define a common experimentation protocol in order to perform comparative studies in this field.

Dissimilarity representation is an interesting approach because although it copes with the problem of having a reduced training set, it also solves the problem of having many classes in a writer-independent verification. Thus, the combination of this approach with SVM (to perform writer-independent verification) and with HMM (to perform writer-dependent verification) may be an interesting choice of a multistage system. Moreover, the utilization of SVMs facilitates the implementation of a reject mechanism based on ROC (receiver operating characteristic) curves (Fawcett, 2006; Tortorella, 2005).

Finally, regarding feature extraction techniques, the ESC (Sabourin et al., 1993) appears to be a good trade-off between global and local features since it permits the projection of the handwriting at several resolutions.

REFERENCES

Abdelghani, I., & Amara, N. (2006). Deux approches neuronales pour la vérification hors-ligne de la signature manuscrite. *Proceedings of the Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle.*

Alkoot, F., & Kittler, J. (2000). Feature selection for an ensemble of classifiers. *Proceedings of the World Multiconference on Systematics, Cybernetics and Informatics.*

Al-Shoshan, A. (2006). Handwritten signature verification using image invariants and dynamic features. *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, 173–176.

Ammar, M. (1991). Progress in verification of skillfully simulated handwritten signatures. *International Journal of Pattern Recognition and Artificial Intelligence*, *5*(1-2), 337–351.

Ammar, M., Yoshida, Y., & Fukumura, T. (1986). A new effective approach for off-line verification of signatures by using pressure features. *Proc. 8th Int. Conf. on Pattern Recognition* (pp. 566–569).

Ammar, M., Yoshida, Y., & Fukumura, T. (1999). Off-line preprocessing and verification of signatures. *International Journal of Pattern Recognition and Artificial Intelligence*, *2*(4), 589–602.

Ando, S., & Nakajima, M. (2003). An active search method for local individual features in off-line signature verification. *Systems and Computers in Japan*, *34*(12), 64–76.

Armand, S., Blumenstein, M., & Muthukkumarasamy, V. (2006). Off-line signature verification using the enhanced modified direction feature and neural-based classification. *Proceedings of the International Joint Conference on Neural Networks*, 1663–1669.

Bajaj, R., & Chaudhury, S. (1997). Signature verification using multiple neural classifiers. *Pattern Recognition*, *30*, 1–7.

Bastos, L., Bortolozzi, F., Sabourin, R., & Kaestner, C. (1997). Mathematical modelation of handwritten signatures by conics. *Revista da Sociedade Paranaese de Matemática*, *18*, 135–146.

Bicego, M., Murino, V., & Figueiredo, M. (2004). Similarity based clustering of sequences using hidden Markov models. *Pattern Recognition*, *37*(12), 2281–2291.

Blatzakis, H., & Papamarkos, N. (2001). A new signature verification technique based on a two-stage neural network classifier. *Engineering Applications of Artificial Intelligence*, *14*, 95–103.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *2*, 123–140.

Cardot, H., Revenu, M., Victorri, B., & Revillet, M. (1994). A static signature verification system based on a cooperating neural network architecture. *International Journal on Pattern Recognition and Artificial Intelligence*, *8*(3), 679–692.

Cha, S. (2001). *Use of distance measures in handwriting analysis* [unpublished doctoral dissertation]. Buffalo, NY: State University of New York at Buffalo.

Chalechale, A., Naghdy, G., Premaratne, P., & Mertins, A. (2004). Document image analysis and verification using cursive signature. *IEEE International Conference on Multimedia and Expo*, *2*, 887– 890.

Chen, S., & Srihari, S. (2006). Combining one- and two-dimensional signal recognition approaches to off-line signature verification. *Proceedings of the Document Recognition and Retrieval XIII*, 606701 1- 606701 10.

Chuang, P. (1977). *Machine verification of handwritten signature image.* International Conference on Crime Countermeasures-Sci, 105–109.

Coetzer, J. (2005). *Off-line signature verification* [unpublished doctoral dissertation]. University of Stellenbosch.

Coetzer, J., Herbst, B., & du Preez, J. (2004). Off-line signature verification using the discrete radon transform and a hidden Markov model.

EURASIP Journal on Applied Signal Processing, 4, 559–571.

Deng, P., Jaw, L.-J., Wang, J.-H., & Tung, C.-T. (2003). Trace copy forgery detection for handwritten signature verification. *Proceedings of the IEEE International Carnahan Conference on Security Technology.*

Deng, P., Liao, H., Ho, C., & Tyan, H. (1999). Wavelet-based off-line handwritten signature verification. *Computer Vision and Image Understanding, 76*(3), 173–190.

El-Yacoubi, A., Justino, E., Sabourin, R., & Bortolozzi, F. (2000). Off-line signature verification using HMMS and cross-validation. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 859–868.

Fadhel, E., & Bhattacharyya, P. (1999). Application of a steerable wavelet transform using neural network for signature verification. *Pattern Analysis and Applications, 2*, 184–195.

Fang, B., Leung, C., Tang, Y., Tse, K., Kwok, P., & Wong, Y. (2003). Off-line signature verification by tracking of feature and stroke positions. *Pattern Recognition, 36*, 91–101.

Fang, B., Leung, C., Tang, Y., Tse, K., & Wong, Y. (2002). Off-line signature verification with generated training samples. *Proceedings of the IEE Vision, Image and Signal Processing.* Vol. 149, 85–90.

Fang, B., & Tang, Y. (2005). Improved class statistics estimation for sparse data problems in off-line signature verification. *IEEE Transactions on Systems, Man and Cybernetics, 35*(3), 276–286.

Fang, B., Wang, Y., Leung, C., & Tse, K. (2001). Off-line signature verification by the analysis of cursive strokes. *International Journal of Pattern Recognition and Artificial Intelligence, 15*(4), 659–673.

Fasquel, J., & Bruynooghe, M. (2004). A hybrid opto-electronic method for fast off-line handwritten signature verification. *International Journal on Document Analysis and Recognition, 7*(1), 56–68.

Fawcett, T. (2006). An Introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Ferrer, M., Alonso, J., & Travieso, C. (2005). Off-line geometric parameters for automatic signature verification using fixed-point arithmetic. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 993–997.

Fierrez-Aguilar, J., Alonso-Hermira, N., Moreno-Marquez, G., & Ortega-Garcia, J. (2004). An off-line signature verification system based on fusion of local and global information. *Lecture Notes in Computer Science: Biometric Authentication, 3087*, 295–306.

Franke, K., Zhang, Y.-N., & Koppen, M. (2002). Static signature verification employing a Kosko-neuro-fuzzy approach. *Proceedings of the International Conference on Fuzzy Systems - Advances in Soft Computing*, 185–190.

Frias-Martinez, E., Sanchez, A., & Velez, J. (2006). Support vector machines versus multilayer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence, 19*(6), 693–704.

Gonzalez, R. & Woods, R. (2002). In *Digital Image Processing, 2nd edition*. Upper Saddle River, NJ: Prentice Hall.

Griess, F., & Jain, A. (2002). On-line signature verification. *Pattern Recognition, 35*, 2963-2972.

Günter, S., & Bunke, H. (2002). Creation of classifier ensembles for handwritten word recognition using feature selection algorithms. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 183–188.

Guo, J., Doermann, D., & Rosenfeld, A. (2000). Off-line skilled forgery detection using stroke and sub-stroke properties. *Proceedings of the 15th International Conference on Pattern Recognition*, 2355–2358.

Hanmandlu, M., Yusof, M., & Madasu, V. (2005). Off-line signature verification and forgery detection using fuzzy modeling. *Pattern Recognition*, *38*(3), 341–356.

Haykin, S. (1998). *Neural networks, a comprehensive foundation*. 2nd edition. Prentice Hall.

Huang, K., & Yan, H. (1997). Off-line signature verification based on geometric feature extraction and neural network classification. *Pattern Recognition*, *30*(1), 9–171.

Huang, K., & Yan, H. (2002). Off-line signature verification using structural feature correspondence. *Pattern Recognition*, *35*, 2467–2477.

Igarza, J., Hernaez, I., & Goirizelaia, I. (2005). Static signature recognition based on left-to-right hidden Markov models. *Journal of Electronic Imaging*, *14*(4).

Justino, E. (2001). *O grafismo e os modelos escondidos de Markov na verificação automática de assinaturas* [unpublished doctoral dissertation]. Brazil: PUC-PR.

Justino, E., Bortolozzi, F., & Sabourin, R. (2001). Off-line signature verification using HMM for random, simple and skilled forgeries. *Proceedings of the International Conference on Document Analysis and Recognition*, 105–110.

Justino, E., Bortolozzi, F., & Sabourin, R. (2005). A comparison of SVM and HMM classifiers in the off-line signature verification. *Pattern Recognition Letters*, *26*(9), 1377–1385.

Kalera, M., Srihari, S., & Xu, A. (2004). Off-line signature verification and identification using distance statistics. *International Journal of Pattern Recognition and Artificial Intelligence*, *18*(7), 1339–1360.

Kholmatov, A. (2003). *Biometric identity verification using on-line and off-line signature verification* [unpublished master's dissertation]. Sabanci University.

Kung, S., Mak, M., & Lin, S. (2004). *Biometric authentication, a machine learning approach*. Prentice Hall.

Lau, K., Yuen, P., & Tang, Y. (2005). Universal writing model for recovery of writing sequence of static handwriting images. *International Journal of Pattern Recognition and Artificial Intelligence*, *19*, 603–630.

Leclerc, F., & Plamondon, R. (1994). Automatic signature verification, the state of the art—1989-1993. *International Journal of Pattern Recognition and Artificial Intelligence*, *8*(3), 643–660.

Lee, L., Lizárraga, M., Gomes, N., & Koerich, A. (1997). A prototype for Brazilian bankcheck recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, *11*(4), 549–569.

Lv, H., Wang, W., Wang, C., & Zhuo, Q. (2005). Off-line Chinese signature verification based on support vector machines. *Pattern Recognition Letters*, *26*(15), 2390–2399.

Madasu, V. (2006). *Automatic bank check processing and authentication using signature verification* [unpublished doctoral dissertation]. Australia: University of Queensland.

Madasu, V., Hanmandlu, M., & Madasu, S. (2003). Neuro-fuzzy approaches to signature verification. *Proceedings of the 2nd National Conference on Document Analysis and Recognition*.

Martinez, L., Travieso, C., Alonso, J., & Ferrer, M. (2004). Parameterization of a forgery handwritten signature verification system using SVM. *Proceedings of the International Carnahan Conference on Security Technology*, 193–196.

Mighell, D., Wilkinson, T., & Goodman, J. (1989). Backpropagation and its application to handwritten signature verification. In Touretzky, D. (Ed.), *Advances in neural information processing systems I* (pp. 340–347). San Mateo, CA: Morgan Kaufmann.

Murshed, N., Bortolozzi, F., & Sabourin, R. (1995). Off-line signature verification using fuzzy artmap neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, 2179–2184.

Nagel, R., & Rosenfeld, A. (1977). Computer detection of freehand forgeries. *IEEE Transactions on Computers, 26*, 895–905.

Nel, E.-M., du Preez, J., & Herbst, B. (2005). Estimating the pen trajectories of static signatures using hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27,* 1733–1746.

Nemcek, W., & Lin, W. (1974). Experimental investigation of automatic signature verification. *IEEE Transactions on Systems, Man and Cybernetics, 4*, 121–126.

Oliveira, L., Justino, E., Freitas, C., & Sabourin, R. (2005). The graphology applied to signature verification. *Proceedings of the 12th Conference of the International Graphonomics Society*, 286–290.

Oz, C. (2005). Signature recognition and verification with artificial neural network using moment invariant method. *Lecture Notes in Computer Science, 3497,* 195–202.

Ozgunduz, E., Senturk, T., & Karsligil, E. (2005). Off-line signature verification and recognition by support vector machine. *Proceedings of the European Signal Processing Conference*.

Pekalska, E., & Duin, R. (2000). Classifiers for dissimilarity-based pattern recognition. *Proceedings of the International Conference on Pattern Recognition,* Vol. 2, 12–16.

Perez-Hernandez, A., Sanchez, A., & Velez, J. (2004). Simplified stroke-based approach for off-line signature recognition. *Proceedings of the 2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications*, 89–94.

Plamondon, R. (Ed.). (1994). *Progress in automatic signature verification*. Singapore: Word Scientific.

Plamondon, R., & Lorette, G. (1989). Automatic signature verification and writer identification—The state of the art. *Pattern Recognition, 22*, 107–131.

Qi, Y., & Hunt, B. (1994). Signature verification using global and grid features. *Pattern Recognition, 27*(12), 1621–1629.

Quek, C., & Zhou, R. (2002). Antiforgery, a novel pseudo product based fuzzy neural network driven signature verification system. *Pattern Recognition Letters, 23*, 1795–1816.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE, 77*(2), 257–286.

Ramesh, V., & Murty, M. (1999). Off-line signature verification using genetically optimized weighted features. *Pattern Recognition*, *32*, 217–233.

Rigoll, G., & Kosmala, A. (1998). A systematic comparison between on-line and off-line methods for signature verification with hidden Markov models. *Proceedings of the International Conference on Pattern Recognition*, Vol. 2, 1755–1757.

Rumelhart, D., Hinton, G., & William, R. (1986). *Learning internal representations by error propagation*. Vol. 1. MIT Press.

Sabourin, R., Cheriet, M., & Genest, G. (1993). An extended shadow-code based approach for off-line signature verification. *Proceedings of the International Conference on Document Analysis and Recognition*, 1–5.

Sabourin, R., & Drouhard, J. (1992). Off-line signature verification using directional pdf and neural networks. *Proceedings of the International Conference on Pattern Recognition*, 321–325.

Sabourin, R., Drouhard, J., & Wah, E. (1997a). Shape matrices as a mixed shape factor for off-line signature verification. *Proceedings of the International Conference on Document Analysis and Recognition*, Vol. 2, 661–665.

Sabourin, R., & Genest, G. (1994). An extended shadow code approach for off-line signature verification: Part I—Evaluation of the bar mask definition. *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, 450–460.

Sabourin, R., & Genest, G. (1995). An extended-shadow-code based approach for off-line signature verification: Part II—Evaluation of several multi-classifier combination strategies. *Proceedings of the Third IAPR Conference on Document Analysis and Recognition*, 197–201.

Sabourin, R., Genest, G., & Prêteux, F. (1996). Pattern spectrum as a local shape factor for off-line signature verification. *Proceedings of the 13th International Conference on Pattern Recognition*, C43–C48.

Sabourin, R., Genest, G., & Prêteux, F. (1997b). Off-line signature verification by local granulometric size distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(9), 976–988.

Sabourin, R., & Plamondon, R. (1986). Preprocessing of handwritten signatures form image gradient analysis. *Proceedings of the 8th International Conference on Pattern Recognition*, 576–579.

Sabourin, R., Plamondon, R., & Beaumier, L. (1994). Structural interpretation of handwritten signature images. *International Journal of Pattern Recognition and Artificial Intelligence*, *8*(3), 709–748.

Sansone, C., & Vento, M. (2000). Signature verification: Increasing performance by a multi-stage system. *Pattern Analysis and Applications*, *3*, 169–181.

Santos, C., Justino, E., Bortolozzi, F., & Sabourin, R.(2004). An off-line signature verification method based on the questioned document expert's approach and a neural network classifier. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 498–502.

Scholkopf, B., Platt, J., Taylor, J., Smola, A., & Williamson, R. (2001). Estimating the support of a high dimensional distribution. *Neural Computation*, *13*, 1443–1471.

Senol, C., & Yildirim, T. (2005). Signature verification using conic section function neural network. *Proceedings of the 20th International Symposium Computer and Information Sciences*, 524–532.

Srihari, S., Xu, A., & Kalera, M. (2004). Learning strategies and classification methods for off-line signature verification. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 161–166.

Tang, Y.Y., Tao, Y., & Lam, E.C.M. (2002). New method for feature extraction based on fractal behavior. *Pattern Recognition*, *35*(5), 1071–1081.

Tax, D. (2001). *One-class classification* [unpublished doctoral dissertation]. TU Delft.

Tortorella, F. (2005). A roc-based reject rule for dichotomizers. *Pattern Recognition Letters*, *26*, 167–180.

Vapnik, V. (Ed.). (1999). *The nature of statistical learning theory*. 2nd edition. New York: Springer-Verlag.

Vélez, J., Sánchez, A., & Moreno, A. (2003). Robust off-line signature verification using compression networks and positional cuttings.

Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, 627–636.

Wayman, J., Jain, A., Maltoni, D., & Maio, D. (Eds.). (2005). *Biometric systems, technology, design and performance evaluation*. New York: Springer.

Wen-Ming, Z., Shao-Fa, L., & Xian-Gui, Z. (2004). A hybrid scheme for off-line Chinese signature verification. *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, Vol. 2, 1402–1405.

Wilkinson, T., & Goodman, J. (1990). Slope histogram detection of forged handwritten signatures. *Proceedings of the SPIE—International Society for Optical Engineering*, 293–304.

Xiao, X., & Leedham, G. (2002). Signature verification using a modified Bayesian network. *Pattern Recognition*, *35*(5), 983–995.

Xuhua, Y., Furuhashi, T., Obata, K., & Uchikawa, Y. (1996). Selection of features for signature verification using the genetic algorithm. *Computers & Industrial Engineering*, *30*(4), 1037–1045.

Ye, X., Hou, W., & Feng, W. (2005). Off-line handwritten signature verification with inflections feature. *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Vol. 2, 787–792.

You, X., Fang, B., He, Z., & Tang, Y. (2005). Similarity measurement for off-line signature verification. *Lecture Notes in Computer Science: Advances in Intelligent Computing*, *3644*, 272–281.

ENDNOTE

[1] In real situations (e.g., banking transactions), a client is asked to supply from three to five signature samples at the time of subscription.

Chapter IV
An Automatic Off-Line Signature Verification and Forgery Detection System

Vamsi Krishna Madasu
Queensland University of Technology, Australia

Brian C. Lovell
NICTA Limited (Queensland Laboratory), and
University of Queensland, Australia

ABSTRACT

This chapter presents an off-line signature verification and forgery detection system based on fuzzy modeling. The various handwritten signature characteristics and features are first studied and encapsulated to devise a robust verification system. The verification of genuine signatures and detection of forgeries is achieved via angle features extracted using a grid method. The derived features are fuzzified by an exponential membership function, which is modified to include two structural parameters. The structural parameters are devised to take account of possible variations due to handwriting styles and to reflect other factors affecting the scripting of a signature. The efficacy of the proposed system is tested on a large database of signatures comprising more than 1,200 signature images obtained from 40 volunteers.

INTRODUCTION

A handwritten signature can be defined as the scripted name or legal mark of an individual, executed by hand for the purpose of authenticating writing in a permanent form. The acts of signing with a writing or marking instrument such as a pen or stylus is sealed on the paper. The scripted name or legal mark, while conventionally applied on paper, may also be accomplished using other devices that capture the signature process in digital format.

Hilton (1992) discusses what a signature is and how it is produced. He notes that the signature

has at least three attributes: form, movement, and variation. Since signatures are produced by moving a pen on a paper, movement perhaps is the most important aspect of a signature. Movement is produced by muscles of the fingers, hand, wrist, and, for some writers, arm; these muscles are controlled by nerve impulses. Once a person is used to signing his or her signature, these nerve impulses are controlled by the brain without any particular attention to detail.

The variations in handwritten signatures are quite immense, both within samples from the same individual and to an even larger degree across the population of individuals. The susceptibility of a signature to false imitation is clearly a function of the nature of the signature itself. In a broad sense, signatures can be classified as simple, cursive, or graphical based on their form and content, as shown in Figure 1.

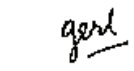
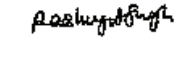
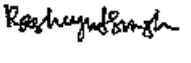
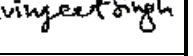
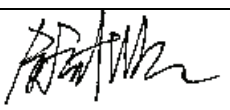
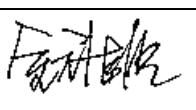
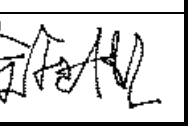
A simple signature is one where a person scripts his or her name in a stylish manner. In this type of signature, it is very easy to interpret all the characters in the name. Cursive signatures, on the other hand, are more complex. Though the signatures still contain all the individual characters within the name, they are, however, drafted in a cursive manner, usually in a single stroke. Lastly, the signatures are classified as graphical when they portray complex geometric patterns. It is very difficult to deduce the name of the person from a graphical signature, as it is more of a sketch of the name of the signer.

HANDWRITTEN SIGNATURES

It is a well-known fact that no two signatures, even if signed by the same person, are ever the same. However, if two signatures are *exactly* alike, then one of them is not a genuine signature but rather a copy of the other—either a machine copy such as one produced by a computer or photocopier, or a manually produced copy such as tracing. In addition, simulation must be taken into account, where an individual copies the signature of another using a genuine signature as a model. In these cases, the simulated writing usually exhibits an incorrect interpretation of inconspicuous characteristics of a genuine signature, which are quite hard to recognize by a nonexpert.

Osborn (1929), one of the earliest experts in the field of document examination, observed that variations in handwriting are themselves habitual. This is clearly seen in any collection of genuine signatures produced at different times and under a great variety of conditions. When carefully examined, these signatures show that running through them is a marked, unmistakable individuality even in the manner in which the signatures vary as compared with one another. He

Figure 1. Types of signatures

Type	Genuine	Skilled forgery	Unskilled forgery
Simple			
Cursive			
Graphical			

further notes that unusual conditions under which signatures are written may affect the signature. For example, hastily written, careless signatures, cannot always be used unless one has sample signatures that have been written under similar conditions. Furthermore, signatures written with a strange pen and in an unaccustomed place are likely to be different than the normal signatures of a person.

Locard (1936), another expert document analyst, surveyed graphometric techniques used for the authentication of questioned documents. Locard categorizes the characteristics of genuine handwritten signatures into two broad classes. The first class is related to those characteristics of genuine signatures that are quite difficult to imitate such as the rhythmic line of the signature consisting of the positional variation of the maximum coordinate of each character in the signature; the local variation in the width of the signature line, which is closely related to the dynamics of the writing process; and the variation in the aspect ratio of the complete signature followed by other local features such as the difference in orientation and relative position. The characteristics of the second class are the ones that are very easily perceived by a casual forger and are therefore easier to imitate. These are usually the general shape of the signature such as the signature's overall orientation and its position on the document.

Osborn (1929) states that the successful forging of a signature or simulating another person's writing by a forger involves not only copying the features of the genuine signature but also hiding his or her own personal handwriting characteristics.

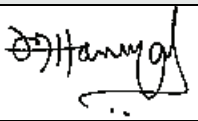
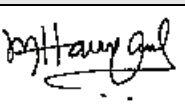
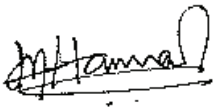
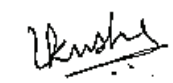
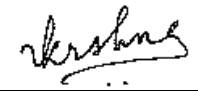
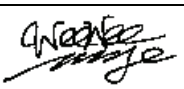
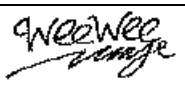
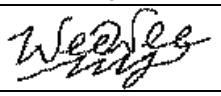
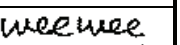
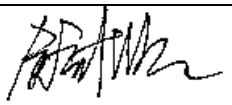
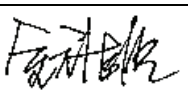
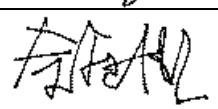
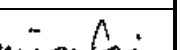
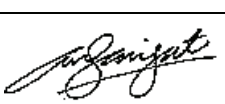
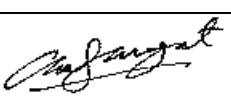
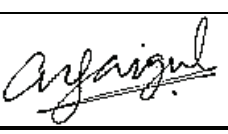
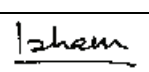
Forgeries in handwritten signatures have been categorized based on their characteristic features (Suen, Xu & Lam, 1999). Following are the three major types of forgeries:

- **Random forgery:** The signer uses the name of the victim in his or her own style to create a forgery known as simple forgery or random forgery. These forgeries represent almost 95% of all the fraudulent cases generally encountered, although they are very easy to detect even by the naked eye (Harrison, 1958).
- **Unskilled forgery:** The signer imitates the signature in his or her own style without any knowledge of the spelling and does not have any prior experience. The imitation is preceded by observing the signature closely for a while.
- **Skilled forgery:** Undoubtedly the most difficult of all forgeries is created by professional impostors or persons who have experience copying the signature. In order to achieve this, one could either trace or imitate the signature by hard way.

We now list some of the most important characteristics of genuine signatures and forgeries as outlined by several document examiners in the past. The understanding of these characteristics is important for determining those aspects or features of the signatures that are most important for automatic signature verification. Once the unique features have been selected, a knowledge base of all the feature values of the reference signature can be built so that when a test signature comes across the system, only the feature values are needed. This will eliminate the need for storage of all the signature images.

- **Enlargement of characters:** A forgery is usually larger than the original signature. This is due to the fact that a forger carefully observes the genuine signature before imitating it at the same time. The feedback mechanism in the brain of the forger is slower than the process taking place in the mind of the original writer, and consequently more time is spent drawing each letter. This makes a forgery larger than the original both in terms of the size of letters and the size of the entire signature. Similarly, the

Figure 2. Types of forgeries

Genuine	Skilled forgery	Unskilled forgery	Random forgery
			
			
			
			
			

complete signature can also be larger than the original, so enlargement does not just apply to individual letters.

- **Tendency of curves to become angles:** Curved letters are often observed in the forgery as being more angular. The forger takes care to obtain the correct letter shape by using a slower speed to produce the curve accurately. Ironically, this results in more angular letters as greater time elapses in the making of the curves. In the same way, angled letters in the original signature can become smooth curves.
- **Retouching:** Retouching results when the imitation has been done already but an addition is made to it at a later stage. Lines may appear to be thicker at these points, or there may be lines that do not follow the continual flow of the pen as in the original signature.
- **Poor line quality:** The ink reveals variation in light and shade; pressure and speed, with either more or less ink appearing on the page. It is more usual to find that the pressure used for the questioned signature is harder than that of the real signature. However, a lighter pressure can sometimes be detected. This may be due to a tremor caused by trembling of the hand; poor line quality, or writing too slowly.
- **Hesitation:** In the process of creating a forgery, the forger may pause to consult the genuine signature and then continue duplicating it. This can often create blobs (when the pen leaves an ink mark on the page), which may not be obvious.
- **Punctuation:** Full stops, dots on small letter "i" are in the wrong place, missing, or added.
- **Differing pressure:** Refers to the differences in pen pressure applied while signing. The pen pressure may be too heavy or too light, depending on the style of the forger. Pressure differences occur at different places from the genuine signature, as identical pen pressure is difficult to achieve. Most of the forgeries come out too dark or too light, as everyone has a different pen style, but it is also hard to vary pressure in the same way as a genuine signer.

- **Sudden endings:** Usually the original signature just trails off, whereas the forgery just stops. Sudden endings are a characteristic feature of a forgery, as it is very difficult to trail off in the same way as the genuine signature. Most often, it is simply easier to end more definitely rather than trailing off in a particular pattern.
- **Forger's characteristics:** The forger unconsciously reveals characteristics of his or her own handwriting when doing the forgery. Basic letter shapes, spacing, and position of letters in relation to base line are very similar to the forger's own. However, this is difficult to detect if the forger is unknown or, in other words, only possible when the forger's characteristics are definitely known.
- **Baseline error:** The forger, in an attempt to correctly imitate the size and shape of a signature, often neglects to ensure that the imaginary line that runs across the base of the signature is similar in the forged signature to the genuine signature. The baseline in a signature is not horizontal, and any notable variances in the baseline are almost sure signs of forgery.
- **Spacing:** Spacing may be larger or smaller between individual letters, between whole words, and between punctuation and letters that cannot be copied by tracing a signature.
- **Bad line quality:** Bad line quality is apparent by hesitant or shaky pen strokes and occurs when the forgery has been done too slowly.
- **Forming characters not appearing in signatures:** Unintelligible signatures are rationalized by a forger so that individual letters can be discerned in the forgery, whereas they are not apparent in the genuine signature. A characteristic of poor forgers is often caused by them knowing the name they are trying to forge and by unconsciously including letters that do not appear in the genuine signature. If the forger is unsure of the name, then incorrect letters may appear clearly in the forgery.

These characteristics demonstrate that human signature verification is far from trivial, but clearly, most of these points cannot be applied to computerized signature verification. The aim of this study is therefore to investigate the intrinsic properties of signatures that are repeated again and again so as to increase the reliability of uniquely identifying a person on the basis of his or her handwritten signature.

HANDWRITTEN SIGNATURE FEATURES

The handwritten signature is a behavioral biometric, which means that the biometric measurement is not based on any physiological characteristic of the individual, but on behavior that can change over time. The process of determining the legitimacy of a handwritten signature is termed *signature verification*. Since an individual's signature alters over time, the use of signature verification for authenticating sensitive financial transactions over a long period may lead to high error rates. Enrollment to a signature verification system requires the collection of an exclusive set of signature samples that are similar in nature so as to locate an adequate number of common characteristics. Inconsistent signatures lead to high false rejection rates and high enrollment failure rates for individuals who do not sign in a consistent way. However, the positive aspect of signature verification technology is that unlike other physiological biometrics such as face, fingerprint, or iris, if the signature biometrics of an individual are compromised, individuals can simply change their signatures.

In the recent past, many questions have been raised about the scientific basis of the expert opinion offered by forensic document examin-

ers. In order for forensic document examination to retain its credibility and legal acceptability as a science, there must be some statistically sound basis for the decision. In addition, such scientific information is also useful for the efficient development of automatic signature verification systems. This research is concerned with the automatic analysis of perceptible features in handwritten signatures to determine the features that distinguish a forgery from a genuine signature. In static or off-line signature verification systems, the signature image is characterized as a vector of elements, each one representative of the value of a feature. The careful selection of this feature vector is crucial for the success of any signature verification system.

Types of Features

Features extracted for off-line signature verification can be broadly divided into three main types (Fang, Leung, Tang, Tse, Kwok & Wong, 2003; Lee & Pan, 1992):

i. **Global features** depict or categorize the signature as a whole. These features are usually extracted from all the pixels that lie within the region circumscribing the signature image, such as the length, width or baseline of the signature, although global features are easily extractable and less sensitive to noise, as small distortions in isolated regions of the signature do not cause a major impact on the global feature vector. They are, however, dependent upon the overall position alignment and therefore highly susceptible to distortion and style variations.
ii. **Local features** represent a segment or limited region of the signature image, such as critical junctions and gradients. These features are generally derived from the distribution of pixels of a signature, such as local pixel density or slant. Local features are more sensitive to noise within the region under consideration but unaffected by other regions of the signature. Although they are computationally expensive, they are much more accurate than global features.
iii. **Geometric features** describe the characteristic geometry and topology of a signature, thereby preserving their global as well as their local properties. These features have a high tolerance to alterations and style variations, and they can also tolerate a certain degree of translation and rotation variations.

General Overview of Signature Features

Many types of features have been proposed for off-line signature verification systems with varying degrees of success. Since dynamic information is not available in static signatures, features can only be extracted from the geometric analysis of signatures. Some of the most widely used parameters are the signature image area, the signature height and width, the ratio between the signature height and its width, the ratio between middle zone width and signature width, global and local slant, the number of characteristics points (endpoints, cross-points, cusps, etc.), number of loops, the presence of the lower zone parts, and the number of elements in the signature (Ammar, 1991; Ammar, Yoshida, & Fukumura, 1990; Blatzakis & Papamarkos, 2001; Plamondon & Lorette, 1989).

The coefficients obtained from Fourier, Hadamard, and Wavelet transforms have also been used as parameters for off-line signature verification (Deng, Liao, Ho, & Tyan, 1999; Fadhel & Bhattacharyya, 1999; Murshed, Sabourin, & Bortolozzi, 1997; Nagel & Rosenfeld, 1977). Projection-based features include the number of vertical and horizontal projection peaks, and maximum vertical and horizontal projections (Ammar, 1991; Ammar et al., 1990). The main contour-based features are concerned with the use of parameters extracted from signature envelope

and outlines (Bajaj & Chaudhury, 1997; Cardot, Revenu, Victorri, & Revillet, 1994). Furthermore, texture-based features derived from the co-occurrence matrices of the signature image have also been considered by Blatzakis & Papamarkos (2001).

Many of the local features are of the same characters as the global ones. The difference is that they are applied either to the cells of a grid covering the signature or to the specific elements obtained after signature segmentation. They include slant of the element; density factor; length ratio of two consecutive parts; position relation between the global baseline and the local one; upper, central, and corner line features; and critical points (Ismail & Gad, 2000; Qi & Hunt, 1994; Quek & Zhou, 2002). In grid-based features, the signature image is divided into rectangular regions, and ink distribution in each region is evaluated (Blatzakis & Papamarkos, 2001; Drouhard, Sabourin, & Godbout, 1996; Sabourin, Genest, & Prêteux, 1997).

The features based on geometrical properties of a signature image are useful for the detection of random and simple forgeries, but they fail to recognize skilled forgeries, which are almost identical to the genuine signatures in terms of global shape and orientation. There have been attempts to extract dynamic information from static images. Parameters like stroke direction, length, width, and curvature variation are estimated with these techniques. Various levels of pressure features have also been extracted from the signature images (Ammar, 1991; Ammar et al., 1990; el-Yacoubi, Justino, Sabourin & Bortolozzi, 2000; Justino, Bortolozzi & Sabourin, 2001; Rigoll & Kosmala, 1998). It is assumed that they are connected with varying speeds at various parts of the signature (Quek & Zhou, 2002).

Qi and Hunt (1994) discuss a static signature verification based on global and local features of a signature image. The global features are height and width of a signature image, width of the signature image with blank spaces between horizontal elements removed, slant angle of the signature, vertical center of gravity of black pixels, maximum horizontal projection, area of black pixels, and baseline shift of the signature image. The local (or grid) features include the structural information of image elements; for example, angle of a corner, curvature of an arc, intersection between the line strokes, and number of pixels within each grid. These features are found to give good results compared to structural features.

From this discussion, it is understood that an appropriate combination of global and local features will produce more distinctive and more efficient features, because by localizing global features, the system will be able to avoid major shortcomings of both the approaches and at the same time benefit from their combined advantages.

DATA ACQUISITION

The first step in the design of a static signature verification system is data acquisition. Handwritten signatures are collected from various individuals, and some unique features are extracted from them to create a knowledge base for each individual. The features stored in the knowledge base are then learned by the system and used as a reference for comparing with those of the test signature in the recognition phase. A standard database of signature samples is thus needed for calculating the performance of the signature verification system and also for comparison with the results obtained using other techniques on the same database. Unfortunately, no such standard benchmark database exists in the field of signature verification due to the confidentiality and privacy issues associated with handwritten signatures.

The proposed signature verification system is trained and tested on a database consisting of a total of 1,200 handwritten signature images. Out of these, 600 are authentic signatures, and the other 600 are forgeries. These signatures are

obtained from 40 volunteers with each person contributing 15 signature samples, among which 10 are used for learning purposes and the rest for testing (see Table 1).

The selection of an optimum number of samples for training is a critical point during the construction of the signature databases. To investigate signatures, Osborn (1929) recommends that several genuine signatures should always be obtained, if possible, and five signatures always provide a more satisfactory basis for an opinion than just one signature, and 10 signatures being better than five. Hence, after much consideration, we have fixed the size of the training set to 10 samples for each person to reflect their signature variations optimally. In addition, the system is trained with only genuine signatures (i.e., none of the forgeries are used for training the system). Most of the signature verification systems trained with both genuine and forged signatures have been subject to errors. For example, the automatic off-line signature verification of Pender (1991) has a false acceptance rate (FAR) of 100% when trained with only genuine signatures. This means that it could not distinguish even a single forgery from genuine signatures when the system is not trained with the samples of forged signatures.

The signatures are handwritten on a white sheet of paper using any type of pen or pencil, and are scanned at a resolution of 300 dpi. A scanned image of the special sheet designed for collecting signatures is shown in Figure 3. The signatures are collected over a period of four weeks to account for the variations in signature style with time. The forgeries are also collected over the same time frame. The random forgeries are obtained by supplying only the names of the individuals to the casual forgers who never had any access to the actual genuine signatures. The unskilled forgeries, in turn, are obtained by providing sample genuine signatures to the forgers, who are then allowed to practice for a while before imitating them to create the forgeries. Each volunteer had to provide five imitations of any one of the genuine signatures, apart from his or her own signature. These samples constitute the set of unskilled forged signatures for the set of genuine signatures. We have then requisitioned the services of two expert forgers to provide five forgeries of each genuine signature in the test set so as to create the skilled forged samples of all the persons.

PREPROCESSING

The signature images scanned during the data acquisition phase are extracted and preprocessed in this module. The steps of preprocessing are briefly discussed in the following sections.

Binarization

Binarization is the first step in preprocessing of signature images. In this process, the input gray scale image is converted into a two-tone image format (i.e., black and white pixels, commonly represented by 1 and 0, respectively).

Table 1. Signature database

Types of Signatures	Training Set	Test Set	TOTAL
Genuine signatures	40 x 10	40 x 5	**600**
Skilled forgeries	-	40 x 5	**200**
Unskilled forgeries	-	40 x 5	**200**
Random forgeries	-	40 x 5	**200**

Slant Normalization

A practical signature verification system must be able to maintain high performance regardless of the size and slant of a given signature. For handwritten signatures, one of the major variations in writing styles is caused by slant, which is defined as the slope of the general writing trend with respect to the vertical line. It is important that the system be insensitive to slant; hence, the need for slant correction in the signature image.

The image matrix is divided into upper and lower halves. The centers of gravity of the lower and upper halves are computed and connected. The slope of the connecting line defines the slope β of the window (image matrix). The slant-corrected image is obtained by applying the following transformation to all black pixels with coordinate points x, y in the original image:

$$x' = (x - y) \times \tan(\beta - \beta_0), \quad y' = y \qquad (1)$$

where x' and y' are slant corrected coordinates and β_0 is a parameter specifying the default (normal) slant.

Slant correction needs to precede other preprocessing tasks (i.e., it is applied before smoothing), because smoothing tends to change the image topology, and the correction operation usually creates rough contours to the character.

Figure 3. Signature data acquisition

Skeletonization

A two-tone digitized image is defined by a matrix A, whose element $a_{i,j}$ is either 1 if character is present or 0 otherwise. Iterative transformations are applied on A to obtain a thinned image, which is of one pixel thickness. This process is termed "skeletonization," as the output image is a skeleton of the original image. The modified safe point thinning algorithm (SPTA) (Shih & Wong, 1995) is used in this work for the task of skeletonization.

Smoothing

Because of the excessive processing performed during the slant correction and thinning stage, we find that a signature often contains barbs and some redundant dark pixels that are not relevant in maintaining the connectivity of the image. Some such points have been identified and Boolean expressions developed to rid the image of points such as those described next. In the following depictions, "1" represents a dark pixel, "0" represents a white pixel, while "X" represents a don't-care condition. The central pixel is not relevant in maintaining the connectivity of the image, as path depicted by the arrows connects the remaining pixels. Points of similar configuration but different orientations (three other possible) are identified and removed (converted to white). Another set of points, termed extra corners, is also identified and deleted, as shown in Figure 4(b). Again, seeing the connectivity, we remove the central dark pixel. Three other orientations can be easily identified and the corresponding points deleted. Finally, the following set of points is identified in Figure 4(c). Here again, the central point is deleted.

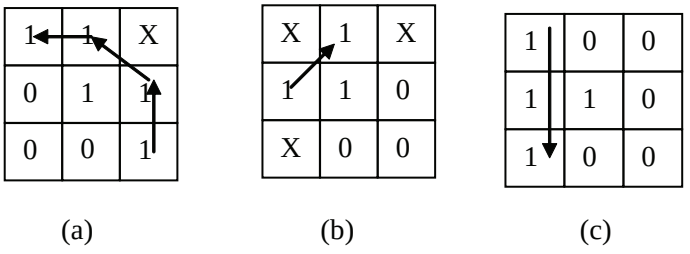
Endpoint Smoothing

Another novel approach has been devised to smoothing and removal of spurious tails of signature images that are distorted initially or during preprocessing. Two types of points in signatures are considered for this process:

- **Endpoints:** Dark points with only one dark neighbor out of eight closest neighbors.
- **Junction points:** Dark points with more than two dark neighbors out of eight closest neighbors.

The approach essentially involves identification of all endpoints in the preprocessed image. Starting with each endpoint, a path is traced until we either reach a junction point or exceed a heuristically determined path length. If we reach a junction point within the length specified, the path of dark pixels starting from that endpoint to the junction point is determined a spurious tail and deleted. If during traversal the path length is exceeded, we retain the branch and divert our search to the next endpoint until all points are covered. A bottleneck faced in the performance of this approach is the size of the heuristically

Figure 4. Smoothing using maintenance of connectivity



determined path length. If correctly chosen, it gives some spectacular results. However, incorrectly chosen lengths would either leave the image unaffected or may delete branches that should not be deleted otherwise.

Size Normalization

After the binarization process, there would be extra zeros on all four sides of the signature image, as zero padding is applied during binarization. To standardize the size of the signatures, extra rows and columns containing only zeros are removed from all four sides of the image. Normalization is thus the process of equating the size of all signature samples so as to extract features on the same footing. To achieve this, we use standard bilinear transformation, by which every input bitmap P of size $m \times n$ is transformed into a normalized bitmap Q of size $p \times q$. Both p and q are quadrilateral regions. All the signature images are standardized to a fixed window of size 120 x 60 pixels.

FEATURE EXTRACTION

The success of a pattern recognition system depends largely on the type of features extracted from the dataset. The chief objective of this process is to extract those features that will enable the system to correctly discriminate one class from the other. In this section, we will present our signature grid method, which has been devised for extracting innovative angle and distance features. The motivation behind the design of the grid is illustrated to prove its efficacy. Edge-based direction features adopted from handwriting recognition are also discussed.

Grid-Based Approaches

The structural information contained in a handwritten signature is obtained using a grid that is superimposed on the size-normalized signature image. The feature vector of each grid element includes the boundary code and the total number of pixels inside the grid. The boundary grid is a binary vector that is defined as:

$$b_i = \begin{cases} 1 & \text{pixel at i}^{\text{th}} \text{ position} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where b_i is the distance from the upper-left corner of each grid when moving counterclockwise on the boundary.

The length of the boundary code is equal to four times the side of each square grid. This boundary code is an incomplete, linear approximation to the structure within each grid when the size of the grid is relatively small, because there are multiple ways to linearly connect a given set of boundary locations within a grid. To alleviate this ambiguity, the intersection between the line stroke and the grid is thinned so each intersection is represented by only one code element b_i. The total grid feature is thus represented as:

$$v_g = (n, b_1, b_2, ..., b_{4l}) \quad b_i \in (0,1) \forall i \quad (3)$$

where n is the total number of pixels within each grid, and l is the side length (in pixels) of the squared grid.

Murshed, et al. (1997) performed a local analysis of the shape of a signature within a predefined search region called the identity grid, which is designed for each writer in the system. The signature image is centralized on the identity grid, which is divided into nine regions that are further divided into squares of size 16 x 16 pixels. The *xy*-coordinates of each 16 x 16-pixel square indicate a location of a graphical segment in the identity grid of a particular writer. Feature extraction is performed on each of the 16 x 16-pixel squares that contain a graphical segment by first calculating the center of the square and then extracting the graphical segment enclosed within the 32 x 32-pixel square.

A signature image of 512 x 128 pixels is centered on a grid of rectangular retinas, which are excited by local portions of the image (see Figure 5). Each retina has only a local perception of the entire scene, and granulometric size distributions are used for the definition of local shape descriptors in an attempt to characterize the amount of signal activity exciting each retina on the focus of the attention grid.

In Quek and Zhou (2002), the skeletonized image is divided into 96 rectangular segments and for each segment; area (the sum of foreground pixels) is calculated. The results are normalized so the lowest value (for the rectangle with the smallest number of black pixels) would be zero, and the highest value (for the rectangle with the highest number of black pixels) would be one. The resulting 96 values form the grid feature vector. A representation of a signature image and the corresponding grid feature vector is shown in Figure 6. A black rectangle indicates that for the corresponding area of the skeletonized image, there would be the maximum number of black pixels. On the contrary, a white rectangle indicates the smallest number of black pixels.

Signature Grid Method

The signature grid is defined as the region of interest within which the signature image is enclosed. The size of the grid, therefore, depends on the signature being enclosed within it. The grid is divided into eight partitions, which in turn are subdivided into 12 equal boxes.

The chief motivation behind the use of a signature grid is to divide the signature into local regions or boxes, which, over a set of all samples of a writer, form a fuzzy set. In this way, we are able to capture the global behavior through the local features, which forms an intelligent knowledge base of unique features for a particular individual. The other motivation for designing the grid is to reduce the area of focus to just the signature image.

Figure 5. A signature image centered on a grid of rectangular retinas

Figure 6. The grid feature vector for a signature

The preprocessed image is partitioned into eight portions using the equal horizontal density method. In this method, the binarized image is scanned horizontally from left to right and then from right to left, and the total number of dark pixels is obtained over the entire image. The pixels are clustered into eight regions such that an approximately equal number of dark pixels falls in each region. This process, known as the horizontal density approximation method, is illustrated in Figure 7.

From Figure 7, we note that the total number of points (dark pixels) is 48. If we divide the total pixels by four, we obtain 12 pixels per partition. Since the partition is done columnwise, obtaining exactly 12 points in each partition is difficult. Therefore, we take approximately 12 points in each partition using a two-way scanning approach. In this method, we scan the image from left to right until we reach the column where the number of points in a particular partition is 12 or more. We repeat the same procedure while scanning the image in a right-to-left direction. Then we partition the image in both directions: from left to right and right to left. Next, we take the average of two column numbers in each partition. Each partition is now resized to a fixed window of size 38 x 60 pixels and is thinned again. This partition is again subdivided into four rows and three columns, constituting 12 boxes. In total we have 96 boxes for a single signature. This approach is termed the signature grid method. The idea behind this method is to collect the local information contained in the box.

Signature Grid Features

Signature grid features are extracted using the signature grid method, which is based on the spatial division of the signature image. The signature is initially preprocessed and partitioned using the signature grid, as explained in the previous sections. The signature grid is divided into 96 (12 x 8) equal boxes superimposed on the signature image. The bottom left corner of each box is taken as the absolute origin (0,0), and distance and angle features are computed with reference to the origin of the box. The vector distance for *k*th pixel in *b*th box at location (*i*, *j*) is calculated as:

$$d_k^b = \sqrt{\left(i^2 + j^2\right)} \qquad (4)$$

These vector distances constitute a set of features based on distance. Similarly, for each *k*th black pixel in a box at location (*i*, *j*), the corresponding angle is computed in a similar manner.

By dividing the sum of distances of all black pixels (having value '1') present in a box with their total number, a normalized vector distance, γ_b, for each box is obtained as:

$$\gamma_b = \frac{1}{n_b} \sum_{k=1}^{n_b} d_k^b \qquad (5)$$

where, n_b is number of pixels in b^{th} box.

Figure 7. Partition using horizontal density approximation method

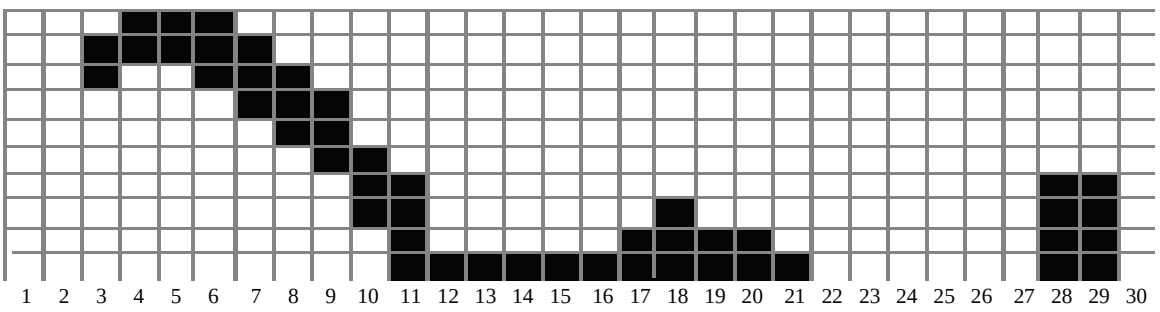
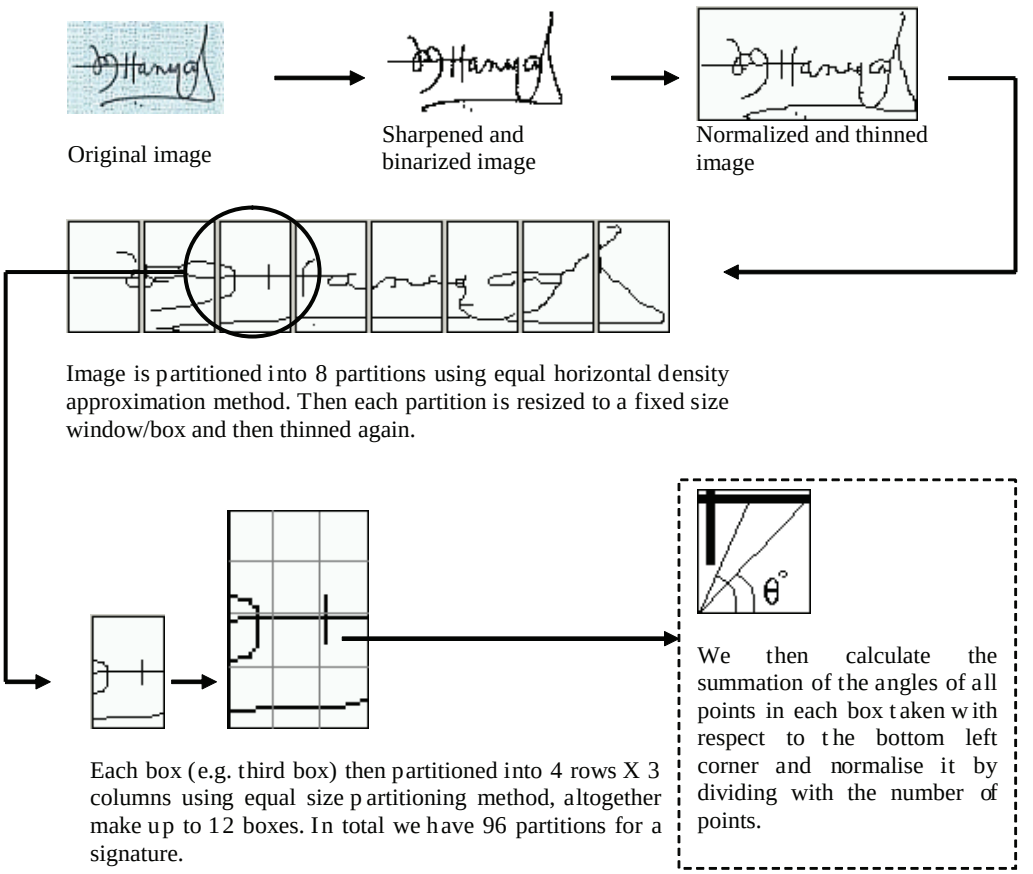


Figure 8. Preprocessing and feature extraction



Then the sum of all angles in a box *b* is divided by the number of 'l' pixels present in that box to yield a normalized angle γ_b :

$$\gamma_b = \frac{1}{n_b}\sum_{k=1}^{n_b}\theta_k^b \qquad (6)$$

where, n_b is number of pixels in *b*th box.

The angle and distance features obtained from all 96 boxes constitute the complete feature set of a particular signature sample. For the present problem of signature verification and forgery detection, we have experimented with both distance and angle distributions. However, it is found that the angle distribution is better than distance distribution due to its nonlinearity (see Figure 9).

Hence, the choice fell on extracting angle information from the boxes. We now discuss the computational aspects. Table 2 gives the angle features of one of the signatures used for training. The eight rows stand for the eight partitions of the signature, while the columns symbolize the further divisions within each partition.

VERIFICATION SYSTEM

Automatic verification of handwritten signatures on bank checks is integral to the success of a bank check processing and authentication system. The focus of this chapter is hence on the development of an automatic system for verification and forgery detection of handwritten signatures extracted from paper documents. The features considered in the recognition system are angle and distance features.

Figure 9. Distance and angle feature distributions for signatures

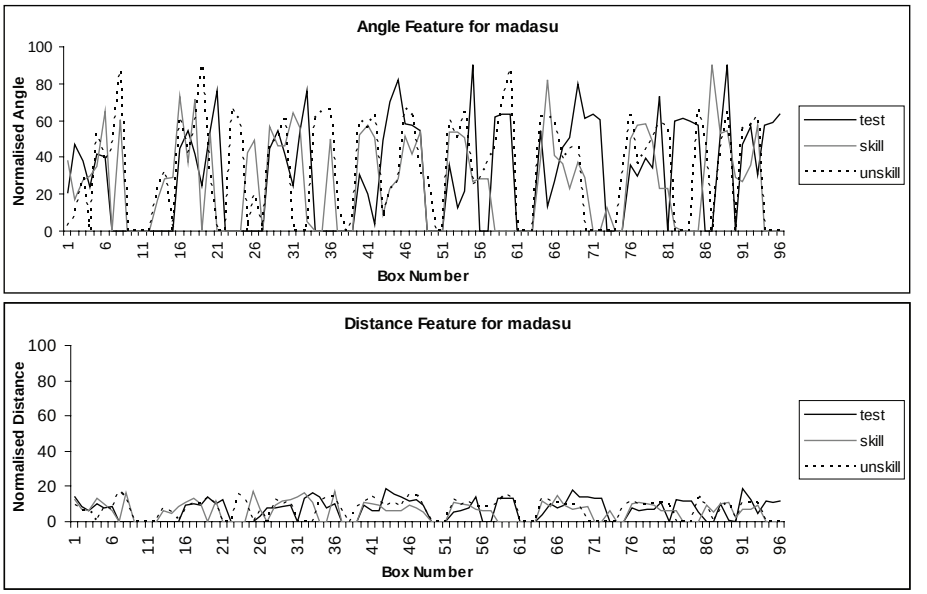


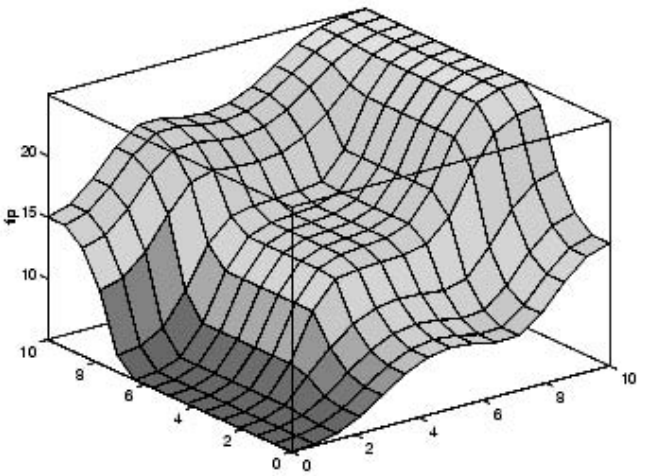
Table 2. Angle features of one of the signatures used for training

Partition Cluster	1	2	3	4	5	6	7	8	9	10	11	12
1	21.1	46.8	38.1	23.9	41.6	40.1	0	0	0	0	0	0
2	0	0	0	44.8	54.8	41.7	24.7	54.5	76.1	0	0	0
3	0	0	40.1	46.9	39.8	0	72.4	60.2	21.0	46.1	56.4	0
4	0	0	30.5	20.5	3.7	49.4	70.2	81.9	58.0	57.5	54.2	0
5	0	0	35.6	12.7	21.3	90	0	0	61.6	63.3	63.3	0
6	0	0	54.7	13.6	26.9	45.6	50.6	79.9	60.9	63.3	60.5	0
7	0	0	36.1	30.1	39.4	33.9	73.4	0	59.6	61.3	59.6	57.4
8	0	0	52.1	90	0	47.1	56.6	30.7	57.4	59.1	63.3	0

The verification system is based on the Takagi-Sugeno (TS) fuzzy model (Takagi & Sugeno, 1985). A Takagi-Sugeno fuzzy inference system is well suited to the task of smoothly interpolating the linear gains that would be applied across the input space; it is a natural and efficient gain scheduler. It is also suitable for modeling nonlinear systems by interpolating multiple linear models. A graphical representation of a TS model is illustrated in Figure 10.

Signature verification and forgery detection are carried out using angle features extracted from the signature grid. Each feature corresponds to a fuzzy set over all the samples of the training set. The features are fuzzified by an exponential membership function involved in the TS model, which is modified to include structural parameters to account for variations in signing styles. The membership functions constitute weights in the TS model. The optimization of the output of the TS model with respect to the structural parameters

Figure 10. Takagi-Sugeno fuzzy model



yields the solution for the parameters. The simplified form of the TS model is derived by fixing the coefficients of consequent parts of the rules made up of all input features and also by considering a single rule for all input features.

System Design

The proposed system includes both signature verification and forgery detection modules. The difference between them is that verification is based on inherent characteristics of a signer, whereas detection is based on specification of a limit, which exceeds the inherent variation in the genuine signatures of a signer. Various categories of forgery arise, depending on the limit of variation allowed over the inherent variation. The various phases of the verification and detection are discussed in the following sections.

Model Formulation

Since the main thrust here is to establish the genuineness of the signature, thereby detecting the forgeries, we have employed the TS fuzzy model for this purpose. In this study, we consider each feature as forming a fuzzy set over large samples, because the same feature exhibits variation in different samples giving rise to a fuzzy set. So our attempt is to model the uncertainty through a fuzzy model such as the TS model. The overall system organization is depicted in Figure 11.

The First Formulation

Let x_k be the k^{th} feature in a fuzzy set A_k, so the k^{th} IF THEN fuzzy rule in the TS model has the following form:

$$\textit{Rule } k\text{: IF } x_k \text{ is } A_k \quad \text{THEN } y_k = c_{k0} + c_{k1}x_k \qquad (7)$$

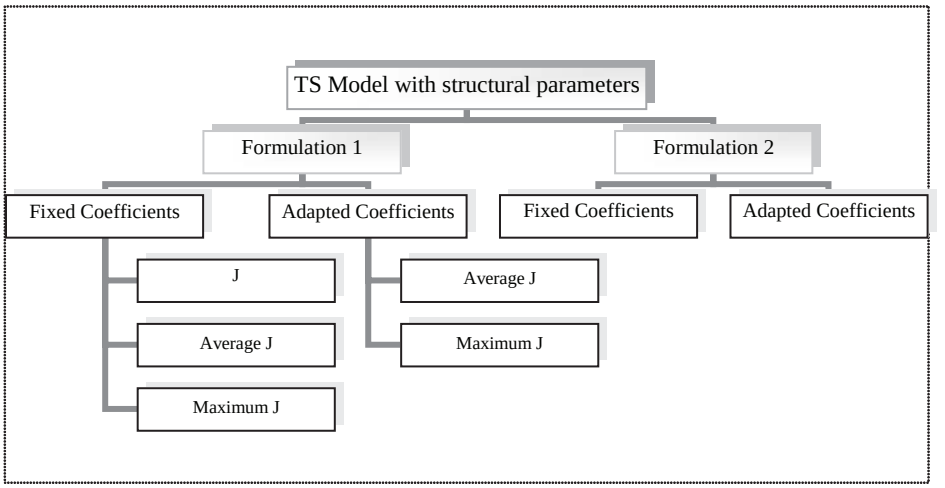
Each feature will have a rule, so we have as many rules as the number of features. The fuzzy set A_k is represented by an exponential membership function (MF) that includes two structural parameters, s_k and t_k. This membership function is expressed as:

$$\mu_k(x_k) = \exp - \left[\frac{(1-s_k) + s_k^2 \left| x_k - \overline{x_k} \right|}{(1+t_k) + t_k^2 \sigma_k^2} \right] \qquad (8)$$

where $\overline{x_k}$ is the mean, and σ_k^2 is the variance of k^{th} fuzzy set.

The structural parameters are included in the TS model so as to track the intraclass variations in

Figure 11. System organization



the various samples of signatures obtained from the same individual. A special condition of this function occurs when s_k = 1 and t_k = -1. In that case, the MF becomes devoid of the structural parameters and is solely dependent on means and variances. The significance of this condition is that the reference signature and the signatures under investigation will have the same statistics. This choice is guided by the consideration of no role for parameters if the signatures of a person don't change. The justification for the modified MF is twofold: (i) easy-to-track variations in means and variances and (ii) no need for any sophisticated learning technique.

The strength of the rule in Equation 7 is obtained as:

$$w_k = \mu_k(x_k) \tag{9}$$

The output is expressed as:

$$Y = \sum_{k=1}^{L} w_k y_k \tag{10}$$

where L is the number of rules.

We define the performance function as:

$$J = (Y_r - Y)^2 \tag{11}$$

where, Y and Y_r denote the output of the fuzzy model and of the real system, respectively.

Since the output of a verification system Y_r is not available, it can be safely assumed to be unity. In order to learn the parameters involved in the membership function (i.e., s_k and t_k) and the consequent parameters c_{k0} and c_{k1}, Equation 11 is partially differentiated with respect to each of these parameters. Accordingly, we have:

$$\frac{\partial J}{\partial c_{k1}} = \frac{\partial J}{\partial Y} \cdot \frac{\partial Y}{\partial y_k} \cdot \frac{\partial y_k}{\partial c_{k1}} = 2\left(Y - Y_r\right) w_k x_k \tag{12}$$

$$\frac{\partial J}{\partial c_{k0}} = \frac{\partial J}{\partial Y} \cdot \frac{\partial Y}{\partial y_k} \cdot \frac{\partial y_k}{\partial c_{k0}} = 2\left[Y - Y_r\right] w_k = 2\delta\, w_k \tag{13}$$

$$\frac{\partial J}{\partial s_k} = \frac{\partial J}{\partial Y}\cdot\frac{\partial Y}{\partial w_k}\cdot\frac{\partial w_k}{\partial t_k} = 2(Y-Y_r)\cdot y_k \cdot \frac{\mu_k\left\{1-2s_k\left|x_k-\overline{x_k}\right|\right\}}{\left\{(1+t_k)+t_k^2\sigma_k^2\right\}} = 2\delta\, y_k\mu_k\left[\left\{1-2s_k\left|x_k-\overline{x_k}\right|\right\}\Big/T\right] \quad (14)$$

$$\frac{\partial J}{\partial t_k} = \frac{\partial J}{\partial Y}\cdot\frac{\partial Y}{\partial w_k}\cdot\frac{\partial w_k}{\partial t_k} = 2\left(Y-Y_r\right)y_k\mu_k\frac{\left\{(1-s_k)+s_k^2\left|x_k-\overline{x_k}\right|\right\}\left\{1+2t_k\sigma_k^2\right\}}{\left\{(1+t_k)+t_k^2\sigma_k^2\right\}^2} = 2\delta\, y_k\mu_k\left\{(1-s_k)+s_k^2\left|x_k-\overline{x_k}\right|\right\}\left\{1+2t_k\sigma_k^2\right\}\Big/T^2 \quad (15)$$

where, $\delta = Y - Y_r$, $T = (1+t_k)+t_k^2\sigma_k^2$, and $k = 1,..,L$ denotes the rule number.

The gradient descent learning technique is applied to learn the parameters as follows:

$$c_{ki}^{new} = c_{ki}^{old} - \in_1 \frac{\partial J}{\partial c_{ki}}, i = 0,1 \quad (16)$$

$$s_k^{new} = s_k^{old} - \in_2 \frac{\partial J}{\partial s_k} \quad (17)$$

$$t_k^{new} = t_k^{old} - \in_3 \frac{\partial J}{\partial t_k} \quad (18)$$

where $\in_1, \in_2, \in_3$, are the learning coefficients such that $\in_1, \in_2$ *and* $\in_3 \in \Re^+$.

We can make use of global learning when we have large datasets, say *M*. This is known as a batch learning scheme in which change in any parameter is governed by the following equation:

$$\Delta w(q) = \sum_{j=1}^{M}\Delta_j w(q) + \alpha_m \Delta w(q-1) - \gamma w(q) \quad (19)$$

The parametric update equation is:

$$w(q+1) = w(q) + \Delta w(q) \quad (20)$$

where, *w* in (21) may stand for any of the parameters c_{ki}, s_k, t_k, q is the q^{th} epoch, α_m is a momentum coefficient in the limits $0 \le \alpha_m < 1$ (typically α_m = 0.9), and γ is a decay factor (typically in the range of 10^{-3} to 10^{-6}).

We can obtain initial $\Delta w(q)$ from Equations 16–18 by computing the partial derivatives of *J*. We will now show that the recognition approach explained previously is a special case of TS model. For this, assume c_{k0} = 1/*L* and c_{k1} = 0 so that y_k = 1/*L* in Equation 7. Substituting this in Equation 10 yields:

$$Y = \frac{1}{L}\sum_{i=1}^{L}\mu_i \quad (21)$$

In Equation 21, *Y* is given by the average of the membership functions (MFs), and we will now prove that this average MF is a special case of the TS model. The recursive equations, Equations 16–18, are iterated until the summation of δ for all feature values is small enough. The initial values of the structural parameters are obtained from:

$$\frac{\partial J}{\partial s_k} = 0 \Rightarrow 1-2s_k\left|x_k-\overline{x_k}\right| = 0 \Rightarrow s_k = \frac{1}{2\left|x_k-\overline{x}_k\right|} \quad (22)$$

$$\frac{\partial J}{\partial t_k} = 0 \Rightarrow 1+2t_k\sigma_k^2 = 0 \Rightarrow t_k = -\frac{1}{2\sigma_k^2} \quad (23)$$

Note that these initial values do not yield satisfactory results. We have to fine tune these values to obtain an efficient set of values.

The Second Formulation

Alternatively, it is possible to use only a single rule for all input features. The corresponding TS model will have the fuzzy rule of the form:

Rule: IF x_1 is A_1, x_2 is A_2,..., x_n is A_n

$$\text{THEN } y = c_0 + \sum_{i=1}^{n} c_i x_i \tag{24}$$

The performance function now becomes:

$$J = \{Y_r - wy\}^2 \text{ with } w = \Pi_{j=1}^{n} \mu_j \tag{25}$$

The derivatives of J with respect to c_0, c_i, s_i, t_i are given by the following equations:

$$\frac{\partial J}{\partial c_0} = -2\{Y_r - wy\}\Pi_{j=1}^{n}\mu_j \tag{26}$$

$$\frac{\partial J}{\partial c_i} = -2\{Y_r - wy\}x_i\Pi_{j=1}^{n}\mu_j \tag{27}$$

$$\frac{\partial J}{\partial s_i} = -2\{Y_r - wy\}y\mu_i \frac{\{1 - 2s_i(x_i - \overline{x_i})\}}{\{(1+t_i) + t_i^2\sigma_i^2\}} \tag{28}$$

$$\frac{\partial J}{\partial t_i} = -2\{Y_r - wy\}y\mu_i \frac{\{(1-s_i) + s_i^2(x_i - \overline{x_i})\}\{1 + 2t_i\sigma_i^2\}}{\{(1+t_i) + t_i^2\sigma_i^2\}^2} \tag{29}$$

The parameters can be found by the gradient descent technique. We will now derive the simplified version of the performance function. For this, assume $c_0 = 0$ and $\forall c_i = 0$ in Equation 23. This results in the equation:

$$y = w = \Pi_{j=1}^{n}\mu_j \tag{30}$$

From Equations 21 and 30, we observe that if we have a rule for each input feature, the simplified performance function is given by the average MF, whereas if all input features are linked by a single rule, the simplified performance function corresponds to the multiplication of all MFs. We find that Equation 30 is more stringent than Equation 21 as it requires that all membership values must be nonzero. The recognition using Equation 21 is bound to be better in view of a large number of rules and parameters involved. So our implementation follows this recognition strategy, but by making subtle changes to suit real-world problems.

Implementation

The proposed system is applied on the signature database described in detail in Section 4. For implementation, we will consider two cases: in the first case, we use the simplified TS model in which the coefficients of the THEN part (consequent) are fixed, whereas in the second case, we adapt the coefficients.

Case 1: TS Model with Consequent Coefficients Fixed

In view of Equation 21 and taking $Y_r = 1$, Equation 11 becomes:

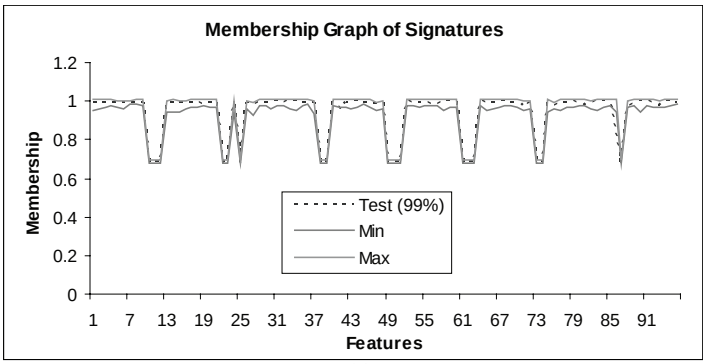
$$J = (1 - \frac{1}{L}\sum_{i=1}^{L}\mu_i)^2 \tag{31}$$

With this performance index, we compute $\frac{\partial J}{\partial s_i}$ and $\frac{\partial J}{\partial t_i}$ in order to update the structural parameters s_i and t_i; $i = 1,..,96$. Using these values, we compute the membership functions for all the features. This process is repeated for all the training samples of a person. Here, we have devised an innovative approach for the classification of all signatures (i.e., test signatures and random, skilled, and unskilled forgeries) of a person.

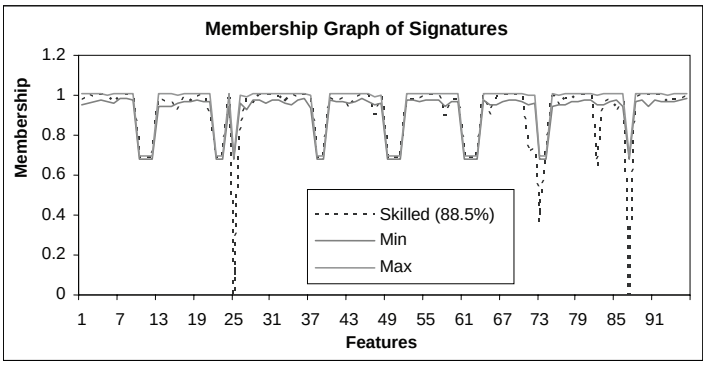
Innovative approach using variation in MF

In order to know the extent of variation in the genuine signatures, we determine the maximum and minimum membership functions for each feature over all signatures in the training set. The difference between these two gives the inherent variation in the signatures of a person. We add some tolerance to the maximum and delete the same from the minimum so as to increase the

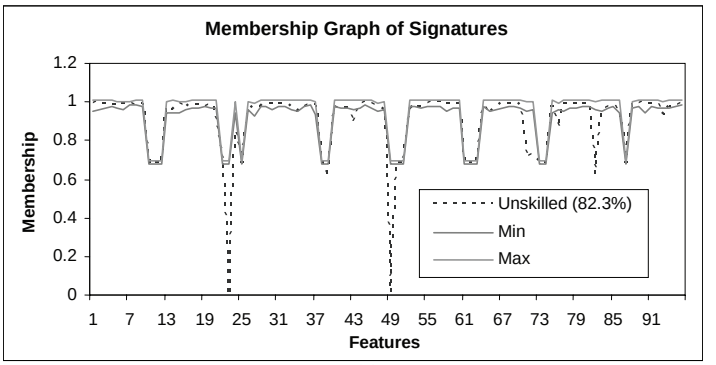
Figure 12. Membership graph of (a) genuine, (b) skilled forgery, and (c) unskilled forgery



(a)



(b)



(c)

range of variation in the various signatures. This tolerance is meant for possible increase in the inherent variation over a time.

We now use the inherent variation to judge the test signatures. We will also explain its utility in the testing phase. For a particular feature, if the membership value lies within the range of variation, which is given by the difference of minimum and maximum thresholds, it is counted as "true." The total number of "true" cases for a particular signature is divided by the total number of features (i.e., 96) to get the percentage. For example, in Figure12a, the test signature has 99% of its features lying well within the threshold, as can be seen from the membership function (i.e., 95 out of 96 features are within the range of inherent

Table 3. Initial values of the structural and learning parameters

Parameter		Simplified TS Model Initial Values		TS Model Initial Values
s		0.1		1
t		1.4		2
c_0		1/96		1/96
c_1		0		0
ε_1		-		0.00000001
ε_2		0.01		0.01
ε_3		0.01		0.01
Precision		**0.01**		**0.01**

Table 4. Results using formulation 1 with fixed consequent coefficients

Signature Type	Total	Accepted	Rejected
(a) *J*			
Genuine	200	200 (100%)	0 (0%)
Skilled forgery	200	0 (0%)	200 (100%)
Unskilled forgery	200	0 (0%)	200 (100%)
Random forgery	200	0 (0%)	200 (100%)
(b) *Average J*			
Genuine	200	184 (92%)	16 (8%)
Skilled forgery	200	44 (22%)	156 (78%)
Unskilled forgery	200	8 (4%)	192 (96%)
Random forgery	200	0 (0%)	200 (100%)
(c) *Maximum J*			
Genuine	200	200 (100%)	0 (0%)
Skilled forgery	200	42 (21%)	158 (79%)
Unskilled forgery	200	6 (3%)	194 (97%)
Random forgery	200	0 (0%)	200 (100%)

variation). The skill-forged and unskilled-forged signatures have corresponding figures of 88.5% (Figure12b) and 82.3% (Figure12c), respectively. We set the minimum limit or acceptable percentage for genuine signature at 91%, referring to the output result of signature of one particular individual. Signatures that have a percentage less than 91% are treated as forged signatures. Table 3 gives the initial values of learning and structure parameters.

Intuitive Approaches Taking the Average and Max of J

Next, we used the performance index given by Equation 11 and its derivatives to adapt the structural parameters during the training phase. These are used to determine the extent of inherent variation in terms of *J* in the training phase. We have tried two intuitive approaches. In the first case, we have taken average *J*, and in the second case, we have taken maximum *J*, both serving as thresholds. The samples in the testing phase are judged by comparing their *J* values against

the thresholds. Table 4(a) summarizes the results of forgery detection using this innovative approach. Tables 4(b) and 4(c) provide the results of forgery detection using the average *J* and max of *J*, respectively. Comparing these results, we find that the innovative approach yields the best performance.

Case 2. TS Model with Adaptive Consequent Coefficients

Next, we used the performance index given in Equation 11 and its derivatives to adapt both consequent coefficients and the structural parameters during the training phase. As already mentioned, we used both the average and maximum values of *J* for the detection of forgeries. Tables 5(a) and 5(b) show results using these two thresholds.

The trained features and structural parameters for one signature set are enumerated in Table 7. For the second formulation involving a single TS rule, the results with consequents fixed are shown in Table 6(a), and results with coefficients adapted are shown in Table 6(b). As compared to the results of the first formulation, these results are not promising for the reasons cited previously. Here, we have not made use of the average and maximum values of *J*.

Figure 13 shows the sample output of the program for two authors. All the experiments have been carried on a Pentium III, 1.1GHz Celeron processor having 256MB SDRAM with a Windows XP operating system. With this configuration, the system takes about 19 seconds to train 10 signature images, while it takes about two seconds to test one signature. Surprisingly, only a single

Table 5. Results using formulation 1 with coefficients adapted

Signature Type	Total	Accepted	Rejected
(a) Average J			
Genuine	200	172 (86.0%)	28 (14%)
Skilled forgery	200	47 (23.5%)	153 (76.5%)
Unskilled forgery	200	8 (4%)	192 (96.0%)
Random forgery	200	0 (0%)	200 (100%)
(b) Maximum J			
Genuine	200	200 (100.0%)	0 (0%)
Skilled forgery	200	44 (22%)	156 (78%)
Unskilled forgery	200	6 (3%)	194 (97.0%)
Random forgery	200	0 (0%)	200 (100%)

Table 6. Results using formulation 2

Signature Type	Total	Accepted	Rejected
(a) *Fixed Consequent coefficients*			
Genuine	200	125 (62.5%)	75 (37.5%)
Skilled forgery	200	68 (34%)	132 (66%)
Unskilled forgery	200	51 (25.5%)	149 (74.5)
Random forgery	200	50 (25%)	150 (75%)
(b) *Adapted Consequent coefficients*			
Genuine	200	107 (53.5%)	93 (46.5%)
Skilled forgery	200	84 (42%)	116 (58%)
Unskilled forgery	200	68 (34%)	132 (66%)
Random forgery	200	45 (22.5%)	155 (77.5%)

Table 7. Trained features and structural parameters of one particular signature set

Partition	Cluster											
	1	3	3	4	5	6	7	8	9	10	11	12
	(a) ***Features of a trained signature sample***											
1	21.1	46.8	38.1	23.9	41.6	40.1	0	0	0	0	0	0
2	0	0	0	44.8	54.8	41.7	24.7	54.5	76.1	0	0	0
3	0	0	40.1	46.9	39.8	0	72.4	60.2	21.0	46.1	56.4	0
4	0	0	30.5	20.5	3.7	49.4	70.2	81.9	58.0	57.5	54.2	0
5	0	0	35.6	12.7	21.3	90	0	0	61.6	63.3	63.3	0
6	0	0	54.7	13.6	26.9	45.6	50.6	79.9	60.9	63.3	60.5	0
7	0	0	36.1	30.1	39.4	33.9	73.4	0	59.6	61.3	59.6	57.4
8	0	0	52.1	90	0	47.1	56.6	30.7	57.4	59.1	63.3	0
	(b) ***Parameter 'S'***											
1	0.100523	0.100618	0.100621	0.100581	0.100226	0.100256	0.100698	0.10075	0.10076	0.09998	0.09998	0.09998
2	0.100412	0.10046	0.100436	0.10023	0.100368	0.100492	0.100734	0.100624	0.100479	0.09998	0.09998	0.100392
3	0.09998	0.100263	0.10047	0.100406	0.100626	0.100601	0.10074	0.100735	0.100683	0.100607	0.100696	0.100721
4	0.100365	0.09998	0.09998	0.100569	0.100606	0.100577	0.100444	0.100705	0.100749	0.100417	0.100089	0.100256
5	0.09998	0.09998	0.09998	0.100607	0.100532	0.100555	0.1007	0.100719	0.100721	0.100423	0.10069	0.100703
6	0.09998	0.09998	0.09998	0.10063	0.100585	0.100408	0.100603	0.100619	0.100588	0.10068	0.100156	0.100155
7	0.09998	0.09998	0.100517	0.100134	0.100374	0.100572	0.100638	0.100705	0.100681	0.100157	0.100641	0.100699
8	0.100473	0.100544	0.09998	0.100307	0.100693	0.100689	0.100683	0.100505	0.100149	0.100703	0.100735	0.100764
	(c) ***Parameter 'T'***											
1	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.39953	1.39908	1.39911
2	1.38695	1.399945	1.391186	1.4	1.4	1.4	1.4	1.4	1.4	1.39934	1.39932	1.39989
3	1.38929	1.399922	1.399964	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
4	1.39922	1.39908	1.39972	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
5	1.39461	1.39921	1.39989	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
6	1.39558	1.39567	1.39945	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
7	1.39115	1.39229	1.39939	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
8	1.4	1.399631	1.39906	1.4	1.4	1.399985	1.4	1.4	1.4	1.4	1.4	1.4

Figure 13. Sample outputs from the program for the signer (a) Weewee (b) Mionfai

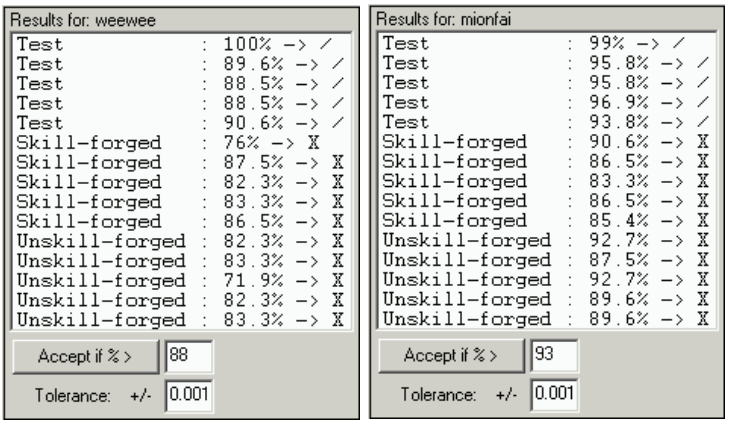
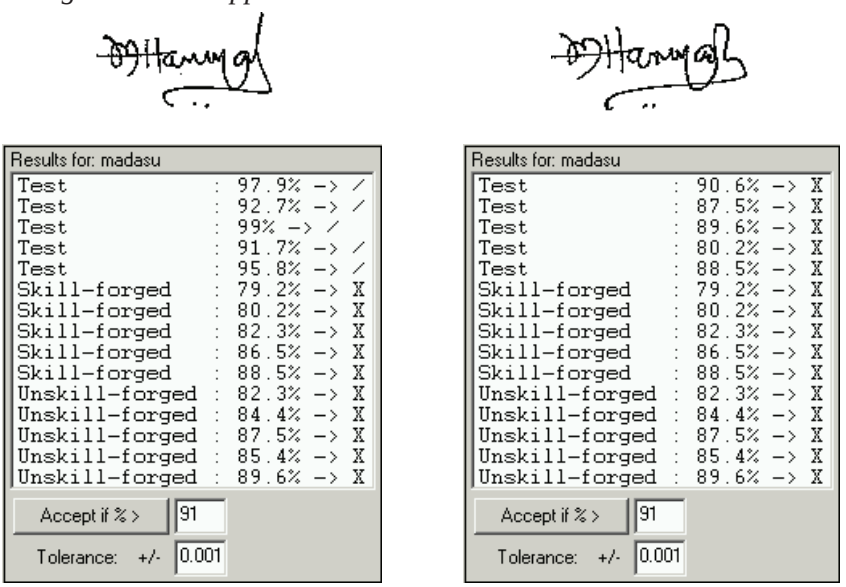


Figure 14. Sample outputs from system showing the old signatures are treated as forged when the thresholds of the current signatures are applied



(a) Current signature of the signer

(b) Old signature of the same signer

iteration is required to achieve the convergence as the learning parameters and initial structure parameters have been selected optimally.

Experiments

In this section, we will describe a few experiments we conducted on the signature database using the grid method to test its robustness with respect to scale invariance, continuity, and stability.

In the first experiment, to test the efficacy of the proposed signature verification system, we have subjected it to a typical assessment. The current signatures of a particular signer who had changed his signature a few years ago have been used to train the parameters and thresholds for testing the old signatures. As the old signatures have a slight change at their ends, the verification system declared the old signatures as forged. The sample outputs for the typical case are shown in

Figure 15. Signature samples of an individual of varying sizes and stroke widths

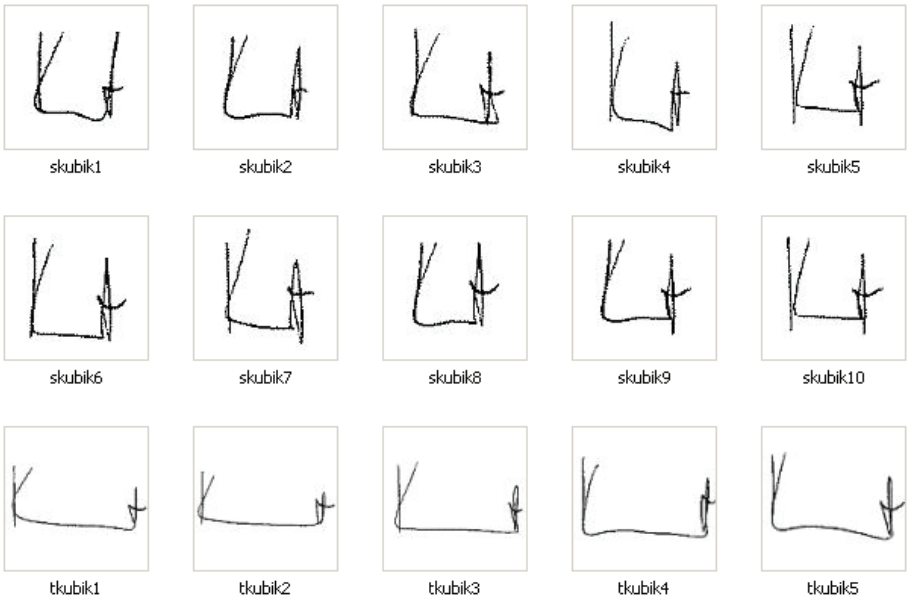


Figure 14. This test demonstrates the capability of the system in detecting even the slightest changes in the signature samples, even if they are acquired from the same person who has changed his or her signature style. This is because of the change in the *s* and *t* parameters, which are crucial to the success of the system. It is therefore important to adapt these parameters according to the new reference signatures, as the previous values were computed using the old reference signatures.

In the second experiment, we took signatures of various sizes of the same person to test the effectiveness of the signature grid that encompasses the signature samples (see Figure 15). We have also analyzed the effect of an elongated signature grid whenever the width of the signature strokes extends beyond the normal signing style. This can easily happen as some of the extreme features of the signature may alter from time to time, depending on the width of the signature strokes. For this purpose, we took signature samples of varying stroke widths and used them as test signatures for computing the recognition accuracies. It has been observed that even after size normalization, the inherent features of a person's signature are preserved, as the recognition rates come out to be near perfect. This particular characteristic of the system is vital for the success of a commercial signature verification system because of the tendency of signers to sign their names with varying stroke widths, depending on the availability of space for putting the signature.

CONCLUSION

In this chapter, an off-line signature verification and forgery system based on additive fuzzy modeling is presented. The handwritten signature images are preprocessed and angle features extracted from them via a novel grid method. These features are then modeled using the Takagi-Sugeno fuzzy model, which involves two structural parameters in its exponential membership function. Each angle feature yields a fuzzy set when its values are gathered from all samples because of the

variations in handwritten signatures. Two cases are considered. In the first case, the coefficients of the consequent part of the rule are fixed so as to yield a simple form of the TS model, and in the second case, the coefficients are adapted. In this formulation, each rule is constituted by a single feature. In the second formulation, we consider only one rule encompassing all the features. Here again, we have derived two models, depending on whether coefficients of the consequent part are fixed or adapted. However, this formulation is not implemented, as the membership values are found to be very small for some fuzzy sets. The efficacy of this system has been tested on a large database of signatures. The verification system achieved 100% success in verifying genuine signatures and detecting all types of forgeries (i.e., random, unskilled, and skilled) on a signature database consisting of 1,200 signature samples. A simple form of the TS model in the first formulation is found to be better than that with coefficients adapted. We have also demonstrated the effectiveness of the intuitive approach for signature verification over other approaches using the performance index.

REFERENCES

Ammar, M. (1991). Progress in verification of skilfully simulated handwritten signatures. *International Journal of Pattern Recognition and Artificial Intelligence*, *5*(1-2), 337–351.

Ammar M., Yoshida, Y., &. Fukumura, T. (1990). Structural description and classification of signature images. *Pattern Recognition*, *23*, 697–710.

Bajaj, R., & Chaudhury, S. (1997). Signature verification system using multiple neural classifiers. *Pattern Recognition*, *30*(1), 1–7.

Blatzakis, H., & Papamarkos, N. (2001). A new signature verification technique based on a two-stage neural network classifier. *Engineering Applications of Artificial Intelligence*, *14*, 95–103.

Cardot, H., Revenu, M., Victorri, B., & Revillet, M.J. (1994). A static signature verification system based on a cooperating neural networks architecture. *International Journal of Pattern Recognition and Artificial Intelligence*, *8*(3), 679–692.

Deng, P.S., Liao, H.-Y.M., Ho, C.W., & Tyan, H.-R. (1999). Wavelet based off-line handwritten signature verification. *Computer Vision and Image Understanding*, *76*(3), 173–190.

Drouhard, J.P., Sabourin, R., & Godbout, M. (1996). A neural network approach to off-line signature verification using directional PDF. *Pattern Recognition*, *29*(3), 415–424.

el-Yacoubi, A., Justino, E.J.R., Sabourin, R., & Bortolozzi, F. (2000). Off-line signature verification using HMMS and cross-validation. *Proceedings of the Ninth IEEE Workshop on Neural Networks for Signal Processing*, 859–868.

Fadhel, E.A., &. Bhattacharyya, P. (1999). Application of a steerable wavelet transform using neural network for signature verification. *Pattern Analysis and Applications*, *2*, 184–195.

Fang, B., Leung, C.H., Tang, Y.Y., Tse, K.W., Kwok, P.C.K., & Wong, Y.K. (2003). Offline signature verification by tracking of feature and stroke positions. *Pattern Recognition*, *36*, 91–101.

Harrison, W.R. (1958). *Suspect documents: Their scientific examination*. London: Sweet & Maxwell Ltd.

Hilton, O. (1992). Signatures review and a new view. *Journal of Forensic Sciences*, *37*(1), 125–129.

Ismail, M.A., & Gad, S. (2000). Off-line Arabic signature recognition and verification. *Pattern Recognition*, *33*(10), 1727–1740.

Justino, E.J.R., Bortolozzi, F., & Sabourin, R. (2001). Offline signature verification using HMM for random, simple and skilled forgeries. *Proceedings of the Sixth IEEE International Conference on Pattern Recognition*, 450–453.

Lee, S., & Pan, J.C. (1992). Offline tracing and representation of signatures. *IEEE Transactions on Systems, Man and Cybernetics, 22*(4), 755–771.

Locard, E. (1936). *Traité de criminalistique*. Lyon: Payoy.

Murshed, N.A., Sabourin, R., & Bortolozzi, F. (1997). A cognitive approach to offline signature verification. In H. Bunke & P.S.P. Wang (Eds.), *Automatic bankcheck processing* (pp. 339–364). Singapore: World Scientific Publishing.

Nagel, R.N., & Rosenfeld, A. (1977). Computer recognition of freehand forgeries. *IEEE Transactions on Computers, 26*(9), 895–905.

Nemcek, W.F., & Lin, W.C. (1974). Experimental investigation of automatic signature verification. *IEEE Transactions on Systems, Man and Cybernetics, 4*, 121–126.

Osborn, A.S. (1929). *Questioned documents*. New York: Boyd Printing.

Pender, D.A. (1991). *Neural networks and handwritten signature verification* [doctoral dissertation]. Stanford University.

Plamondon, R., & Lorette, G. (1989). Designing automatic signature verification and writer identification—The state of the art. *Pattern Recognition, 2*(2), 107–131.

Qi, Y., & Hunt, B.R. (1994). Signature verification using global and grid features. *Pattern Recognition, 27*(12), 1621–1629.

Quek, C., & Zhou, R.W. (2002). Antiforgery: A novel pseudo-outer product based fuzzy neural network driven signature verification system. *Pattern Recognition Letters, 23*(14), 1795–1816.

Rigoll, G., & Kosmala, A. (1998). A systematic comparison between on-line and off-line methods for signature verification with hidden Markov models. *Proceedings of the 14th International Conference on Pattern Recognition*, 1755–1757).

Sabourin, R., Genest, G., & Prêteux, F.J. (1997). Off-line signature verification by local granulometric size distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*(9), 976–988.

Shih, F.Y., & Wong, W.-T. (1995). A new safe-point thinning algorithm based on the mid-crack code tracing. *IEEE Transactions on Systems, Man and Cybernetics, 25*(2), 370–378.

Suen, C.Y., Xu, Q., & Lam, L. (1999). Automatic recognition of handwritten data on cheques—Fact or fiction? *Pattern Recognition Letters, 20*, 1287–1295.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on System, Man and Cybernetics, 15*, 116–132.

Xiao, X., & Leedham, G. (2002). Signature verification using a modified Bayesian network. *Pattern Recognition, 35*(5), 983–995.

Chapter V
Introduction to Speech Recognition

Sergio Suárez-Guerra
National Polytechnic Institute, Mexico

Jose Luis Oropeza-Rodriguez
National Polytechnic Institute, Mexico

ABSTRACT

This chapter presents the state-of-the-art automatic speech recognition (ASR) technology, which is a very successful technology in the computer science field, related to multiple disciplines such as the signal processing and analysis, mathematical statistics, applied artificial intelligence and linguistics, and so forth. The unit of essential information used to characterize the speech signal in the most widely used ASR systems is the phoneme. However, recently several researchers have questioned this representation and demonstrated the limitations of the phonemes, suggesting that ASR with better performance can be developed replacing the phoneme by triphones and syllables as the unit of essential information used to characterize the speech signal. This chapter presents an overview of the most successful techniques used in ASR systems together with some recently proposed ASR systems that intend to improve the characteristics of conventional ASR systems.

INTRODUCTION

Automatic speech recognition (ASR) has been one of the most successful technologies allowing the man-machine communications to request some information from it or to request to carry out some given task using the natural oral communication. The artificial intelligence field has contributed in a remarkable way to the development of ASR algorithms.

The more widely used paradigm in ASR systems has been the phonetic content of the speech signal, which varies from language to language, but there are no more than 30 different phonemes without some variations, such as accentuation, duration, and the concatenation.

The last one includes the co-articulation such as demisyllables and triphones. Considering all variations, the number of phonetic units will be increased considerably. Recently some researchers have considered the use of syllables instead of phonemes as an alternative for development of the ASR systems, because in general, the natural way to understand the language by the human brain is to store and recognize syllables not phonemes.

The automatic speech recognition is a very complex task due to the large amount of variations involved in it, such as intonation, voice level, health condition and fatigue, and so forth (Suárez, 2005). Therefore, in the automatic speech recognition system, for specific or general tasks, there is an immense amount of aspects to be taken into account. This fact has contributed to increase the interest in this field, and as a consequence, several ASR algorithms have been proposed during the last 60 years.

A brief review of ASR systems can be summarized as follows. At the beginning of 1950s, the Bell Laboratories developed an ASR system that was able to recognize isolated digits. The RCA Laboratories developed a single-speaker ASR for recognition of 10 syllables. The University College in England developed a phonetic recognizer, and in the MIT Lincoln Laboratory a speaker independent vowel recognizer was developed. During the 1960s, some basic tools for ASR systems were developed. The dynamic time warping is developed by the NEC Laboratories and Vintsyuk of the Soviet Union. In the Carnegie Mellon University, the automatic continuous speech recognition system with small vocabularies HAL 9000 was developed. During the 1970s, several isolated word recognition systems were developed, such as a large vocabulary ASR system by IBM. During this time also there was an important increase on the investment to develop ASR systems, such as the DARPA and HARPY project in the U.S. During the decade of the 1980s, the first algorithms for continuous speech recognition system with large vocabularies appeared. Also during this time the hidden Markov model (HMM) and neural networks were introduced in (development of) the ASR systems; one of these types of systems is a SPHINX system. The ASR systems have appeared as commercial systems during the decade of the 1990s thanks to the development of fast and cheap personal computers that allow the implementation of dictation systems and the integration between speech recognition and natural language processing. Finally, during recent years, it was possible to use the voice recognition systems in the operating systems, telephone communication system, and Internet sites where Internet management using voice recognition, Voice Web Browsers, as well as Voice XML standards are developed.

STATE-OF-THE-ART

During the last few decades, the study of the syllables as a base of language model has produced several beneficial results (Hu, Schalkwyk, Barnard, & Cole, 1996). Hu et al. (1996) realized an experiment where syllables belonging to the name of months in English were recognized. They created a corpus with a total of 29 syllabic units, and 84.4% efficiency was achieved in their system. Boulard (1996) realized similar works using the syllables in German. Hauenstein (1996) developed a hybrid ASR system: HMM-NN (hidden Markov models-neural networks) using syllables and phonemes as basic units for the model. He realized a performance comparison between the system using only syllables and another one using only phonemes, and he concluded that the system that combined both units (syllables and phonemes) presents higher performance than the system using syllables or phonemes separately. Wu, Shire, Greenberg, and Morgan (1997) proposed an integration of information at syllables level within the automatic speech recognizer to improve their performance and robustness (Wu, 1998; Wu et

al., 1997), taking 10% of the recognition error for digits voices of OGI (Oregon Graduate Institute) corpus. In a work by Wu (1998), 6.8% of recognition error rate for digits data using corpus of digits from telephone conversation was reported; here, the RSA system was a phoneme-syllable hybrid system. Jones, Downey, and Mason (1999) experimented with HMM to obtain the representations of the units at the syllable level. From these experiments they concluded that it is possible to substantially improve the performance of the ASR using a medium size database comparing with the monophonic models (Jones et al., 1999). In this case, the recognition rate was 60% compared with 35% obtained when a monophonic model was used, concluding that the practical applications must be satisfied by a hybrid system (Fosler-Lussier, Greenberg, & Morgan, 1999). They found that a great amount of phonetic phenomena that appeared in the spontaneous speech have syllabic features, and it is necessary to use more phonetic contextual windows for phonetic-based ASR systems (Fosler et al., 1999).

Weber (2000), in his experiments using segmentation of speech signal with different duration, found the recognition error rate of word (WER: Word Error Rate) is increased when the speech signal is distorted by noise. WER can be reduced increasing segmentation size using syllabic-based ASR. Meneido and Neto (2000) used the information at the syllable level to apply an automatic word segmentation system to improve ASR systems, which was applied to the Portuguese, and the WER was reduced. Some works began to consider a language model to improve the ASR system. The work of Meneido in Portuguese showed a method of incorporation of the speech information at the syllable level with Spanish language, because both languages share common features in the syllable level (Meneido & Neto, 2000; Meneido, Neto, & Almeida, 1999). Also the work of Menido et al. (1999) reports a successful detection of starting point of the syllable with about 93%, and they considered that wide range windows of context entrance (260m) are most appropriate.

In 1996, a summer meeting about continuous speech recognition was held in the Language and Speech Processing Center of Johns Hopkins University, where it was demonstrated that the use of the syllable level and the use of the individual internal pressure can reduce the WER of the triphones-based ASR systems (Wu, 1998). In the same year, Hauenstein (1996) introduced the syllabic units into the continuous speech recognition systems and realized a performance comparison with a phonetic units-based system. His experiments showed that a phonetic-based system presents better performance for the continuous speech recognition system, while a syllabic-based system is better for the discontinuous speech recognition system, reaching more than 17.7% of a recognition error.

Five years later, Ganapathiraju, Hamaker, Picone, and Doddington (2001) showed that the syllabic-based system reaches a similar performance to the phonetic-based system for continuous speech recognition task. Firstly, they observed that there are a considerable number of triphones with little information to construct an appropriate speech model. In addition, they noticed that in the triphones, there is minimum integration of the spectral and temporal dependencies due to the short duration of masks, and one by one mapping of the words to its corresponded phonemes caused a great amount of different categories for the same sound. To solve this problem, they proposed an integrated system with syllables model and an acoustic model. The proposed system reduced 1% of WER with respect to a triphones-based system; here a switchboard database was used. They reported an 11.1% error rate when the syllable of the context was used independently. In 1997, the project generated by Edinburgh, "Improvement of the speech recognition using syllabic structure" (Project ESPRESSO), which is motivated by the fact that the conventional HMM-based ASR system no longer provided the expected performance,

and it is necessary to find more efficient models than the conventional one that used phonemes of the context. The project intended to find a suitable model to capture the context dependency without requiring an excessive number of parameters for its representation. Its first stage, in which the use of the phonetic syllables characterized for speech recognition was researched by King, Taylor, Frankel, and Richmond (2000), was finished in 1999, while in the second phase, new models for the speech recognition were developed, taking into account the continuous nature of the schemes found in the previous phase. This phase still is in the accomplishment process.

For the Spanish, speech recognizers for specific applications have been developed. Córdoba, Menéndez-Pidal, and Macías Guasara (1995) worked in an ASR system for digits for telephony applications. Also the problem of variation by the emotional state of the speaker was analyzed (Montero, Gutiérrez Arreola, & Colás, 1999). In Mexico the authors have worked on the development of an ASR version for the Spanish spoken in Mexico, which is based on a platform developed originally by the English language (Fanty, 1996; Munive, Vargas, Serridge, Cervantes, & Kirschnning, 1998). Regarding the study about the syllables, Feal (2000) mentioned the idea to use them as a unit for the synthesis of the Spanish spoken speech, and presents a list of the syllables that are used in this language, using a database created from books of Spanish Literature (Feal, 2000). Also, he proposed an algorithm for syllables segmentation in continuous text from the phonetic transcription of the words.

For languages like the Cantonés, spoken in China, which are also based on syllabic units, several experiments have been carried out to analyze the performance of a speech recognition system based on syllables (Lee & Ching, 1997; Peskin, Gillick, & Liberman, 1991). Lee and Ching (1997) used a speaker dependent speech recognition system, with a vocabulary of between 40 and 200 syllables. In the case of the 200 syllables, a recognition rate of 81.8% was obtained. Some languages, such as Cantonese, are mainly based on tones with monosyllabic features; thus the ASR systems suitable for this kind of language were developed. The system proposed by Peskin et al. (1991) consists of two essential parts: a detector of tones and a basic syllables recognizer using a multilayer perceptron neuronal network. The vocabulary used to evaluate the ASR system was of 40 to 200 syllables, which represent the 6.8% and 34.4% of the total of the syllables of such language. The number of speakers was equal of 10: five men and five women. Another language that has been the object of study is Mandarin language (Chang, Zhou, Di, Huang, & Lee, 2000), which is similar to the previous one; it is based on tones and syllables, but unlike Cantonese, the tone at the end of the syllable is emphasized. Within the main features of this language is that it inserts the fundamental tone ("pitch") as a recognition parameter. The database used in this case included a total of 500 speakers and 1000 phrases; the algorithm obtains the "pitch" in real time, adding also the delta and double delta components. They obtained an error rate of 7.32% when no pitch processing was made. When they added the pitch for the analysis of tones, the error rate was reduced to 6.43%. When the information of the pitch and the set of syllables were combined, the error rate was 6.03%, which represents a reduction of an accumulated error of 17.6%.

These results denote that factors such as the signal energy and the duration can contribute to reduce the ambiguity problem of different tones. As mentioned in some references of this chapter, it can be applied to the Spanish language such as the works of Meneido, Neto and Almeida (1999) and Meneido and Neto (2000), which were done for the Portuguese. Meneido et al. (1999) show that recent developments have allowed the observance that the syllable can be used like a unit of recognition for the Portuguese language, which is a highly syllabic language, because the borders of the syllables are better defined than the phonemes

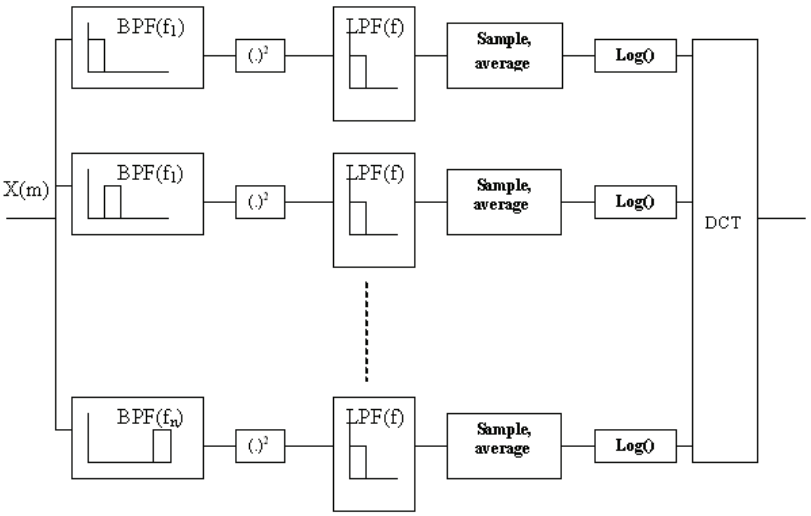
borders. Among their main works, the segmentation of speech signal to syllabic units is proposed; it is realized by means of the feature extraction oriented to perception. These features were post-processed by means of a simple threshold-based system or neural networks based on the syllables borders estimation. The experimental results showed 93% of correct detection of beginning of syllables with 15% of insertion rate, using a window of approximately 260 ms. The obtained results showed that, with a window of approximately 260 ms, 93% of correct detection of the beginnings was reached, with an insertion rate of 15%. The inclusion of the triphones on part of the study of the syllables is another excellent aspect mentioned by Meneido and Neto (2000). As an introduction to their research, they considered that an exact knowledge of the beginnings of the syllable can be useful to increase the recognition rate obtained until now. The database used for testing the ASR system was the BD-PUBLIC, from which they obtained a segmentation adapted of the same one of the order of 72%. In this work they propose four methods to segment the signal in syllabic units. Using the database under analysis they extracted a corpus with a total of 750 phrases, containing a total of 3408 words, from which 1314 were different with a total of 616 different syllables. It is possible to emphasize that the work of Meneido and Neto (2000) looks strongly influenced by the work of Villing, Timoney, Ward, and Costello (2004), in which the utility that has the syllabic unit is mentioned. These results were obtained from a set of experiments using digits of the database SWITCHBOARD. In such experiments they demonstrated that the syllables have better execution than the triphones in approximately 1.1%. Also, the analysis of an algorithm used for the segmentation of syllabic units appears, which are compared with the results of the algorithms of Mermelstein and Howitt used to make the same activity. A recognition rate of 93.1% reported is greater than the 76.1% obtained by the algorithm of Mermelstein and the 78.9% obtained by the Howitt algorithm. Another contribution of Meneido (Meneido & Neto, 2000) consists in the performance analysis that used the neuronal network and the Hidden Markov Models, and in the recognition of phrases of the Portuguese by means of a new method of segmentation in syllabic units. Hartmut et al. (1996) proposed an algorithm for the segmentation of the word in syllabic units, providing a segmentation error of the 12.87% for isolated words and of the 21.03% for the spontaneous speech. This algorithm used a great amount of digital filters to perform the corresponding segmentation.

FUNDAMENTALS OF VOICE RECOGNITION

The automatic speech recognition is a complex pattern recognition task in which the speech signal is analyzed to extract the most significant features, once the speech signal has been digitalized with a sampling rate between 8 and 16 kHz. This section describes some of the most used feature extraction methods reported in the literature (Jackson, 1996; Kirschning-Albers, 1998; Kosko, 1992; Sydral, Bennet, & Greenspan, 1995). The first method is the Fourier analysis, which can be observed from Figure 1, which consists of applying the fast Fourier transform (FFT) to the set of samples under analysis. Regularly, this representation in the frequency domain is distributed by using the well-known MEL scale, where the frequencies smaller than 1KHz are analyzed using a linear scale, and the frequencies larger than 1kHz use a logarithmic scale (Bernal, Bobadilla, & Gomez, 2000), with the purpose of creating an analogy with the internal cochlea of the ear that in its natural way works like a frequencies splitter. This can be observed in the expression of equation 1.

Linear predictive coding (LPC) is a method whose purpose is to find a set of representative vectors denominated vectors code, from which it forms a matrix of representative vectors that as

Figure 1. Filters banks implementation for cepstral-based system proposed by Suk and Flanagan



well conforms what book is denominated code. Based on its hypothesis in the fact that sample XT of a signal can be predetermined from the k previous samples, if we know the weight by which each one of them is affected by all the XT-K, previous XT-K-1, XT-K-2, and so forth, samples.

Cepstrales coefficients analysis is another method used for feature extraction of speech signals that represent the inverse Fourier transform of the logarithm of power spectral density of the speech signal (Rabiner & Biing-Hwang, 1993). Finally, the perceptual linear prediction analysis (PLP) is another widely used method, resulting of the physiological characteristics, but that cannot be represented graphically (Kirschning-Albers, 1998).

$$Mel(f) = b\log_{10}(1+\frac{f}{c}), \qquad (1)$$

The essential characteristic of the speech signal is its excessive time varying characteristics. At the present time, the speech signal is analyzed from two different points of view: the acoustic level and the temporary level (Tebelskis, 1995). From the acoustic point of view, the aspects to be analyzed are the accent, the pronunciation, the frequency resonance of vocal tract ("pitch"), and the volume, among others, while in the case of the temporary variation, the different durations presented in a set of speech samples are analyzed. Although in general, both previously mentioned aspects are not independent, they are assumed to independent among them. Between both aspects mentioned before, the time variation is easier to handle. In principle, a linear deformation type of an unknown speech signal was used to compare with a given reference signal. In this case the obtained result was not optimal. Next, a non-linear type deformation was used, which gave as a consequence the appearance of DTW (Dynamic Time Warping) algorithm (Rabiner & Levinson, 1990). At the present time, such algorithm has yet to be used to a great extent. The acoustic variation is more difficult to model, due to its heterogeneous nature. Therefore, the study of the speech recognition has extended its field in this aspect. The different perspectives to analyze the speech recognition are reduced to the following ones: models of reference or groups, knowledge, stochastic or statistical models, artificial neuronal networks, and hybrid methods. Among them the most widely used method is the statistical method

that makes use of the hidden Markov model (Kita, Morimoto, & Sagayama, 1993). Several developed systems using such methods appear in specific domains, although these present some limitations that suggest the necessity to develop new methods.

The speech is a continuous time signal and then in order to be processed by a digital processor is indispensable in its digitalization, with an appropriate sampling rate. This factor is very important, however, although a high quality digitalized speech signal could be available since there are still many factors that must be taken in account to be able to develop reliable ASR systems (Kirschning-Albers, 1998) such as the vocabulary size, if the system is speaker dependent or speaker independent, if the ASR is required to recognize isolate words or continuous speech, if the ASR is intended for a particular task, or for general applications, and so forth.

The number of words that the ASR system is required to recognize is a very important factor because the system performance decreases when the vocabulary size increases. Reports of an acceptable recognition average show that most ASR performs fairly well when the word numbers is smaller than 1000 and that the performance worsens when the word number is beyond 1000.

The ASR, depending on the particular application, can be speaker dependent and the speaker independent. The speaker dependent ARS is commonly used when a high recognition rate is required, although its use is restricted to one particular speaker, while the speaker independent ASR can be used in applications that require handling a relatively large number of users, although its recognition performance degrades when the number of speakers increases. Also, both ASR systems can be classified into isolated and continuous speech ASR, depending on the particular application. In the first case, the ASR is required to recognize isolated words or short sentences without relation with other previous speech. In continuous speech recognitions, the system is required to recognize long sentences and even a conversation. In this case, the relations among the different components of a given sentence become very important. Because both situations are quite different, the ASR systems required in both applications are substantially different. Besides the characteristics mentioned previously, the ASR systems can be divided also into ASR systems for particular applications such as to recognize some specific commands, or for general applications.

Another aspect that must be taken in account to develop reliable ARS systems is the distortion introduced by the additive noise as well as variation introduced in the speaker voice due to sickness, possible stress, and so forth.

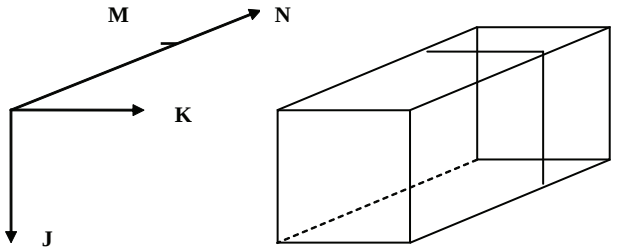
GENERAL FRAMEWORK FOR SPEECH RECOGNITION

All researches related with speech recognition require carrying out a set of tasks to be performed independently of the goal to be reached; some of them are summarized as follows: (a) to prepare the speech corpus to use in the research; (b) analysis of the required conditions to obtain a corpus of good quality; (c) to define the preprocessing to be applied to the corpus file; (d) determine the algorithms and methods that will be used for features extraction to be use in speech recognition; (e) estimate the cepstrals or melspec and so forth; (f) to train the recognition system; and g) to verify the system performance using the correct recognition average.

Prepare the Speech Corpus to Use in the Research

The speech corpus is a set of files with digitized speech that are used for feature extraction during training and evaluation of the developed ASR system. Independently of the application, isolated word recognition, that is, words or short phrases,

Figure 2. Three-dimensional representation speech corpus



speaker dependent or independent approach, as well as speaker recognition application, which will be analyzed in an accompanying chapter, the method to construct a reliable speech corpus is basically the same one. The continuous speech problem is not analyzed in this chapter.

To construct a reliable speech corpus, the followings aspects must be considered: (a) a codebook must be designed containing the words or phrase to be recognized by the ASR system. Here the codebook size depends on the application; (b) for each speaker working in the developing of the system (K speakers), N files of each command will be recorded; (c) the recorded files are divided into sets, M commands, will be used to train the system, and N-M for testing the behavior of the system using the recognition rate.

Requirements to Obtain a Good Quality Corpus

It is very important, during the sound recording stage, to consider the recording conditions, such as distance to the microphone and adjustment of the gain and sensitivity of the microphone. All recorded commands must be monitored graphically to be sure that they fulfill the minimum requirements such as the signal level. It is important that all recorded signals have similar average amplitude and that these must not be saturated nor have very little amplitude. An example of these two cases is shown by Figure 3(a) and 3(b), respectively, while Figure 3(c) shows the same signal with a good amplitude level. It is very important to avoid the situations shown in Figure 3(a) and 3(b), which can be a cause of performance degradation. The amplitude normalization, in the preprocessing stage, can be used to reduce this problem; however, it is important to avoid the recording and storage of files in the corpus with very low or very large amplitude. In Figure 3, three examples of recorded signals are shown. The first one (Figure 3[a]) is saturated; the second one (Figure 3[b]) has very low amplitude; and the third (Figure 3[c]) is an amplitude normalized signal.

Preprocessing Applied to the Corpus Files

The preprocessing stage is commonly used to improve the system during the training and normal operation stages. It is usually carried out in a series of steps that standardize the characteristics of the speech signals recorded in the time domain. This stage can be summarized as follows: (a) normalization of the recorded signal amplitude, (b) pre-emphasizing the standardized signal, and (c) detection of the starting and final speech signal point, with the purpose of eliminating the silent intervals at the beginning and ending of the words.

The file amplitude normalization is done with the purpose of obtaining a greater similarity between files that contain the same command, avoiding variations of the amplitude during the recording process due to speaker position with respect to the microphone, fatigue, or distraction.

Figure 3. Examples of recording with amplitude: (a) saturate, (b) low, and (c) medium

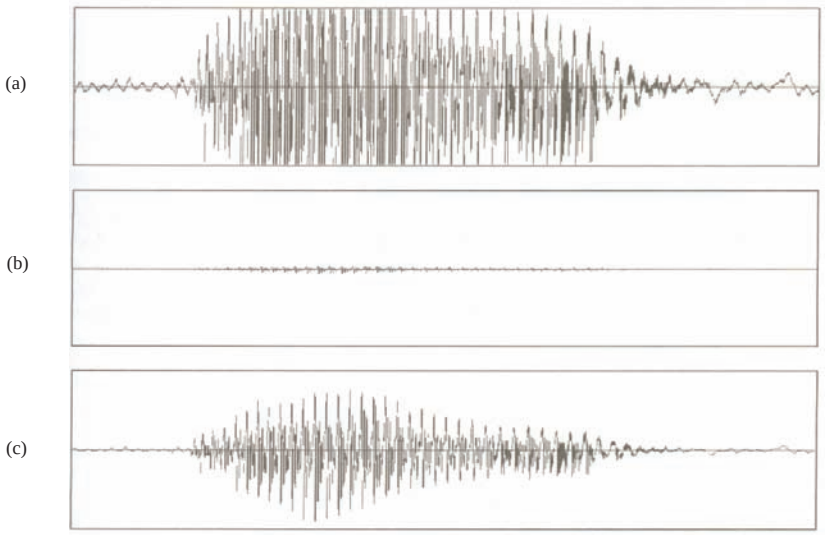
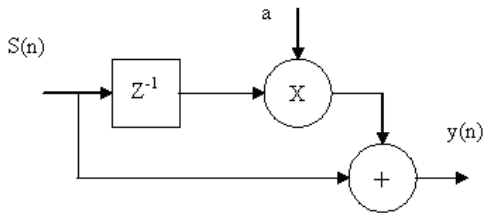


Figure 4. Pre-emphasis filter response



$$y(n) = S(n) - a*S(n-1)$$

The pre-emphasis is used to compensate the lost that suffer the high speech signal frequencies by effect of the propagation and radiation from the vocal cavity to the microphone. The pre-emphasis is performed by filtering the speech signal with a first order filter whose output signal is given by passing the speech signal through a first order filter as shown in Figure 4. This filter improves the efficiency of the stages used to calculate the speech spectrum, increasing, from the hearing point of view, the sensibility of the frequencies components larger than 1 KHz. Note that the pre-emphasis is not applied if the task consists of the analysis and extraction of the fundamental tone.

The detection of beginning and final point of a word is done by taking into account the activity, the energy activity, and zero crossing of the speech signal, *S*(*n*), with respect to the values that are in silence conditions. The duration of the segment under analyze is of 5 ms approximately. The behavior of the energy increase or the zero crossing indicates that a speech signal is present. Clearly, this activity can be due to additive noise; however, in most cases the ASR system can control the presence of it during the recordings. The zero crossing variation is mainly due to the emission of an explosive signal (`p', `t', `k', `b') or random noise (`s', `f', `ch', `j', `z'), while the energy variation happens in the presence of loudness vowels (`á', `é', `í', `ó', `ú'), semivowels, or consonants (`m', 'n', `ñ', `w', `d', `l', `g', `y'). In order to detect the presence of isolated words, the same procedure is done, but in this occasion the detection starts at the end of the data set and proceeds toward the

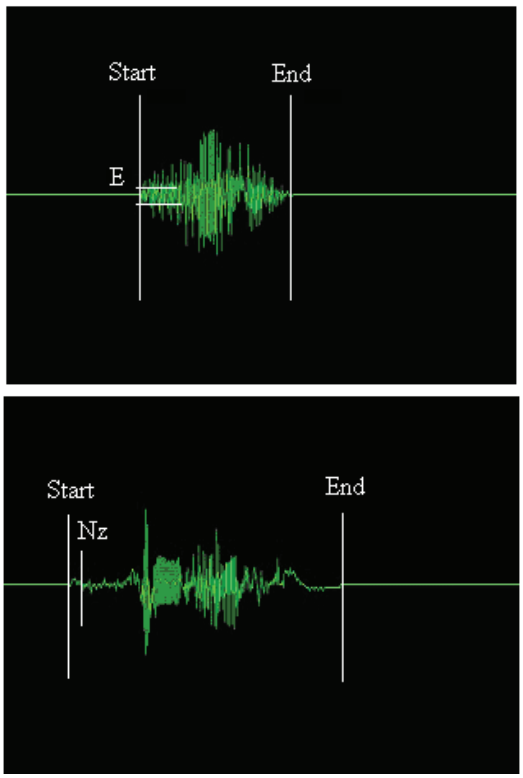
principle. If the task of determining beginning and end involves a phrase where several words exist, the process is done by determining where there are segments larger than, usually between 30 and 50 milliseconds, according to the characteristics of the speaker who produces the voice.

In Figure 5 we can observe the necessity of working with the energy threshold and zero crossing, according to the type of sound to be analyzed at the initial word: (a) sonorous sound "ahora," (b) noise sound "silos." How it is possible to be appreciated. For the case of the word "silos," the energy threshold does not detect the beginning of the word, and the same happens for the end.

Energy Analysis

Consider the energy contained in a silence segment of 1s duration, which for $N = f_s$, where f_s is the sampling frequency has an energy given by:

Figure 5. Start and end point detection in (a) Spanish word "ahora," (b) Spanish word "silos"



$$E_{sil} = \sum_{i=0}^{N-1} (S_i)^2, \quad (2)$$

while the energy average is given by:

$$Ep_{sil} = \frac{E_{sil}}{20}. \quad (3)$$

It is the average energy in a 5 ms segment, that is, the size of segment that is used to estimate the time interval in which the voice activity is present. Next the energy threshold use to determine the starting and ending point of the speech segment is given by:

$$Eu = Ep_{sil}(1+\%), 0.1 < \% < 0.2, \quad (4)$$

where the parameter % is chosen by the user. This is the value that must be considered during the analysis. The first sample of the segment that fulfills that condition is considered as the starting point of the speech segment under analysis, while the last sample of the last segment that satisfies this condition is considered as the ending point.

Analysis of Zero Crossing

Consider the number of zero crossings of a silence segment of 1s length, assuming that $N = f_s$ samples:

$$NZ_{sil} = \sum_{0}^{N-1} Signo(Si+1) - Signo(Si) > 0. \quad (5)$$

In a similar way the average value for a segment of 5 ms is given by:

$$NZP_{sil} = \frac{NZsil}{20}. \quad (6)$$

This value represents the average of zero crossing in a 5 ms segment, which is the size of the segment used to analyze and to determine the starting point of speech segment when it begins with a noise consonant. Next, *NZu*, which is the zero crossing threshold, is given by:

$$NZu = MZp_{sil}(1+\%), \quad (7)$$

where the value of % is chosen by the ASR user. This value must be considered by the analysis. Here, as in the energy average method, the first sample of the first segment that fulfills this condition is considered as the first sample of the speech signal segment. Once the starting point is detected, a segment of 20 or 25 ms is used for feature extraction.

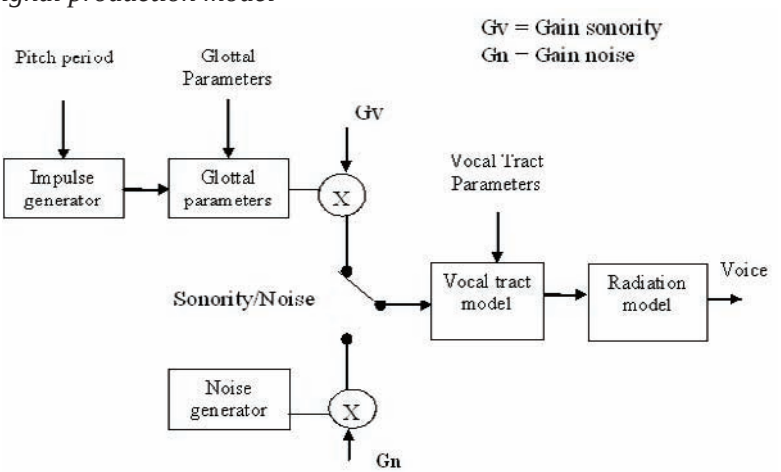
Feature Extraction Algorithms

Some authors consider the feature extraction as part of the preprocessing stage; however, because of the importance that the feature extraction has in the performance of ASR systems, the feature extraction is presented separately. Several methods have been proposed to feature extraction; among them, the autoregressive parameters obtained using the linear prediction method, a_p, provides the better results to characterize the human speech. Using these parameters we can assume that the speech signal can be represented as the output signal of an all pole digital filter whose excitation is an impulse sequence with a frequency equal to the pitch of speech signal under analysis, when the segment is voiced, or with noise when the segment is unvoiced. A variant is to use the cepstrales coefficients of the speech signal, which can be obtained from the LPC coefficients; both of them have been shown to perform fairly well in feature extraction in speech recognition system. The speech production model is shown in Figure 6.

The speech production model shown in Figure 6 is strongly dependent on the estimation of the autoregressive parameters, a_p, which can be obtained using the linear prediction method, which can be summarized as follows: (a) firstly the input signal is segmented to obtain the useful signal, without silence, in segments from 25 to 20 ms overlapping the segments; (b) apply the Hamming window to the segmented signal (Childers, 2000; Williams, 1996); (c) estimate the prediction order, that is, the amount of p values for each segment; (d) calculate the autoregressive coefficients or linear prediction coefficients, LPC, for each segment; (e) using the estimated LPC parameters estimate the cepstrales coefficients, (f) finally obtain the coefficients average.

To obtain the features vector, firstly the speech signal is divided in segments of 20 to 25 ms, which is an established standard to characterize the dynamics of the operation of the phonate system. From multiple observations one concludes

Figure 6. Speech signal production model



that the change from a phoneme to another one in the speech happens approximately in that time interval, although there are phonemes like the explosives that are a little shorter. The overlapping of segments is necessary to obtain the dynamics of change of the most representative segment characteristics. Usually the overlapping is applied with a 50% overlap. If the objective using the autocorrelation is to determine the existence of the pitch, FO, in the segment under analysis, the segment duration is taken with a length of 40 ms.

To avoid distortion of the segmented speech due to the discontinuities introduced during the segmentation process, a window function, typically the Hamming window, given by following equation, is used:

$$W(n) = 0.54 - 0.46\cos(2\pi n / N), \text{ for } 0 \le n \le N-1,$$

where N is the number of samples of the used segment.

After the speech signal is segmented, the p autocorrelation coefficients are estimated, where p is the linear predictor order. The autocorrelation function can be estimated using the unbiased autocorrelation or the biased autocorrelation algorithms (Childers, 2000), where for the biased case we have:

$$Rss(k) = \frac{1}{N-|k|}\sum_{i=0}^{N-1-|k|} S(i+|k|)S(i), \qquad (8)$$

where |k| < **p** +1. And for the unbiased case;

$$Rss(k) = \frac{1}{N}\sum_{i=0}^{N-1-|k|} S(i+|k|(S(i), \qquad (9)$$

where |k| < **p** +1, N is the number of speech data and **p** the linear prediction order. The more widely used autocorrelation estimation method is the unbiased one, although solutions exist using the biased algorithm. However, in such a case the calculation of LPC coefficients is done using the covariance matrix.

Once p autocorrelation coefficients are estimated for each segments I, the linear prediction coefficients, a_p, coefficients for each segment, are estimated minimizing the mean square value of prediction error as follows (Childers, 2000): consider that the signal at time n is estimated as a linear combination of the pass samples of input signal, such that:

$$\hat{s}(n) = -(a_1 s(n-1) + a_2 s(n-2) + ... + a_p(n-p)), \qquad (10)$$

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k s(n-k), \qquad (11)$$

where a_k, k=1,2,…,p are the linear prediction coefficients. Notice that with this model a real value sampled data at time n is predicted, using a linear combination from the previous sampled data. Therefore, it is valid to affirm that a filter can be designed that be able to estimate the data at time n only using the previous data at times n–1, because:

$$\hat{s}(n) = -a_e s(n-1), \qquad (12)$$

where a_e is the linear prediction coefficient and $s(n)$ and $\hat{s}(n)$ are discrete samples. The error sequence between both sequences is given by:

$$e(n) = s(n) - \hat{s}(n) = s(n) + a_e\, s(n-1), \qquad (13)$$

where the linear predictor coefficients are estimated such that mean square value of prediction error becomes a minimum. Consider:

$$E_T{}^2 = \sum_{n=0}^{N-1} e^2(n) = \sum_{n=0}^{N-1} (s(n) - \hat{s}(n))^2$$
$$= \sum_{n=0}^{N-1} (s(n) + a_e s(n-k))^2. \qquad (14)$$

In order to obtain the linear predictor coefficient, a_e, we take the partial derivative of $E_T{}^2$ with respect to a_k and set it to zero, that is:

$$\frac{\partial E_T^2}{\partial a_e} = 0 = 2\sum_{n=0}^{N-1}(s(n) + a_e s(n-1))s(n-1), \qquad (15)$$

to obtain,

$$a_e = \frac{-\frac{1}{N}\sum_{n=0}^{N-1} s(n)s(n-1)}{\frac{1}{N}\sum_{n=0}^{N-1} s(n-1)s(n-1)} = -\frac{Rss(1)}{Rss(0)}. \qquad (16)$$

In order to generalize the previous result for *p* coefficients, it is necessary firstly to consider that (13) and (15) show that the prediction error is orthogonal to the input data (Figure 7).

Consider a general case in which the speech sample at time n is estimated as a linear combination of p previous samples as follows:

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k s(n-k) \qquad (17)$$

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + e(n) = \hat{s}(n) + e(n). \qquad (18)$$

Next consider the *Z* transform of equation (18), which is given as:

$$S(z) = -\left[\sum_{k=1}^{p} a_k z^{-k}\right] S(z) + E(z) = \hat{S}(z) + E(z). \qquad (19)$$

Then,

$$S(z) = \frac{E(z)}{1 + \left[\sum_{k=1}^{p} a_k z^{-k}\right]} = \frac{E(z)}{A(z)}, \qquad (20)$$

where,

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}. \qquad (21)$$

Equation (20) denotes the transfer function of an all pole filter shown in Figure 8.

The total quadratic error for this interval is given in Equation 22, where $s(n)$ are the windowed data. Next consider the minimization of equation (22) that is obtained taking the derivate of E_T with respect to the filter coefficients and setting it equal to zero, that is,

$$\frac{\partial E_T^2}{\partial a_k} = 0, \quad k = 1, 2, \ldots p. \qquad (23)$$

Thus, from equations (22) and (23) it follows that:

$$\sum_{k=1}^{p} a_k \sum_{n=0}^{N-1} s(n-k)s(n-i) = -\sum_{n=0}^{N-1} s(n)s(n-i), \quad k = 1, 2, \ldots p \qquad (24)$$

Equation 22.

$$E_T^2 = \sum_{n=0}^{N-1} e^2(n) = \sum_{n=0}^{N-1} (s(n) - \hat{s}(n))^2 = \sum_{n=0}^{N-1} \left(s(n) + \sum_{k=1}^{p} a_k s(n-k) \right)^2 \qquad (22)$$

Figure 7. Orthogonally between the data and error

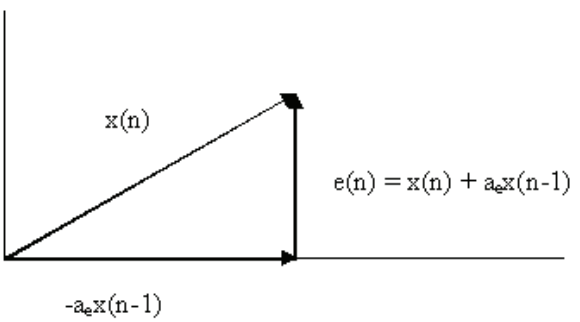
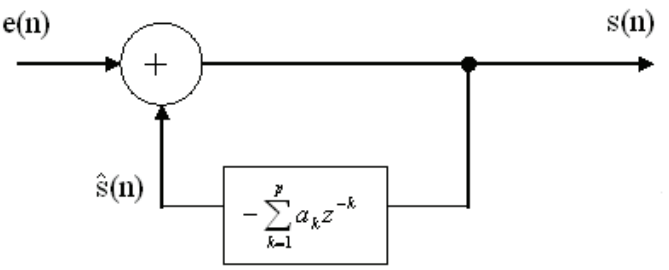


Figure 8. AR filter



which can be expressed in terms of the autocorrelation function of input signal as follows:

$$\sum_{k=1}^{p} a_k Rss(i-k) = -Rss(i), \quad k = 1,2,...p, \qquad (25)$$

where,

$$Rss(k) = \frac{1}{N} \sum_{n=0}^{N-1-|k|} s(n)s(n+|k|). \qquad (26)$$

The term 1/*N*, not mentioned previously, is the scale factor of the partial autocorrelation, which is a guarantee for the stability of the estimated LPC coefficients. Writing equation (25) in a matrix form can be seen in Equation 27.

Taking into account that autocorrelation coefficients are real and even, that is, *Rss*(*k*–*m*)=*Rss*(*m*–*k*), for *k*=1,2,3,..p and *m*=1,2,3,…,*p*, the matrix on the left part becomes a Toeplitz type matrix (left part), and then the terms a_k can be obtain by using the Levinson-Durbin algorithm. An additional equation for the estimation of a_k is based on total quadratic prediction error, which is given by:

$$E_T^2 = Rss(0)a_0 + \sum_{k=1}^{p} a_k Rss(-k), \qquad (28)$$

where a_0=1. Again assuming that the autocorrelation coefficients are even, as in equation (27), from equation (28), what follows can be seen in Equation 29.

Finally, the linear predictor coefficients can be obtained solving the normal equations given by equations (27) or (29), using the Leveinson-Durbin algorithm (see Box 1).

Within the second loop, the one of *k*, is formed the a_p LPC coefficients.

Equation 27.

$$\begin{bmatrix} Rss(0) & Rss(-1) & \cdots & Rss(-(p-1)) \\ Rss(1) & Rss(0) & \cdots & Rss(-(p-2)) \\ \vdots & \vdots & \cdots & \vdots \\ Rss(p-1) & Rss(p-1) & \cdots & Rss(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} Rss(1) \\ Rss(2) \\ \vdots \\ Rss(p) \end{bmatrix} \qquad (27)$$

Equation 29.

$$\begin{bmatrix} Rss(0) & Rss(-1) & \cdots & Rss(-p) \\ Rss(1) & Rss(0) & \cdots & Rss(-(p-1)) \\ \vdots & \vdots & \cdots & \vdots \\ Rss(p) & Rss(p-1) & \cdots & Rss(0) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} E_T^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (28)$$

The advantage of estimating the coefficients of the all pole filter using the partial correlation coefficients α is that the resulting filter is always stable. Thus from the previous process it follows that the LPC coefficients are estimated in a recursive form as shown in Table 1 for a predictor order equal to p=10.

From the table it follows that the terms $a_{0,m} = 1$, because the estimation task due to in each iteration, the terms $a_{\#,m}$ and $a_{m,\#}$, are different, $a_{1,2} \neq a_{1,2}$; the columns value are the result of the vector a_p of LPC coefficients $a_p = a_0, a_1, \cdots, a_p$, in each iteration.

Depending on the ARS application, it may be useful to take the LPC averages of each word or word sections of each word. This a_p coefficients average may be used as the behavior model of each word or section (Buzo & Gray, 1980). Thus in the case of all word, the m^{th} LPC becomes:

$$\hat{a}_m = \hat{a}_m = \frac{1}{I}\sum_{i=1}^{I} a_{i,m}, \quad 1 \le m \le p. \tag{30}$$

The cepstral coefficients, which are widely used in both speech as well as speaker recognition problems, can be directly obtained from the LPC coefficients, or by means of the inverse Fourier transform of input signal power spectral density. These coefficients have shown to be very good parameters for the development of speech recognition, sometimes better than the LPC.

The ceptrals coefficients can be estimated from the LPC coefficients applying the following expression:

$$c_n = -a_n - \frac{1}{n}\sum_{i=1}^{n-1}(n-i)a_i c_{n-i}, \text{ for } n > 0, \tag{31}$$

where c_n is the n^{th} LPC-cepstral coefficients (CLPC), a_i are the LPC coefficients, and n is the cepstral index.

Another form to carry out the cepstrals estimation is use of the power spectral density of the input signal. To this end we can use the homomorphic techniques of signal processing, which have had great importance within the speech recognition field. The homomorphic systems are a class of non-linear systems that obey to a superposition principle. The motivation to do a homomorphic processing of the speech signal is shown in Figure 9.

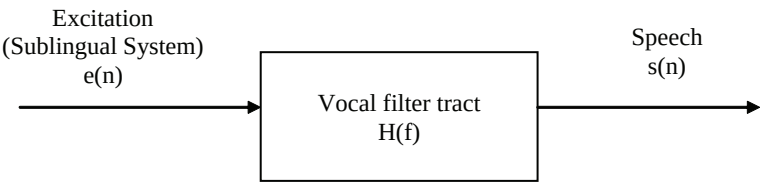
Box 1. Leveinson-Durbin algorithm

Given: p, $Rss[0,p]$
Calculate: $\{\alpha_m, a_{k,m}, c_m: 1 \le m \le p, 1 \le k \le p\}$
Begin: $\alpha_0 = Rss[0,0]$
$a_{0,0} = 1$
Body of the program: for m =0 to p–1, do
$\gamma_m = \sum_{k=0}^{m} Rss[m+1-k]\, a_{k,m}$
$c_{m+1} = -gamma_m/\alpha_m$
$\alpha_{m+1} = \alpha_m(1 - c^2_{m+1})$
$a_{m+1,m} = 0$
for k =0 to m +1, do
$a_{k,m+1} = a_{k,m} + c_{m+1}\, a_{m+1-k,m}$
end loop k
end loop m

Table 1. Vector a_p in iteration p selected

Iteration Coefficients	0	1	2	3	…	9	10
a_0	1	1	1	1	…	1	1
a_1	-	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	…	$a_{1,p-1}$	$a_{1,p}$
a_2	-	-	$a_{2,2}$	$a_{2,3}$	…	$a_{2,p-1}$	$a_{2,p}$
a_3	-	-	-	$a_{3,3}$	…	$a_{3,p-1}$	$a_{3,p}$
a_4	-	-	-	-	…	$a_{4,p-1}$	$a_{4,p}$
a_5	-	-	-	-	…	$a_{5,p-1}$	$a_{5,p}$
a_6	-	-	-	-	…	$a_{6,p-1}$	$a_{6,p}$
a_7	-	-	-	-	…	$a_{7,p-1}$	$a_{7,p}$
a_8	-	-	-	-	…	$a_{8,p-1}$	$a_{8,p}$
a_9	-	-	-	-	-	$a_{9,p-1}$	$a_{9,p}$
a_{10}	-	-	-	-	-	-	$a_{p,p}$

Figure 9. The homomorphic technique can be use to separate the vocal tract action of the excitation signal (lineal filter on the time variable)



The cepstrum estimation involves firstly the calculation of the power spectral density of input signal. Once the power spectral density of the input signal in the selected interval is obtained, next the logarithm of the previous power spectral density is calculated, and the inverse Fourier transform is applied to the resulting signal, which is all defined positive. This procedure is illustrated in Figure 10.

The cepstrum of a speech signal can be divided in two parts: the lower portion of time that corresponds to the transfer function of the vocal track and the high portion that corresponds to the excitation. To smooth the cepstral estimation of the vocal track, the high portion of it can do zero and the Fourier transform applied to the low portion part of the ceptrum, resulting in a smoothed cepstral spectrum. This represents the magnitude of vocal tract response when the excitation effect has been removed, including the fundamental tone. Whereas the LPC analysis represents only the poles of the system, the use of the cepstrales is better like the solution to represent the nasalized voice (Chen, 1988).

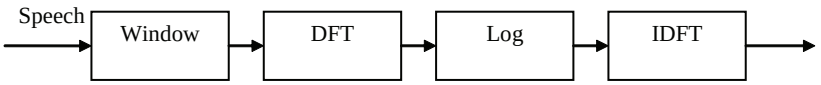
Usually the number of cepstrales coefficients that are used is similar to the number of LPC coefficients, by convenience and to avoid noise.

Consider the inverse Fourier transform of the logarithm of input signal power spectral density given by:

$$c(n) = \frac{1}{N}\sum_{k=0}^{N-1} \log_{10} \left| S_{med}(k) \right| e^{j\frac{2\pi}{N}kn}, \quad 0 \le n \le N-1. \tag{32}$$

In equation (32), the $c(n)$ is known as the n^{th} cepstral coefficient derived from the Fourier transform and N is the number of points used to calculate the discrete Fourier transform (DFT).

Figure 10. Algorithm for cepstrum estimation



This equation is also known as the inverse DFT of the logarithmic spectrum. It can be properly simplified considering that the spectrum of the logarithm is a symmetrical real function, thus equation (32) becomes:

$$c(n) = \frac{2}{N}\sum_{k=1}^{N} S_{med}\left(I(k)\right)\cos\left(\frac{2\pi}{N}kn\right), \qquad (33)$$

where n is the coefficient index, which is usually keep lower than 20, that is, ($n \leq 20$). $I(k)$ denotes a function that translates the position of a value in frequency at the interval where it is contained. It is in fact the measurement of the period of the frequencies that contains the signal. The cepstrum is also useful for the detection of the pitch in voiced segments. In such a case, as shown by Veeneman (1998), the distance from the origin of the graph to its maximum value is the period of the fundamental tone (T0). Usually, to this end, a segment signal of 64 ms, sampled to 8000 Hertz, is used. The election of the maximum point to determine the position of the fundamental tone is based on the fact that for standard voices the pitch takes place between 80 and 250 Hertz, corresponding to the maximum level with the frequency for women and the minimum for the men. That in time is equivalent to an interval between 4 and 12.5 ms.

For isolated words recognition, it is possible to take the average from the LPC or cepstrales coefficients (CLPC) of the total of segments contained in the word to generate an averaged feature vector to be used during the ASR training or during normal operation parameters to determine their similarity with the training models.

System Training

The system training has the purpose to provide the ASR system the capacity to recognize isolated words depending on the speaker or independently of the speaker. In the first case the system will be trained with a corpus consisting of all words that the system must recognize spoken by all possible users of the system; in the second case, the system is trained with a corpus containing all words spoken by a specific speaker. In both cases it is desirable that each word be repeated several times by each speaker, if possible, during different days. Usually some part of the recorded speech signal is used for training and another part is used for testing.

The technique used for the training is chosen by the system designers and can be vector quantization (Barrón, Suárez-Guerra, & Moctezuma, 1999), neuronal networks, self-organized maps, Gaussian mixtures model, hidden Markov models, or combinations of the previous ones, with applications of vocalized segments, continuous speech, and so forth.

ASR Testing

Following the ASR training, system verification will be done using the N-M words not used for ASR training; the recognition success will be verified using these commands.

During the case of training, a question that must be considered is when we must stop the training process to avoid the overtraining phenomenon. It is understood like overtraining, the moment from which the system instead of reducing the error with respect to the testing set when the training time increase, the recognition error increases when the

training time increases. From the methodology point of view, to detect this moment of increase of the error of recognition in the system due to overtraining, it is recommendable to perform a system test at the end of each training session. Also it is recommend that, for the case of neuronal networks to initiate the training of the system with different weights from probability in the vectors that interconnect the cells of the system and to remain in the end with the network that provides the smaller error of trained recognition after.

CONCLUSION

This chapter presents an overview of the speech recognition technology, analyzing the main components of a speech recognition system. Topics such as signal capture, consideration that must be taken to construct a reliable corpus for system training, as well as different methods used for features extraction are analyzed in detail. This chapter complements Chapter XII provided in this book, related to advanced topics of speech and speaker recognition algorithms.

REFERENCES

Barrón, R., Suárez-Guerra, S., & Moctezuma, C. (1999). *Reconocimiento de comandos verbales utilizando cuantización vectorial y redes neuronales* (Tech. Rep. Serie Roja, No. 40). Computing Research Center, The National Polyechnic Institute of Mexico.

Bernal, J., Bobadilla, J., & Gomez, P. (2000). *Reconocimiento de voz y fonética acústica*. Madrid, España: Alfa-Omega.

Boulard, D. S. (1996). A new ASR approach based on independent processing and recombination of frequency bands. In *Proceedings of the International Conference on Spoken Language Processing* (ICSLP 1996) (Vol 1. pp. 426-429) ACM Digital Library.

Buzo, A., & Gray, R. (1980). Speech coding based upon quantization. *IEEE Transaction on Communications, COM-28*(1).

Chang, E., Zhou, J., Di, S., Huang, C., & Lee, K. (2000). Large vocabulary Mandarin speech recognition with different approaches in modeling tones. In *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China (pp. 983-986). ACM Digital Library.

Childers, D. G. (2000). *Speech processing and synthesis toolboxes*. New York: John Wiley & Sons.

Córdoba, R., Menéndez-Pidal, X., & Macías Guasara, J. (1995). Development and improvement of a real-time ASR system for isolated digits in Spanish over the telephone line. In *Proceedings of Eurospeech'95* (pp. 1537-1540). Madrid.

Fanty, M. (1996). *Overview of the CSLU toolkit*, (Tech. Rep. No. CSLU-011-1995). Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology.

Feal, L. (2000). *Sobre el uso de la sílaba como unidad de síntesis en el español* (Tech. Rep. No. IT-DI-2000-0004). Informatics Department, Universidad de Valladolid.

Fosler-Lussier, E., Greenberg, S., & Morgan, N. (1999). Incorporating contextual phonetics into automatic speech recognition. In *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition* (pp. 611-614). San Francisco.

Ganapathiraju, A., Hamaker, J., Picone, J., & Doddington, G. (2001). Syllable-based large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing, 9*(4), 358-366.

Harmut, R., Pfitzinger, S. B., & Heid, S. (1996) Syllable detection in read and spontaneous speech.

In *Proceedings of 4th International Conference on Spoken Language Processing,* Philadelphia (pp. 1261-1264). Washington, DC: IEEE Computer Society.

Hauenstein, A. (1996). *The syllable re-revisited* (Tech. Rep. No. tr-96-035). Munich, Germany: Siemens AG, Corporate Research and Development.

Hu, Z., Schalkwyk, J., Barnard, E., & Cole, R. (1996). Speech recognition using syllable-like units. In *Proceedings of the International Conference on Spoken Language Processing,* Philadelphia, PA, (vol. 2, pp. 1117-1120). Washington DC: IEEE Computer Society.

Jackson, L. B. (1996). *Digital filters and signal processing.* New York: Kluwer Academic Publishers.

Jones, R., Downey, S., & Mason, J. (1999). Continuous speech recognition using syllables. In *Proceedings of Eurospeech'96* (vol. 3, pp. 1171-1174). Rhodes, Greece.

King, S., Taylor, P., Frankel, J., & Richmond, K. (2000). Speech recognition via phonetically featured syllables. *PHONUS, 5,* 15-34.

Kirschning-Albers, I. (1998). *Automatic speech recognition with the parallel cascade Neural Network*. Unpublished doctoral dissertation, University of Tokushima, Japan.

Kita, K., Morimoto, T., & Sagayama, S. (1993). LR parsing with a category reachability test applied to speech recognition. *IEICE Trans. Information and Systems, E76-D*(1), 23-28.

Kosko, B. (1992). *Neural networks for signal processing.* Englewood Cliffs, NJ: Prentice Hall.

Lee, T., & Ching, P. (1998). A neural network based speech recognition system for isolated Cantonese syllables. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97) Hong Kong* (vol. 4, p. 3269). Washington DC: IEEE Computer Society.

Meneido, H., & Neto, J. (2000). Combination of acoustic models in continuous speech recognition hybrid systems. In *Proceedings of the International Conference on Spoken Language Processing,* Beijing, China (vol. 9, pp. 1000-1029). ACM Digital Library.

Meneido, H., Neto, P., & Almeida, L. (1999). Syllable onset detection applied to the Portuguese language. In *Proceedings of Eurospeech'99* (p. 81). Budapest, Hungry.

Montero, H., & Neto, J. (2000). Combination of acoustic models in continuous speech recognition hybrid systems. In *Proceedings of Eurospeech'99* (pp. 2099-2102). Budapest.

Munive, N., Vargas, A., Serridge, B., Cervantes, O., & Kirschnning, I. (1998). Entrenamiento de un reconocedor fonético de digitos para el español mexicano usando el CSLU toolkit. *Revista de Computación y Sistemas, 3*(2), 98-104.

Peskin, B., Gillick, L., & Liberman, N. (1991). Progress in recognizing conversational telephone speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP'97) Toronto, Canada (vol. 3, pp. 1811-1814). Washington, DC: IEEE Computer Society.

Rabiner, L., & Biing-Hwang, J. (1993). *Fundamentals of speech recognition.* Englewood Cliffs, NJ: Prentice Hall.

Rabiner, L. R., & Levinson, S. E. (1990). Isolated and connected word recognition-theory and selected applications. In A. Waibel & K. Lee (Eds.), *Readings in speech recognition* (pp. 115-153). New York: Morgan Kaufman Publishers.

Suárez-Guerra, S. (2005). ¿100% de reconocimiento de voz? Unpublished.

Sydral, A., Bennet, R., & Greenspan, S. (1995). *Applied speech technology.* New York: CRC.

Tebelskis, J. (1995). *Speech recognition using neural networks.* Unpublished doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Veeneman, D. E. (1988). Speech signal analysis. In C. H. Chen (Ed) *Signal processing handbook*. New York: Marcel Dekker, Inc.

Villing, R., Timoney, J., Ward, T., & Costello, J. (2004). Automatic blind syllable segmentation for continuous speech. In *Proceedings of the Irish Signals and Systems Conference 2004.* Belfast, UK (pp. 41-46). IET Digital Library.

Weber, K. (2000). Multiple timescale feature combination towards robust speech recognition. In *Konferenz zur Verarbeitung natürlicher Sprache KOVENS2000.* Ilmenau, Germany.

Williams, C. S. (1986). *Designing digital filters.* Englewood Cliffs, NJ: Prentice Hall.

Wu, S. (1998). *Incorporating information from syllable-length time scales into automatic speech recognition.* Unpublished doctoral dissertation, Berkeley University.

Wu, S., Shire, M., Greenberg, S., & Morgan, N. (1997). Integrating syllable boundary information into automatic speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Munich, Germany (vol. 1, pp. 987-990). Washington, DC: IEEE Computer Society.

Chapter VI
Seeking Patterns in the Forensic Analysis of Handwriting and Speech

Graham Leedham
Griffith University, Australia

Vladimir Pervouchine
University of New South Wales (Asia), Singapore

Haishan Zhong
Nanyang Technological University, Singapore

ABSTRACT

This chapter examines features of handwriting and speech and their effectiveness at determining whether the identity of a writer or speaker can be identified from his or her handwriting or speech. For handwriting, some of the subjective and qualitative features used by document examiners are investigated in a scientific and quantitative manner based on the analysis of three characters ("d," "y," and "f") and the grapheme "th." For speech, several frequently used features are compared for their strengths and weaknesses in distinguishing speakers. The results show that some features do have good discriminative power, while others are less effective. Acceptable performance can be obtained in many situations using these features. However, the effect of handwriting forgery/disguise or conscious speech imitation/alteration on these features is not investigated. New and more powerful features are needed in the future if high accuracy person identification can be achieved in the presence of disguise or forgery.

INTRODUCTION

In forensic analysis, there is often a need to identify a person or persons from evidence obtained at the scene of a crime. For example, evidence is often needed in criminal investigations to determine whether or not a piece of writing or a signature was written by a particular individual. Or it may

be necessary to produce evidence on the identity of the speaker in a recorded voice message left on an answering machine. Alternatively, such analysis may need to be applied to detect the identity of wanted individuals passing through security monitoring and screening operations in such sensitive areas as airport boarding gates and immigration desks.

In all these scenarios, it is necessary to accommodate the natural variations in handwriting and voice caused by environmental factors such as background noise and personal stress/emotional level, which affect both speech and handwriting, and the writing implement or physical position (e.g., sitting, standing, or writing while traveling in a vehicle), which affects the handwriting. In addition, writers or speakers may attempt to disguise their handwriting or voices to avoid detection. Or people may wish to try to pass themselves off as someone else by forging their handwriting so it looks like that of someone else, or they may adjust their voices or accents to imitate someone else.

When there is a need to seek the identity of an unknown speaker or writer, the existing techniques for such forensic analysis are carried out by experts who use their experience and knowledge of handwriting and speech to reach an expert decision. The techniques they use involve detailed examination of the handwriting or speech using various tools such as imaging devices and microscopes for handwriting, and spectral analysis and signal processing tools for speech analysis.

Generally, in forensic analysis, there is a questioned sample of handwriting or speech from an unknown person and other known samples of handwriting or speech from known individuals. These known samples are usually from suspects who may have uttered or written the questioned sample. It is the task of the forensic expert to reach an expert opinion, with supporting evidence, as to whether the unknown sample is the same or different from a known sample.

While forensic examiners use various computer tools to extract features and information from speech and handwriting samples, the final pattern matching or decision-making process to determine whether there is a match between the questioned sample and one of the known handwriting or speech samples is left to human judgment. This is not entirely without justification, as people are extremely capable of matching patterns and seeing similarities between samples, especially when they have become skilled in examining particular types of data. However, the techniques experts use are generally qualitative in nature, and there is little scientific basis on which to base their pattern-matching decision.

In this chapter, we describe the research work we and others have carried out to produce tools that assist forensic examiners in their pattern-matching process. The research seeks to identify and extract important and significant unique features of handwriting and speech and then automatically extract those features and apply automatic pattern-matching techniques to assist the experts in their forensic examination and provide a more scientific basis to support their expert opinion.

BACKGROUND

Handwriting and speech are personal biometrics that have long been considered unique to a person and his or her recognition used as evidence in court. However, the methods used to determine the authenticity of speech or handwriting and attributing it to an individual author have been increasingly questioned, as they are heavily reliant on qualitative opinion of expert witnesses and not always supported by quantitative scientific evidence.

For many years there has been considerable research into identifying what a speaker is saying (speech recognition) and, to a lesser extent, who the speaker is (speaker recognition). However, within the forensic context, differences in

recording equipment and transmission channels used, the presence of background noise, and voice variation due to differences in communication context (voice changes due to the environment and emotional state of the speaker) are a major challenge. Consequently, the impact of automatic speaker recognition technology in forensic analysis has been relatively modest, and forensic speaker identification remains dominated by the use of a variety of largely subjective procedures. Recent developments in the interpretation of the evidential value of forensic evidence favor methods that make it possible for results to be expressed in terms of a likelihood ratio. Traditional methods in the field of speaker identification generally do not meet this requirement. Conclusions in the form of a binary yes/no decision or a qualified statement from an expert witness of the probability of the hypothesis rather than the evidence are increasingly criticized for being logically flawed. Against this background, the need to put alternative scientifically supported validation procedures into place is becoming more widely accepted (Broeders, 2001).

Similarly, for many centuries, handwriting examination in the form of signature verification has been used for authentication purposes; in more recent years, considerable research has been carried out to automatically recognize handwriting. Experts in forensic document analysis throughout the world daily perform examinations of handwritten documents to determine the authorship of a questioned document or to detect evidence of forgery or disguise. The methods used by forensic document examiners are based on a set of established and well-documented techniques (Harrison, 1981; Hilton, 1993; Robertson 1991). Examiners look at features of handwriting that characterize shapes of letters, lines, and the document as a whole. The techniques have been derived from experience and are generally internationally accepted by various forensic laboratories. However, while they are intuitively reasonable, the methods of document analysis lack a scientific basis. Because of this, a fundamental question has arisen in several recent court cases querying whether the results of forensic document analysis are scientific and acceptable as evidence (*Daubert et al. vs. Merrell Dow Pharmaceuticals*, 1993; *United States vs. Starzecpyzel*, 1995). In order to establish the acceptability of forensics examination as a scientifically proven process, it is necessary to carry out a detailed examination of handwriting to determine (i) whether it is a biometric (i.e., does it exhibit the individuality of the writer) and (ii) whether the methods used by forensic document examiners can be supported by scientific study.

The remainder of this chapter describes research that examines pattern recognition and matching techniques by computer, which can be applied to the forensic analysis of both handwriting and speech.

FORENSIC FEATURES OF HANDWRITING

Introduction and Problem Statement

There are two types of features that can be extracted from images of handwriting: document examiner features and computational features. Computational features are features that can be automatically and readily measured by computer algorithms but do not necessarily have any visually perceivable meaning to humans. Normally, such features are extracted by applying various mathematical filters and transforms to the document image or its parts. The advantage of using computational features is that they are strictly defined and can usually be measured regardless of the content of a document. Extensive study of such features, their consistency, and discriminative power as well as their application to writer classification has been performed in the Center of Excellence for Document Analysis and Recognition (CEDAR) at the State University of New York at Buffalo.

In establishing the scientific basis for forensic analysis of handwriting, the first issue to solve is the questionable individuality of handwriting. Computational features have been shown sufficient to validate the individuality hypothesis (Srihari, Cha, Arora & Lee, 2002). However, many of the computational features are purely computational; they do not represent any features that forensic document examiners use in their methods. The features that document examiners extract to determine the authorship of documents under question are, however, entirely structural and visually discernable features. Huber and Headrick (1999) compiled a list of frequently used document examiner features into a set of 21 discriminating elements of handwriting.

If handwriting is indeed individual, then in order to establish the scientific basis for forensic document analysis, it is necessary to determine whether the current methods that forensic document examiners use do enable them to distinguish writers and accurately and correctly detect and measure the individuality in handwriting. Hence, it is important to determine whether writers can be distinguished using document examiner features of handwriting. Here arises a problem. Many of the 21 features in the list proposed by Huber and Headrick (1999) are defined quite ambiguously and can be very subjective in terms of their validity and usage. A possible solution is to formalize some of those features from the list; that is, to map them into other features that in turn are strictly defined and can be expressed in a numerical manner and automatically extracted and measured by computer algorithms. It is likely to be impossible to fully formalize all 21 features of handwriting used by forensic document examiners. Hence, reasonable assumptions are that a human expert can (i) effectively utilize more features of handwriting than a computer system and (ii) determine which features should and should not be used in a particular case (i.e., having looked at the handwritten samples under investigation, a human expert is able to intuitively select only the important features of the handwriting under examination). Lack of important features can degrade the performance of a pattern classification machine; presence of unimportant features can also degrade its performance. Consequently, it can be assumed that a human expert is able to distinguish writers or determine the authorship of questioned documents with higher accuracy than a computer system when only document examiner features are used.

The problem of validation of individuality of handwriting has been studied extensively in recent years (Srihari et al., 2002; Srihari, Tomai, Zhang & Lee, 2003; Zhang, Srihari & Lee, 2003). Issues regarding techniques used in the forensic analysis of handwriting, detection and measurement of features, comparison based on pictorial similarities between samples, and so forth have previously been discussed (Found & Rogers, 1995, 1998). It has also been demonstrated that professional document examiners are more accurate and correct in their analyses than lay people (Kam, Fielding, & Conn, 1997, 1998; Kam, Gummadidala, Fielding, & Conn, 2001; Kam, Wetstein, & Conn, 1994).

The work on handwriting analysis presented in this chapter focuses on the problem of validating the methods used by forensic document examiners. Such a study has received little attention previously. The objective of the current work is to formalize some of the document examiner features, particularly micro features; that is, we want to develop a means of measuring these features and expressing them in mathematical terms so they can be used with general pattern classification techniques and enable us to investigate how useful the features are for distinguishing different writers. According to the definitions of computational and document examiner features given in Srihari, Cha, and Lee (2001), the problem of document examiner feature formalization can be seen as a problem of mapping the document examiner features into computational features. However, in this work, the features obtained via

such mapping (all features under investigation) are still referred to as document examiner features in order to distinguish them from purely computational features such as GSC features (Govindaraju, Srihari & Shin, 1999).

Analysis of Forensic Document Features

Only unconstrained genuine handwriting was considered in the study reported here. The problem of authorship identification in court cases frequently involves forged and disguised handwriting. Such documents have not been considered in this study for two reasons. The first and main reason is that before studying complicated cases of deliberately changed handwriting, it is important to study the general case of people's normal handwriting. If the results achieved from the study are negative (i.e., the considered features cannot help distinguish writers based on their normal genuine handwriting), there is no point in applying the same approach to the more complicated cases of disguised or forged handwriting. The second and more practical reason is a lack of available data for forged and disguised handwriting for such studies.

It was decided to formalize some of the features from the "21 discriminating elements" (Huber & Headrick, 1999) and learn how to measure those features and extract them automatically, and evaluate their discriminating power by building a classifier. Since the shape of a character is usually affected by the preceding and following characters as well as the writing conditions such as type of pen and paper, writing constraints, and so forth, it is desirable to extract features of handwriting from samples written under similar conditions and having similar content so the influence of those factors is minimized. Handwriting samples from the CEDAR letter are suitable for this kind of study. The text of the letter was specially designed in the Center of Excellence for Document Analysis and Recognition at the State University of New-York at Buffalo for research purposes and contains each of the 26 English characters at the beginning of a word as a capital and as a small letter in the middle and at the end of a word. The database used in the current study consists of 8-bit grayscale images of 600 samples from 200 writers scanned at 300 dpi and separated from the background. It is part of the CEDAR handwriting database, which was kindly provided for this study by CEDAR.

Choice of Characters for Feature Extraction

According to Srihari, et al. (2001), there are two categories of features classified by the extraction level: macro features (up to the word level) and micro features (character level). The current study is focused on micro features because (i) they are thought to be better endowed with individual traits and are thought to be harder to change under attempts at forgery or disguise, and (ii) extraction of document examiner macro features has been investigated in other studies (Chong, 1996 ; Holcombe & Leedham, 1995; Leedham, Holcombe & Sagar, 1995).

It was necessary to make a decision on three important issues: (1) because the work is too large to undertake on all letters a subset had to be selected for feature extraction; (2) having selected the letters to study, it was necessary to select which features to extract from the chosen letters; and (3) it was necessary to resolve the issue of how to express the features numerically.

In order to decide which letters or letter combinations (graphemes) to use for micro feature extraction, several considerations were taken into account: (1) the letters and/or graphemes must be sufficiently frequent so a number of them can be found in most handwriting samples and hence make it possible to obtain the statistically reliable feature values; (2) according to Eldridge, Nimmo-Smith, Wing, and Totty (1984), letters with ascenders or descenders are particularly

useful for the purpose of writer identification; and (3) there are existing techniques for extraction of some features that can be used.

For the analysis of letter and grapheme frequencies, several books available in the public domain online libraries were used. The total number of words in these novels was about 220,000. The frequencies of occurrence were calculated for each letter as the number of occurrences of the letter normalized to the total number of letters. For grapheme analysis, only two-letter combinations were considered. Figures 1(a) and 1(b) display the observed frequency of occurrences of most frequent letters and graphemes correspondingly.

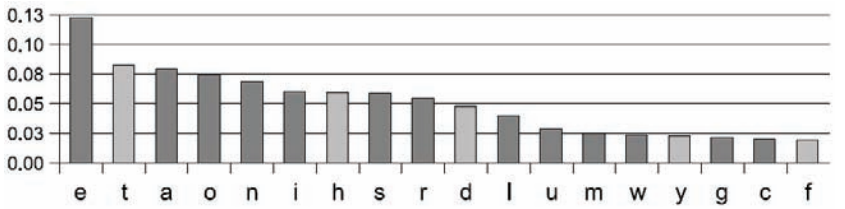
Based on these observations, a decision was made to choose four letters for feature extraction: "d," "y," "f," and the grapheme "th." It should be emphasized that the frequency of the letter and grapheme occurrence is limited to English. In other languages (e.g., German) the distribution is different, and other letters or graphemes may be preferred. The samples of the characters and grapheme were extracted manually from the CEDAR letter samples. In order to minimize the influence of the adjacent characters, it was decided to extract samples of "d" and "y" only from the end position in a word. Samples of the grapheme "th" were extracted from the starting positions.

Choice of Features to Study

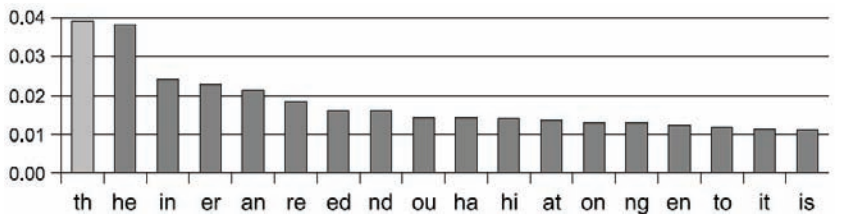
The selection of which features to extract from each character and grapheme was motivated by discussion of micro features in several books on forensic document analysis (Harrison, 1981; Hilton, 1993; Huber & Headrick, 1999). In particular, the list of 21 discriminating elements of handwriting was used as a reference. The list of features extracted in the study is shown in Tables 1(a) through 1(d). Each feature is denoted as f_i, where i is the unique feature index. Not all of the features were extracted at once; initially, only 21 features from the three characters and 10 features (f_{43}...f_{52}) from the grapheme were extracted. Some of the features of the grapheme "th" could not be extracted reliably when thinning-based skeletonization was used (f_{57}..f_{67}) and were extracted later when vector skeletonization was developed.

An important question that arises at this point is, how adequate are the feature extraction methods developed in this research for the purpose of

Figure 1. Character (a) and grapheme (b) frequencies in English text



(a). Character frequency in English texts.



(b). Grapheme frequency in English texts.

Table 1(a). Features extracted from "d"

Height	f_1
Width	f_2
Height to width ratio	f_3
Relative height of ascender	f_4
Slant of ascender	f_5
Final stroke angle	f_6
Fissure angle	f_7

Table 1(b). Features extracted from "y"

Height	f_8
Width	f_9
Height to width ratio	f_{10}
Relative height of descender	f_{11}
Descender loop completeness	f_{12}
Slant at point Y_T	f_{14}
Slant of descender	f_{15}
Final stroke angle	f_{16}

Table 1(c). Features extracted from "f"

Height	f_{19}
Width	f_{20}
Height to width ratio	f_{21}
Slant	f_{22}
Presence of loop at F_T	f_{27}
Presence of loop at F_B	f_{28}

Table 1(d). Features extracted from "th"

Height	f_{43}
Width	f_{44}
Height to width ratio	f_{45}
Distance *HC*	f_{46}
Distance *TC*	f_{47}
Distance *TH*	f_{48}
Angle between *TC* and *TH*	f_{49}
Slant of t	f_{50}
Slant of h	f_{51}
Position of t-bar	f_{52}
Connected/disconnected t and h	f_{53}
Average stroke width	f_{54}
Average stroke pseudo-pressure	f_{55}
Standard deviation of pseudo-pressure	f_{56}
Features extracted from vector skeleton only	
Standard deviation of stroke width	f_{57}
Number of strokes	f_{58}
Number of loops and retraced strokes	f_{59}
Straightness of t-stem	f_{60}
Straightness of t-bar	f_{61}
Straightness of h-stem	f_{62}
Presence of loop at top of t-stem	f_{63}
Presence of loop at top of h-stem	f_{64}
Maximum curvature of h-knee	f_{65}
Average curvature of h-knee	f_{66}
Relative size (diameter) of h-knee	f_{67}

the study. To answer this question, two types of features need to be clearly distinguished. One type of feature is the original document examiner feature selected from the 21 discriminating elements of handwriting (type 1). The problem with those features is their vague definition. If it is not possible to unambiguously define what the feature really is, it is not possible to extract it objectively and correctly. The other type of feature (type 2) is those features that were actually studied. These features were defined so their extraction was straightforward from their definition.

The purpose of the feature extraction was basically to mimic the extraction of features as

performed by document examiners; that is, to extract and study type 1 features from the list of 21 discriminating elements of handwriting. The mapping from features of type 1 to features of type 2 was the formalization of the document examiner features. The question "how suited were the extraction methods of type 2 features for extraction of type 1 features" is, hence, equivalent to the question "how do you evaluate the adequateness of the formalization." In general, it is not clear how to make such an evaluation. For some features, like dimensional features, the formalized features are exactly the same as the originals. For others, like line quality, it is not possible to provide a clear answer because the original feature is not clearly defined. How suitable is the set of features "straightness of t-bar, t-stem, and h-stem" to describe "quality of line?" In order to answer this question, one needs to understand what "quality of line" actually means, and that is not at all clear.

The feature set consists of geometrical characteristics of characters, various angular measures, loop characteristics, and stroke features. There are several binary features that denote presence or absence of some elements such as the t-bar and loops. Such features are assigned the value of 1 (TRUE) if the corresponding element is found in the character image and the value of 0 (FALSE) otherwise. Presence of point *D* in character "y" corresponds to the intersection of the descender and the base, as shown in Figure 2(c). The position of the t-bar feature in grapheme "th" is also a binary feature and is equal to 1 when the t-bar is crossing the stem and 0 in the cases of touching, detached, or absent t-bar.

Characters "d" and "y" were divided into two parts consisting of the base part and the ascender (descender). The border between the parts was defined by the upper point of the loop forming the base part of "d" and the lower point of the base part of "y" correspondingly. When no clear base part could be found, the relative height of the ascender (descender) became equal to 1. If the base part was detected, the feature value was calculated as a/f_1 for character "d" and $1\text{-}d/f_8$ for character "y."

For each character, its slant was measured. For characters "d," "h," and "y" slant was defined as the slant of the ascender (descender). For characters "f" and "t," the slant was defined as the slant of the stem. Such definitions of slant seem reasonable, as people usually make their judgements about the slants of these five characters by the slants of the major strokes of those characters.

A final stroke was defined as the angle between the tangent at the endpoint of the stroke and a horizontal line. Fissure angle for character "d" (f_7) was defined as the angle between the two tangents to the two strokes that form the fissure, as shown in Figure 2(a). Slant at point Y_T of character "y" was defined as the angle between the tangent line at point Y_T and a vertical line.

Loop area was approximated by the number of pixels inside the loop, and loop length was approximated by the number of border pixels. Loop slant was defined the same way as in the FOX system (Solihin, 1997). Loop completeness was defined through the length of the loop and the distance between the starting and ending points. If a loop is complete, the latter is 0.

The length of the descender loop of character "y" was calculated as the distance between the descender self-intersection point and the bottom-most or turning point of the descender.

Average stroke width was calculated in two ways: via the number of border pixels and the total number of black pixels in a binarized image and via tracing the strokes with a small step and evaluating the stroke width at each sample point using the distance map of the binarized image. The latter method could be used only with a vector skeleton. Although the difference in the resulting feature values was insignificant, the latter method also enabled extraction of the standard deviation of the stroke width (f_{57}). Pseudo-pressure was calculated by averaging the gray values of all the pixels that form the strokes.

Figure 2. Illustration of some of the structural features extracted from the characters/grapheme studies

Figure 2(a). Features of "d"

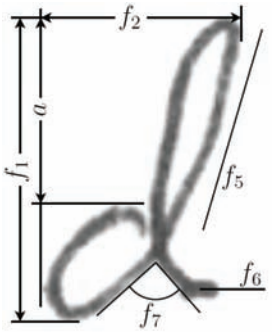


Figure 2(b). Features of "y"

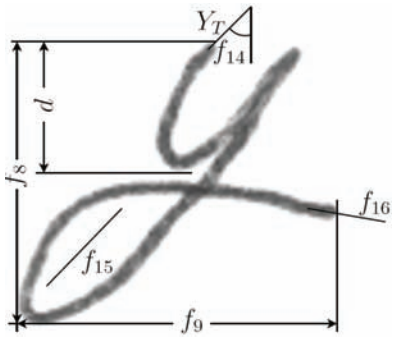


Figure 2(c). Features of "y"

Figure 2(d). Features of "f"

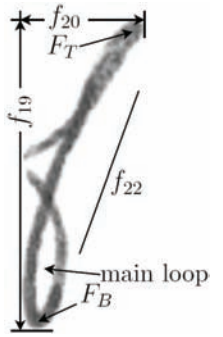
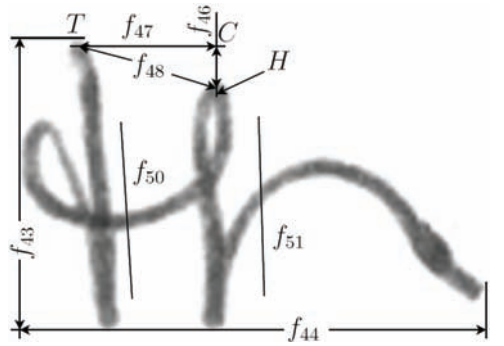


Figure 2(e). Features of "th"



Straightness of t-stem and h-stem (ascender) were defined as the ratio of the length of the curve to the distance between its endpoints. Maximum and average curvatures of the h-knee as well as the relative size (diameter) of the h-knee were calculated in a straightforward way from the B-spline representation of the corresponding stroke.

Definitions of other features can be derived from Figures 2(a)2(e).

Extraction of Features

Initial Image Preprocessing

The automatic feature extraction engine requires the gray-scale image of a character or grapheme sample, the binarized version of this image, and the skeleton of it. Binarization of the images was

straightforward, as the handwriting in the CEDAR samples was on clean white paper. A constant global grey-level threshold of 250 was used, and a pixel was marked as white if its value exceeded the threshold and black otherwise.

In the initial experiments involving the extraction of document examiner features, a skeletonization method based on thinning was used. The thinning-based skeletonization method was developed using the basic thinning function provided in the Matlab Image Processing Toolbox (Guo & Hall, 1989), which is a modified version of the Zhang and Suen (1984) thinning algorithm mther thinning methods were also tried (Suen & Wang, 1994), but no significant improvement in feature extraction was observed. The thinning algorithm was applied to a character image, and then correction of some artefacts produced by the thinning process was performed, as shown in the following pseudo-code:

```
do {
    remove small connected components
    find junction points
    find endpoints
    correct spurious loops
    prune short branches
} while there are some changes in the skeleton image
```

Figure 3(c) shows some artefacts that were removed. Artefact (1) is a spurious loop; Artefact (2) is a small connected component; Artefacts (3) are extra branches.

In later experiments, a new improved skeletonization algorithm was developed that was able to preserve the original junction points and approximate the original handwriting strokes with smooth B-spline curves. The resulting skeleton was in a vector form (a set of B-splines). The feature extraction algorithms were changed so they could extract features from the new skeletons (Pervouchine, Leedham & Melikhov, 2005a, 2005b).

Figure 3. Example images used to extract features of the handwriting from raster skeletons. (a) original, (b) binarized, (c) thinned, and (d) corrected image

Figure 3(a)

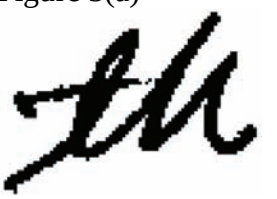


Figure 3(b)

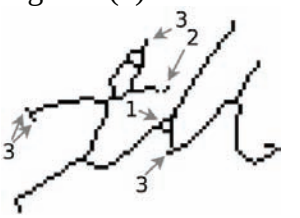


Figure 3(c)

Figure 3(d)

Extraction of Skeleton-Independent Features

Several features were extracted from either the original or the binarized image of handwriting samples. Height, width, and height-to-width ratio were measured from the binarized image by determining the bounding box of the image. The bounding box coordinates x_1, y_1, x_2, y_2, correspond to the topmost, leftmost, bottommost, and rightmost black pixels on the image correspondingly. The feature values were calculated as:

$height = y_2 - y_1 + 1$, $width = x_2 - x_1 + 1$,
$ratio = height/width$

Pseudo-pressure was estimated from the grey levels of the image pixels. Let the set of the foreground pixels be S and the intensity of a pixel be:

$$I(x,y),\ black = 0 \le I(x,y) \le 1 = white$$

The average pseudo-pressure was calculated as:

$$\langle p \rangle = \frac{1}{N_S} \sum_{i:(x_i,y_i)\in S} I(x_i, y_i)$$

And the standard deviation of the pseudo-pressure was calculated as:

$$\sigma_p = \sqrt{\frac{1}{N_S - 1} \sum_{i:(x_i,y_i)\in S} \left(I(x_i, y_i) - \langle p \rangle\right)^2}$$

Extraction of Angular Features

Slant was calculated in two ways for raster and vector skeletons. In both cases, a set of strokes or skeletal branches was first located, which represents the element from which the slant feature was extracted. These elements were ascender for character "d," descender for "y," stems for "t," "f," "h," and so forth. Then, for the raster skeleton, the pixels that belong to the located set of branches were taken, and a regression line $x = ky + b$ was fitted to that set of points, as shown in Figures 4(a) and 4(b). The choice of the fitting line is explained by the fact that slant is normally closer to a vertical line than to a horizontal line. If the set of points to which a line is to be fitted is denoted by (x_i, y_i), $i = 1...N$, slant was calculated as:

$$slant = \arctan \frac{N \sum_{i=1}^{N} x_i y_i - \sum_i x_i \sum_i y_i}{N \sum_i y_i^2 - \left(\sum_i y_i\right)^2}$$

For the vector skeleton, the slant value was calculated by taking a set of sample points s_i along each spline-approximated stroke that represented the element of interest (ascender, descender, etc.) and calculating the angles of tangents at these points $\alpha_i = \arctan k_i$. The slant was calculated as the weighted average of those angles:

$$\alpha_{slant} = \frac{\sum_i l_i \alpha_i}{\sum_i l_i}$$

where l_i is the length of the corresponding curve segment (see Figure 4 (c)).

For the case of the vector skeleton, the final stroke angle was calculated as the angle of the tangent to the corresponding endpoint. For the case of the raster skeleton, a straight line was initially fitted into a set of pixels, but later that was changed to fitting an ellipse into a set of pixels representing the final stroke, and the tangent to the ellipse at the point nearest to the endpoint of the stroke was taken for calculation (Figure 5(a)). In these cases, ellipse fitting algorithm produced high residual

Figure 4. Calculating the slant of a stroke

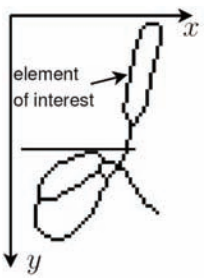


Figure 4(a). Element of interest.

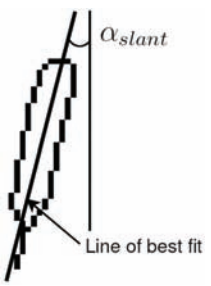


Figure 4(b). Fitting a line of best fit.

Figure 5(a). Final stroke angle

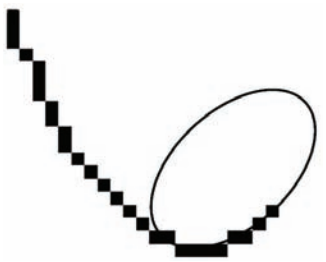


Figure 5(b). Fissure angle

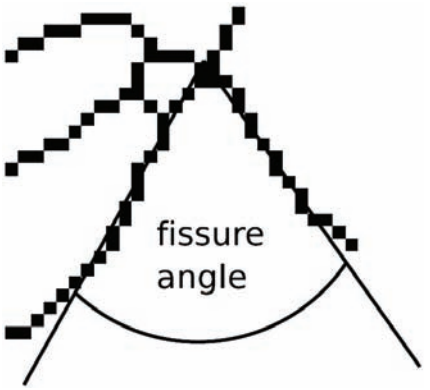


Figure 5(c). Slant from vector skeleton

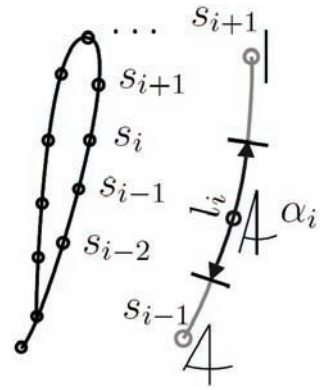
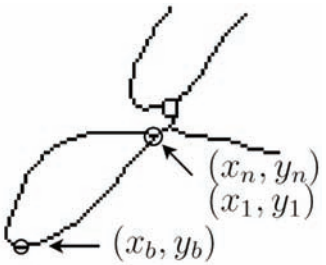


Figure 6. Extraction of descender features



error. To correct this, a straight line was fitted into the set of pixels instead of an ellipse.

Fissure angle measurement was performed similarly to the final stroke measurement. Once the fissure point was located, extraction of the fissure angle in the case of a vector skeleton was straightforward. In the raster skeleton case, two regression lines were fitted, each one approximating the tangent line to the corresponding stroke (see Figure 5(c)). The angle between the regression lines was taken as the fissure angle.

Extraction of Ascender and Descender Features

The descender of "y" was detected by tracing it in both directions from the bottommost black pixel of the skeletonized image, which always belonged to the descender. After tracing of the descender, all the visited pixels were erased and the horizontal distance from the image left border to the leftmost black pixel was calculated. The horizontal level at which the distance function had a sharp increase was taken as the bottommost lowest base level. In cases when no sharp increase was observed, the bottommost black pixel was taken as the lowest base level. If the ordinate of this pixel was y_{base}, the relative height of descender was calculated as $f_{11} = (y_b - y_{base})/ f_8$, where f_8 was the character height feature (Figure 6).

Due to the frequent presence of spurious loops resulting from the skeletonization process or small loops in the ascender of "d," tracing of the ascender part of the skeleton proved to be not as robust as it was for the descender of "y." In order to find the horizontal level that corresponded to the top of the base, two algorithms were used. In the first algorithm, the distance between the left edge of the image and the leftmost black pixel was calculated the same way as for the base part of "y." Sharp increases in the distance function value were marked and sorted according to their increase value. The largest increase was compared to a threshold value. If its magnitude was more

than the threshold, ordinate *y* that corresponded to that increase was taken to be the top of the base, as shown in Figure 7(a). If none of the increases was larger than the threshold, the second detection algorithm was run. The value of the threshold was determined empirically. The second algorithm extracted the width of the character skeleton as the function of *y* (see Figure 7(b)). The value of *y* at which the resulting function *width*(*y*) reached its minimum was taken to be the top of the base. In cases when there were several minima, the lowest one was taken.

Extraction of Other Features

Detection of the top points was made by first detecting all the endpoints in the upper half of a sample skeleton image and then tracing the branches from those endpoints to determine which of them correspond to which elements of "th." Cases of "t" and "h" sharing the top stem point were taken into account. Once the top of the stem points was detected, extraction of the related distance features was simple.

The number of strokes and number of loops and retraced strokes were available directly from the skeleton. Since each retraced stroke can be considered a small loop or a hidden loop, the number of loops, the number of hidden restored loops, and the number of retraced strokes were summed to give the feature value.

Standard deviation of stroke width was extracted by taking a set of sampling points on the skeleton curves with a small step and measuring the cross-section of the strokes on the underlying binarized image as shown in Figure 8. The obtained measurements were then used to calculate the feature value.

The presence of loops at the top of t-stem and h-stem could also be extracted from the raster skeleton. However, experiments on extraction of similar features from character "f" showed that extraction of these features from the skeleton produced by the thinning-based method was unreliable due to spurious loops introduced by the thinning process. Extraction of these features from a vector skeleton was straightforward once the corresponding endpoints were determined.

Straightness of a stroke (t-bar, t-stem, h-stem) was calculated in a similar manner to the descender

Figure 7. Extraction of the height of the ascender in character "d"

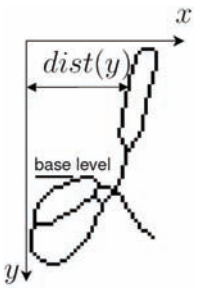


Figure 7(a). Relative height of ascender, first algorithm.

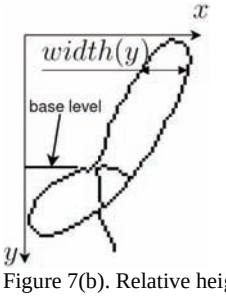
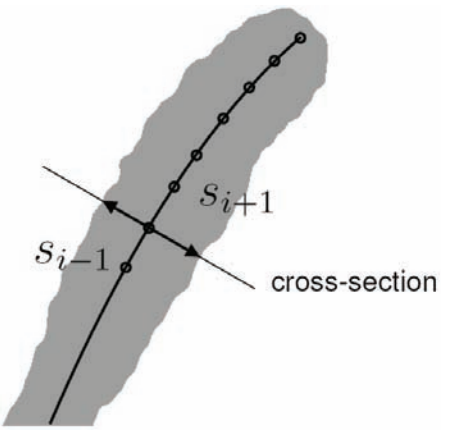


Figure 7(b). Relative height of ascender, second algorithm.

Figure 8. Stroke cross-section to estimate stroke width



of "y" completeness feature. If the distance between the two endpoints was *d* and the length of the stroke was *L*, the straightness of the stroke was given by *straightness* = *L* / *d*. It was close to unity for a straight stroke and significantly larger for a curved stroke.

Maximum and average curvature of a stroke (h-knee) was calculated by taking sample points at small steps along the curve, calculating the curvature at each point and taking the largest value and the weighted average.

Relative size of h-knee was calculated as the largest distance from h-stem to the h-knee curve. It was approximately calculated as the largest horizontal distance between the curves representing the stem and the knee.

Evaluation of the Features

Features of "d," "y," "f," and "th"

Suppose there is a set of features and all subsets are found that are equally good for writer classification. Equally good means that the classification accuracies achieved when those subsets are used do not differ significantly—their average values are indistinguishable. There are three classes of features according to their inclusion in the found feature sets: some features are included in all feature sets, some are not included at all, and the rest are included in some of the sets. The first category of features comprises indispensable features, the second category contains irrelevant features, and the rest of the features are partially relevant.

For each feature subset, its performance was measured in a series of experiments involving writer classification, producing the average accuracy of classification along with its standard deviation. The experiments used n-fold cross-validation (Weiss & Kulikowski, 1991). The data were divided into *n* approximately equal parts, and the classification experiments were then performed *n* times. Each time, another part of the data was taken out. The remaining *n* - 1 parts were used to train the classifier, and the other part was used for testing. The *n* values of the classification accuracy obtained were averaged.

A DistAl neural network was used as a classifier (Yang, Parekh & Honavar, 1997), and the fivefold cross-validation method was used to estimate the writer classification accuracy. The total number of different feature subsets that can be formed from a set of *M* features is $2^M - 1$. When this number is low (e.g., for individual character features), an exhaustive search is possible. When the number becomes large, an exhaustive search is

Table 2. Optimal feature sets, accuracy values, and standard deviations; bit 1 corresponds to presence of the feature in the subset, bit 0 to absence of it

'd'-part	'y'-part	'f'-part	'th'-part	accuracy	$\sigma_{accuracy}$
$f_1 \dots f_7$	$f_8 \dots f_{12}\ f_{14} \dots f_{16}\ f_{18}$	$f_{19} \dots f_{22}\ f_{27}\ f_{28}$	$f_{43} \dots f_{52}$		
1111100	01111001	111001	1111111111	0.58	0.04
1101100	10111011	111001	1111111111	0.57	0.04
1110100	11011010	111001	1111111111	0.55	0.04
1111100	11111001	111001	1101111111	0.54	0.05
1110100	11111101	101000	1111111111	0.54	0.05
1111000	11111010	111001	1111111111	0.53	0.04
1111100	11111001	111001	1111111111	0.53	0.04
1111100	11110011	111001	1011111111	0.53	0.04

not feasible. In this case, it was decided to employ a genetic algorithm for the search.

Table 2 shows the feature subsets of the entire four-character feature set that gave the same highest classification accuracy. The accuracy values presented in the table are indistinguishable from each other at the 1% significance level.

Division of the feature set into three categories of indispensable, partially relevant, and irrelevant features was performed according to the results presented in Table 2. As seen from the results, the highest number of indispensable features belongs to grapheme "th."

Dependence of the classification accuracy on the number of writers whose samples needed to be classified was measured on best feature subsets of individual character feature sets for "d," "y," "f," and "th," and on the best subsets of the feature sets of two-, three-, and four-character combinations. Figure 9 shows the degradation of the classifier performance due to the increasing number of writers. For two- and three-character combinations, the accuracies obtained on various combinations were averaged. As seen from Figure 9, inclusion of more features from more characters into the feature set used for writer classification both increases the classification accuracy and makes the decrease of the performance less rapid. It is worth noting that there is little difference in the classification accuracy between feature sets of three-character and four-character combinations. This resulted from the fact that inclusion of features of grapheme "th" made the most significant contribution to the performance of the classifier compared to inclusion of features of any of the three single characters.

Features of "th" with Vector Skeletonization

An optimal feature subset search was performed using a GA with sharing in the same way as was performed for feature subsets of the three characters and one grapheme. The feature data extracted from samples of 165 writers using vector skeletonization were used. The resulting feature subsets that gave the highest classification accuracy are shown in Table 3. The accuracy values presented in the table are indistinguishable from each other at the 1% significance level.

Figure 9. Degradation of writer classification accuracy with an increasing number of writers when features of different characters/grapheme are included in the feature set

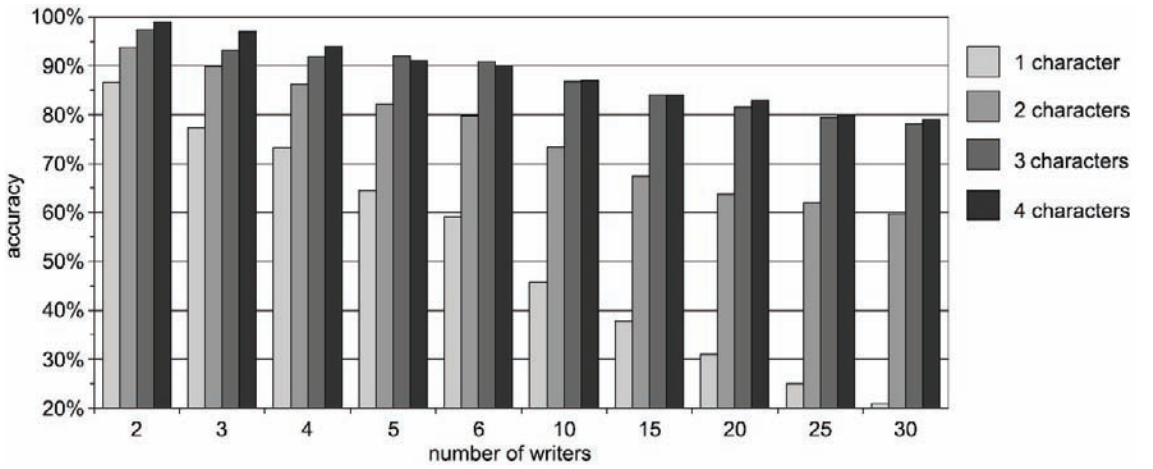


Table 3. Optimal feature subsets of "th" feature set, accuracy values, and standard deviations; bit 1 corresponds to presence of the feature in the subset, bit 0 to absence of it; features are divided into groups of five for convenience

feature set $f_{43} \ldots f_{67}$	accuracy	$\sigma_{accuracy}$
11111 11111 11000 11101 11111	0.67	0.04
10111 11111 11000 11101 01111	0.67	0.04
11111 11111 11100 11101 01111	0.65	0.05
11011 11111 11100 11111 11111	0.64	0.04
11111 11111 11100 11101 01111	0.64	0.04
11111 11111 11000 11101 00111	0.64	0.04

CONCLUSION

It is necessary to note that the features for which relevance was assessed were not exact document examiner features but rather a quantifiable representation of them (formalization). That is why, for example, it is incorrect to say that the final stroke shape of "d" has no discriminative power. Rather, the final stroke angle measured as the tangent angle to its endpoint is not useful for writer classification. It is possible that the current formalization is not sufficiently descriptive, and another formalization of a final stroke shape would change this situation. The same applies to the fissure of character "d." The main purpose of using features that represented loops at the top and bottom points of the f-stem as well as at point Y_T of "y" was to distinguish between the hand-printed and cursive forms of characters. From the results obtained, it is concluded that these features do not help discriminate effectively between the two character forms.

It was shown that a number of the considered document examiner (structural) features indeed possess discriminating power, and thus, use of these features for the purpose of writer identification is justified. It was demonstrated that different characters have different discriminative powers (Figure 9), and there exists a noticeable difference in discriminative power between characters and graphemes. Use of features of grapheme "th" resulted in significantly more accurate identification of writers than the use of any of the features of the three single characters. This supports the suggestion often made in forensic document examination that the shape of a character can (and usually is) greatly affected by its adjacent characters.

The most important (indispensable), partially relevant, and irrelevant features were identified under the assumption that the data were all genuine unconstrained handwriting. The identification of indispensable features provides the information about which features should be given priority in handwriting analysis when the aim is to identify writers. It is possible, however, that under conditions different from those used in the current study (normal unconstrained handwriting), the distribution of the features into the three categories according to their relevance may be different. For example, height-to-width ratio of a character or grapheme is thought to be a more useful feature than both height and width when handwriting is constrained, as in cases where handwriting is extracted from forms. Also when grapheme "th" is not extracted from the beginning of words, the presence of a loop at the top of the t-stem could possibly be more often included in the best feature subsets.

FORENSIC STUDY OF SPEECH FEATURES

Speaker Verification in Forensic Analysis

Speaker verification is part of a general speaker recognition system. It is a technique to verify or reject the claimed speaker's identification by analyzing the features of the speech from the speaker (as opposed to attempting to recognize the words spoken) and comparing them against known samples of the speaker for whom verification is required. This enables voice to be used to identify a person for user authentication purposes (Kunzel, 1994).

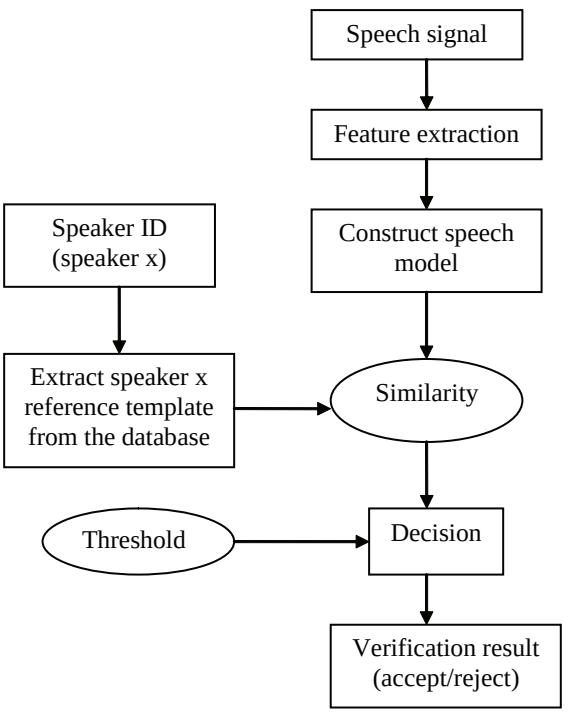
The speech of humans contains rich information such as emotion, identity, content, and language. Using speech as a biometric enables a simple and natural form of identification to take place. For example, in banking transactions over the telephone or other networks, speaker verification can help determine whether it is the correct or authentic person asking for the transaction service. Other biometric applications such as secure access controlled by voice also find speaker verification quite helpful.

In order to teach a machine to distinguish people by voice as precisely as the trained human ear, research into speaker verification involves many techniques such as pattern recognition, feature extraction, and classification. Features that are extracted from the speech signal are important for the accuracy of the verification system. Although the features contain rich information, only some of the information is valuable for identifying speakers. The rest of the features may not be helpful and may even disturb the speaker verification process and degrade its performance. It is therefore important to extract effective features for speaker recognition. The following sections discuss further investigations that have been performed using various kinds of speech features to determine which features are most useful for speaker identification.

Structure of a Speaker Verification System

Figure 10 shows the basic structure of a speaker verification system. Each speaker in the database has a model or reference template that represents their speech. When there is a speaker who claims he or she is speaker *A* in the database, the template of speaker *A* will be picked out of the database and sent to the compare block. The speech from the claimed speaker will be processed, features will be extracted, and a model of the speech will be formed. By comparing the similarity of the two models, the reference template for speaker *A* and the model of the incoming speech, a decision will be made according to similarity or matching threshold. The result of the speaker verification system is only acceptance or rejection. The complete performance of the verification system has no relationship to the number of speakers

Figure 10. Structure of a speaker verification system



in the database, as it is a one-to-one and not a one-to-many match. Therefore, as the number of speakers in the database rises, the performance of the verification system remains constant. The accuracy is a function of the uniqueness of the speech features extracted and the similarity measure employed.

For forensic purposes, a soft decision is required that provides a probabilistic measure that the speaker is who he or she claims to be (Doddington, 1985). To achieve this, the value is not compared to the threshold but rather mapped into the probability value. Establishing the mapping is called calibration (Brummer & du Preez, 2006; Campbell, Brady, Campbell, Granville & Reynolds, 2006; Campbell, Reynolds, Campbell & Brady, 2005).

Features for Speaker Verification

In order to compare the similarity and make a reliable acceptance/rejection decision, all the models/references in the system must be built using the same kind of speech features.

Speech is a complicated signal and can be analyzed at several levels such as the semantic, linguistic, or acoustic level. The transformations of the signal can be used as features that can distinguish differences in the acoustic properties of the speech signal. For a speaker verification system, features that have speaker-dependent differences are more of interest than speaker-independent features. Speaker-dependent features are the result of a combination of anatomical differences inherent in the vocal tract and the learned speaking habits of different individuals (Campbell, 1997). By building statistical models or references that approximate the distribution of these feature vectors for different speakers, the conditional probability of the speaker matching the template of who he or she claims to be can be estimated.

Among all the features that can be extracted from speech, the speech spectrum has been shown to be very effective for speaker identification (Campbell, 1997). This is because the spectrum reflects a person's vocal tract structure, the predominant physiological factor which distinguishes one person's voice from others. Linear Prediction cepstral and reflection coefficients have been used extensively for speaker recognition.

The popular features that are currently being used for speaker identification are acoustic features (e.g., MFCC [mel frequency cepstral coefficients], pitch, LSP [line spectrum pairs], and LPC [linear prediction coefficients]). They are regarded as robust features, but they are not perfect as they contain both speech and speaker information. The presence of speech information can disturb the speaker recognition system and cause accuracy to degrade. Experiments have shown that no single feature achieves high accuracy in speaker recognition. Therefore, a detailed study is needed of efficient features for speaker verification that eliminate the features representing the speech information but retain the features containing the speaker information.

In the following sections, a number of novel acoustic features proposed recently are discussed and compared. Features studied are Mel linear spectral frequencies (MLSF), Hurst parameter related features (pH), linear prediction residual phase, and features based on fractional Fourier transform (MECB, DMECB). All these features are compared with the classic MFCC features to provide a baseline of performance.

Frame-Level Analysis

Extraction of all features discussed here is based on dividing a speech signal into short frames of 20 to 80 milliseconds duration. The frames are obtained using a Hamming window, and one feature vector is obtained for each frame. Thus, the distribution of the feature vectors obtained from the speech signal represents the speaker in the feature space.

Mel Frequency Cepstral Coefficients (MFCC)

MFCCs are popular acoustic features and have been demonstrated to work well for both speaker recognition and speech recognition.

The first step in extracting these features is to perform a fast Fourier transform (FFT) of the speech frame. The second step is to take the magnitude. The third step is to warp the frequencies according to the mel scale. The mel scale is based on the nonlinear human perception of the frequency of the sounds. Therefore, the warping transforms the frequency scale to place less emphasis on high frequencies. In this step, by setting the mel scale window number, the number of MFCC features is determined. The fourth step is to take the logarithmic value of the mel scale result. The fifth and final step is to take the inverse FFT.

Difference feature vectors calculated between the vectors of the adjacent frames are called ΔMFCC (first difference), and the difference vectors calculated in the similar manner using the ΔMFCC are called ΔΔMFCC (second difference). Both the first and second difference features are used together with MFCC because they capture short-term speech dynamics in a time interval of 50 to 100 milliseconds. It is thought that short-term speech dynamics can additionally characterize a person's vocal tract (Oppenheim & Schafer, 2004). However, this interval does not capture longer-term features such as prosodic gestures and syllable usage.

The feature vectors are normalized by subtracting the mean vector and dividing each vector component by its standard deviation.

Mel Linear Spectrum Frequencies (MLSF)

Mel linear spectrum frequencies are similar to linear spectrum frequencies (LSP), calculated from linear prediction (LP) coefficients. To overcome the drawback of the line spectrum frequencies features in that they do not take advantage of the properties of the human ear, such as using mel filter bank to reduce the information in high frequencies, MLSFs are computed from the mel-spectrum energies (Cordeiro & Ribeiro, 2006).

Fast Fourier transform (FFT) and mel filter banks were used to generate the mel spectrum. Then the inverse Fourier transform was applied to calculate the mel autocorrelation of the signal. The MLSF features were then calculated using the Levinson-Durbin recursion. A linear prediction filter of order 16 was used, resulting in 16-dimensional feature vectors. The feature values were normalized by subtracting the mean and dividing by the standard deviation. Since addition of first and second differences (ΔMLSF and ΔΔMLSF) may increase the speaker verification accuracy, both differences were calculated.

Residual Phase

Extraction of most acoustic features is based on a model of a person's vocal tract that consists of an excitation source and a number of linear filters that approximate the vocal tract shape (Campbell, 1999). If the vocal tract-transfer function of the speech production model can be characterized by the predictive coefficients, the prediction error, referred to as the LP (linear prediction) residue, will characterize the excitation signal (Zheng & Ching, 2004). Thus, the LP residue can be used to derive the source information that contains additional information useful for speaker recognition.

One way to extract the speaker-related information is to extract the phase information as proposed by Murty and Yegnanarayana (2006). If the residual signal is $r_n = s_n + \sum_k a_k s_{n-k}$, the analytic signal is given by:

$$R_n = r_n + jh_n ,$$

where h_n is the Hilbert transform of r_n. The magnitude of the analytic signal R_n is $|R_n| = \sqrt{r_n^2 + h_n^2}$, and the phase θ_n is calculated as:

$$\theta_n = \arccos \frac{r(n)}{\sqrt{r_n^2 + h_n^2}}$$

The phase is calculated from short frames of speech of around 5 milliseconds duration, which is approximately the period of the bursts in the excitation source.

Hurst Parameter Features (pH)

Hurst parameter features have been proposed for speaker recognition problem by Sant'Ana, Coelho, and Alcaim (2006). The statistical feature pH is a vector of Hurst parameters obtained from the windowed short-time segments of speech. It expresses the time-dependence of a stochastic process. Since it models the stochastic behavior of the speech signal, it is robust to channel distortion. Because it is not related to the transfer function of the vocal tract, the extraction methods of pH are less complex, and pH can be extracted in real time.

The Hurst parameter is defined by the decaying rate of the autocorrelation coefficient function (ACF): $\rho(k)$ ($-1 < \rho(k) < 1$) as $k \to \infty$. If the speech signal is s_n with the finite variance, the autocorrelation coefficient is:

$$\rho(k) = \frac{\mathrm{cov}[s_n, s_{n+k}]}{\mathrm{var}[s_n]}, k = 0,1,2,...$$

where $-1 \le \rho(k) \le 1$ and $\lim_{k->\infty} \rho(k) = 0$.

The asymptotic behavior of $\rho(k)$ is be given by:

$$\rho(k) \sim H(2H-1)k^{2(H-2)}$$

The feature vector is a vector of Hurst parameters H calculated for frames of a speech signal via Abry-Veitch estimator using discrete wavelet transform (Veith & Abry, 1994). Daubechies wavelets with four, six, and 12 coefficients were used, thus resulting in pH_4, pH_6, and pH_{12} features correspondingly.

Features Based on Fractional Fourier Transform

The fractional Fourier transform is a generalization of the Fourier transform. If the conventional Fourier transform is F^n, where n can only be an integer ($n = 1$ for the direct transform, $n = -1$ for the inverse transform, etc.), the fractional Fourier transform can be thought as F^p, where p is a real number. Thus, it is said to transform a function to an intermediate domain between time and frequency (Almeida, 1994).

Box 1.

$$F^a[s(x_1)] = s(x) = \frac{\exp j(\frac{\pi}{4} - \frac{\pi}{2})}{\sqrt{2\pi \sin\alpha}} \exp(-\frac{j}{2}x^2 \cot(\alpha)) \int_{\infty}^{-\infty} \exp(-\frac{j}{2}x_1^2 \cot(\alpha) - \frac{jx_1 x}{\sin\alpha}) s(x_1) dx_1$$

Box 2.

$$F^{-a}[s(x_1)] = \frac{\exp -j(\frac{\pi}{4} - \frac{\pi}{2})}{\sqrt{2\pi \sin\alpha}} \exp(+\frac{j}{2}x^2 \cot(\alpha)) \int_{-\infty}^{\infty} \exp(+\frac{j}{2}x_1^2 \cot(\alpha) - \frac{jx_1 x}{\sin\alpha}) s(x_1) dx_1$$

where $\alpha = a\pi/2$

The fractional Fourier transform is given by Box 1, and the inverse fractional Fourier transform by Box 2.

Mean energy within critical bands (MECB) features based on the fractional Fourier transform have been proposed by Wang and Wang (2005) for speaker recognition and evaluated on a custom dataset. The critical bands are formed by warping frequency according to mel or bark scale. $MECB_p$ are calculated by taking the fractional Fourier transform of order p of each frame of the signal and then calculating the logarithm of the energy within bands. For the i-th critical band $f_i...f_{i+1}$, the mean energy is:

$$E_i = \frac{\int_{f_i}^{f_{i+1}} |F(f)|^2 \, df}{f_{i+1} - f_i}$$

For two MECB features of orders p_1 and p_2, the difference MECB (DMECB) features are calculated as:

$$DMECB_{p1-p2} = \log\left|\exp(MECB_{p1}) - \exp(MECB_{p2})\right|$$

Techniques for Speaker Recognition

Beside the speech features, the other aspect that affects the similarity comparison is the technique used to build up models or reference templates. The techniques for speaker recognition can be categorized into three major approaches.

The first approach is to use long-term averages of acoustic features such as spectrum representations or pitch. It is also the earliest way researchers used to extract information from a speech waveform. Through averaging out other factors influencing the acoustic features, such as phonetic variations, the speaker-dependent components remain. For spectral features, the long-term average represents a speaker's average vocal tract shape. However, the averaging process lowers the speaker-dependent information, and it is not stable in the long term. The stable long-term statistics can only be derived from analysis of long periods of speech.

The second and most frequently used approach is to build a statistical model of each speaker using speaker-dependent acoustic features that represent the distribution of the feature vectors for each particular speaker. By comparing the acoustic features from phonetic sounds in a test utterance with speaker-dependent acoustic features from similar phonetic sounds, the comparison measures speaker differences rather than textual differences. These probabilistic models are used in speaker recognition, and experiments show that the model of acoustic speech events and a framework is good for dealing with noise and channel degradations.

There are several methods of building models for speakers. One such model is vector quantization (VQ). For VQ, each speaker is represented by a codebook of spectral templates representing the phonetic sound clustered in speech. But the VQ is limited in its ability to model the possible variability encountered in an unconstrained speech task, although this technique has good performance on limited vocabulary tasks.

Hidden Markov modeling (HMM) is another popular technique that is used for probabilistic speaker models for speaker recognition (Matsui & Furui, 1994; Poritz, 1982). The HMM models are not only for the underlying speech sounds, but also for the temporal sequencing among these sounds. Although temporal structure modeling is advantageous for text-dependent tasks, for text-independent tasks, the sequencing of sounds found in the training data does not necessarily reflect the sound sequences found in the testing data and contains little speaker-dependent information.

The Gaussian mixture model (GMM) (Reynolds & Rose, 1995) is also used in speaker recognition. The GMM attempts to build a probabilistic model of the underlying sounds of a person's voice, and does not impose any Markovian constraints between the sound classes. After you have well-trained models of speakers, the models can be readily implemented in a real-time speaker recognition application, as it is computationally efficient.

The third approach to speaker recognition is to use a kernel-based learning method, such as discriminative neural networks (NN) or a support vector machine (SVM). Rather than training individual models to represent particular speakers, SVMs are trained to determine the best decision function plane that discriminates speakers within a known set. For NN, this requires a smaller number of parameters but produces better speaker recognition performance than the idea of building speaker models such as VQ systems. But there is a major drawback to using NN in that when a new speaker is added to the system, the complete network must be retrained again.

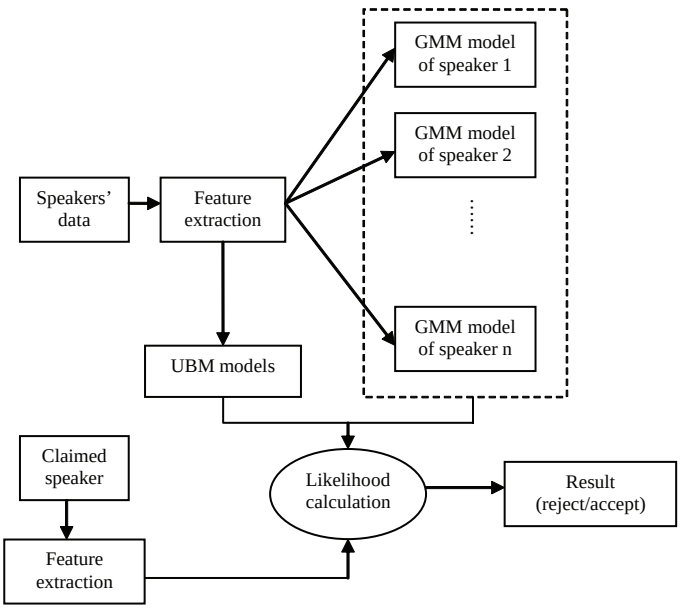
Among these techniques, GMM is used in this chapter as the verification system for testing various features. There are two principal motivations for using Gaussian mixture densities as a representation of speaker identity (Reynolds & Rose, 1995). The first motivation is that the acoustic classes of a speaker's voice, such as vowels and fricatives, can be represented by individual component densities of a multimodal density. These acoustic classes reflect some general speaker-dependent vocal tract configurations that are useful for characterizing speaker identity. Assuming independent feature vectors, the observation density of feature vectors drawn from these hidden acoustic classes is a Gaussian mixture. The second motivation for using Gaussian mixture densities for speaker identification is that a linear combination of Gaussian basis functions is capable of representing a large class of sample distributions. In some sense, the GMM acts as a hybrid between these Gaussian models and VQ by using a discrete set of Gaussian functions, each with their own mean and covariance matrix, to allow a better modeling capability.

In Figure 11, the similarity comparison part and building models part of Figure 10 are modeled in detail. It shows how the GMM is used in the speaker verification process. Variious kinds of features will generate and input into this GMM verification system.

Experiments and Results

In the experiments described here, the GMM classifier with 512 multivariate normal distribu-

Figure 11. The use of GMM for speaker verification



tions with diagonal covariance matrices was used. The NIST 2001 Speaker Recognition Evaluation dataset was used for comparative evaluation. The speech signal recorded from a person was pre-emphasised using a linear filter $H(z) = (1 + 0.97z^{-1})^{-1}$. The results are presented in the form of detection error tradeoff (DET) curves as well as the equal error ratio (EER).

Mel Frequency Cepstral Coefficients (MFCC)

For MFCC feature extraction, the speech signal was divided into 30-millisecond frames with 1/3 overlap. Twelve MFCC coefficients were calculated along with the first and second differences, resulting in 36 dimensional feature vectors. The DET curve for the MFCC + Δ + ΔΔ is shown in Figure 12.

Mel Linear Spectrum Frequencies (MLSF)

MLSF features were extracted from frames of 30-millisecond length with 1/3 overlap. Adding the first and both the first and second differences was tried. It was found that adding the first order differences improved the EER from 18.8% to 16.4%. Adding the second order differences improved the accuracy further, with EER equal to 16.0%. The DET curves are shown in Figure 13.

Hurst Parameter Related Features (pH)

For pH feature extraction, various frame lengths were tried (30, 60, 80, and 100 milliseconds) as well as various frame overlaps (1/3, 1/2, and 2/3). It was found that a frame length of 60 to 100 milliseconds gave the best results in terms of speaker recognition accuracy, while the shorter frames resulted in degraded performance. The frame overlap was not found to show any noticeable difference.

The feature vectors were extracted using Daubechies wavelets of different orders giving pH_4, pH_6, and pH_{12} features with dimensionality of 5, 4, and 3, respectively. They were also concatenated, resulting in pH_{4+6+12} feature vectors.

It was found that the performance of the speaker verification system does not depend on

Figure 12. DET curve and EER value for MFCC features with first- and second-order differences

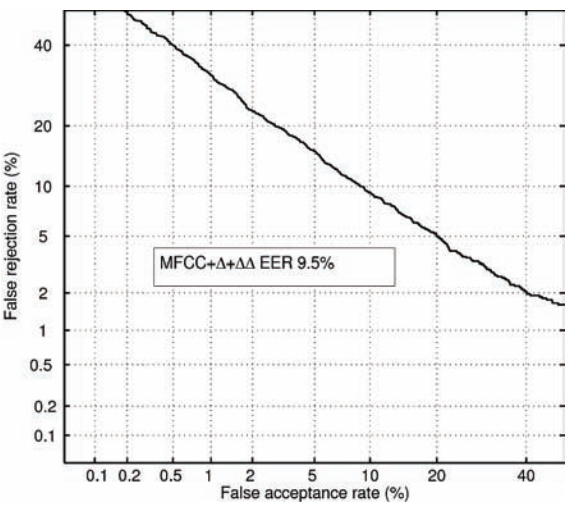
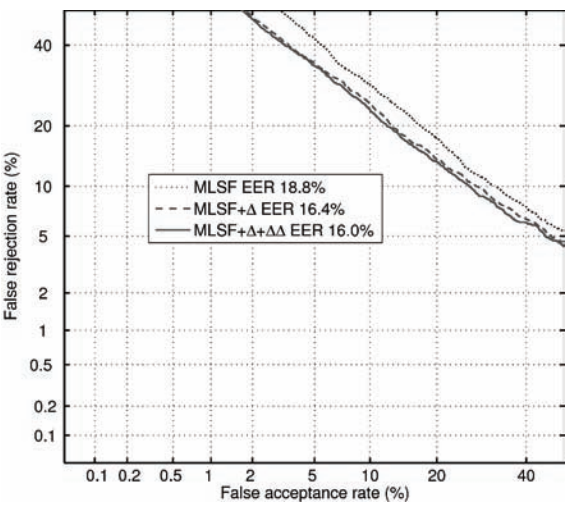


Figure 13. DET curves and EER values for MLSF features with first- and second-order differences



which wavelets are used for feature extraction, as all three pH features resulted in similar speaker verification accuracy, as shown in Figure 14. However, the combined features pH_{4+6+12} resulted in a significant improvement of the accuracy with the EER dropping from 29.0% to 20.8%.

Linear Prediction Residual Phase

For the residual phase features, various frame lengths and LP orders were tried. The results are summarized in Table 10. It was found that the frame length of around 6 milliseconds gave the highest accuracy of speaker verification, although varying the frame length from 3 to 12 milliseconds as well as varying LP order from 6 to 10 did not result in large changes in accuracy.

Figure 15 shows the DET curves obtained using LP residual phase features with frame length of 6 milliseconds and LP orders of 6 and 10. It was also found that adding the first difference features did not change the system performance. Consequently, adding the second difference was not tried.

Features Based on Fractional Fourier Transform

MECB and DMECB features were extracted from 30-ms-long frames with 1/3 overlap. A fractional Fourier transform of orders p = 0.1, 0.2,...,1.0 was tried resulting in features $MECB_p$. $DMECB_{p1\text{-}p2}$ calculated between p_1 = 1.0 and p_2 = 0.1, 0.2,...,0.9. The EER values achieved with $MECB_p$ for p = 0.5...1.0 are summarized in Table 11, and the DET curves are shown in Figure 16. The accuracy achieved with MECB of orders less than 0.5 was very low. It was found that as p decreased from 1.0 to 0.1, the accuracy of the speaker verification also decreased. Adding the first difference features to MECB did not result in significant changes in the accuracy.

The EER values for $DMECB_{1.0\text{-}p2}$ features are summarized in Table 12, and the DET curves for the three features, which resulted in the highest speaker verification accuracy, are shown in Figure 17. As seen from the table, the accuracy of speaker verification increases gradually as the difference

Figure 14. DET curves and EER values pH features

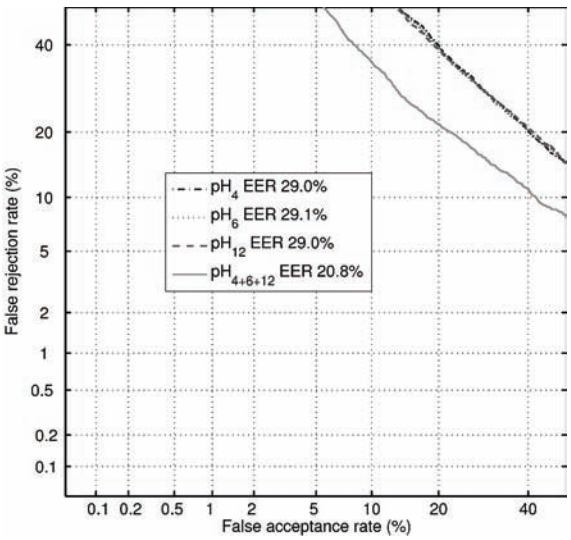
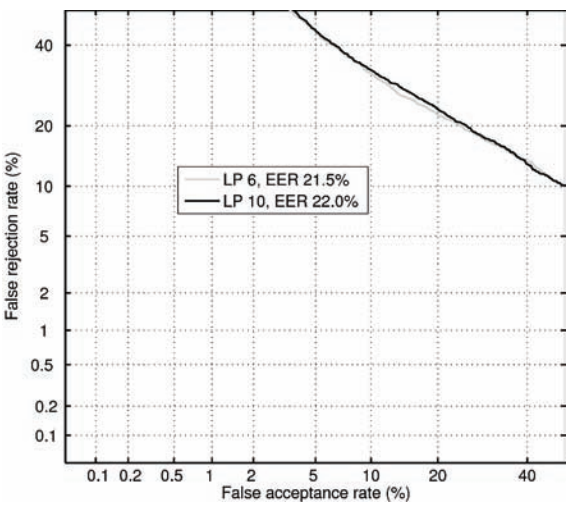


Figure 15. DET curves and EER values for residual phase features



between p_1 and p_2 increases. However, starting from $p_2 = 0.5$, the increase changes to a plateau.

Combination of Features

To determine whether some of the studied features carry any additional information compared to MFCC features, a combination of classifiers that used various features was tried. The combination was implemented using an SVM classifier, where original scores were used as feature values. Part of the score was used to train the classifier, and the rest was used to test the performance. The SVM used had a radial basis function kernel:

$$K(\vec{u},\vec{v}) = \exp\left(-\gamma \left|\vec{u}-\vec{v}\right|^2\right)$$

Good values of parameter γ and error cost C were determined by trial and error. To make the results comparable to those of acoustic features alone, a fivefold cross-validation scheme was applied. The test set of speakers was divided into five approximately equal parts. Each time, one part was left as a testing set, and the remaining four were used to train the SVM. The results of the five tests were then united. The SVM was designed to produce a soft decision rather than a binary class label. The values obtained from all five tests were treated as new scores and used to plot the DET curve as well as calculate the EER value.

It was decided to combine scores produced by GMM using the following features: MFCC with the first and second differences, MLSF with the first and second differences, LP Residual Phase (LP order 6, frame length 3 milliseconds), pH4+6+12, MECB1.0, and DMECB1.0-0.3. The resulting speaker verification accuracy is presented in Figure 18. For the purpose of comparison, the DET curve for the MFCC features with the first and second differences is also shown in the figure.

Table 10. EER achieved with LP residual phase using various LP orders and frame lengths

LP order	Frame Length		
	3 ms	6 ms	12 ms
6	21.8%	21.5%	22.7%
10	21.6%	22.0%	21.9%

Table 11. Equal error rates for MECB features of various orders

$MECB_p$, p	1.0	0.9	0.8
EER, %	17.6	18.7	21.2
$MECB_p$, p	0.7	0.6	0.5
EER, %	24.2	27.5	31.4

Table 12. Equal error rates for DMECB features of various orders

$DMECB_{1.0\text{-}p2}$, p_2	0.9	0.8	0.7	0.6	
EER, %	19.7	19.4	18.9	18.3	
$DMECB_{1.0\text{-}p2}$, p_2	0.5	0.4	0.3	0.2	0.1
EER, %	17.8	17.5	17.1	17.6	17.6

Figure 16. DET curves with EER values for MECB features extracted using various order fractional Fourier transform

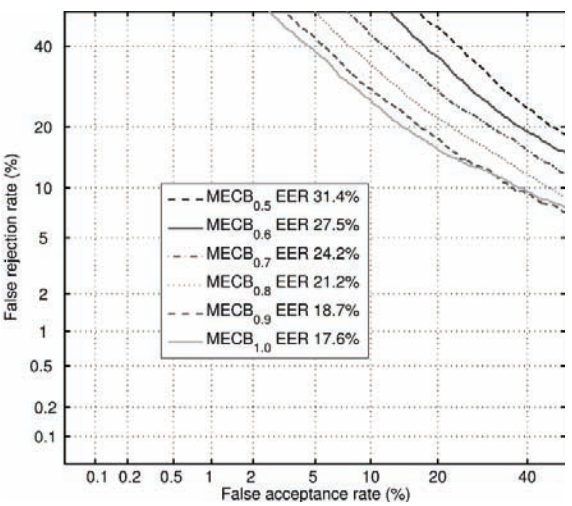


Figure 17. DET curves with EER values for some DMECB features calculated as difference between $MECB_{1.0}$ and MECB of other orders

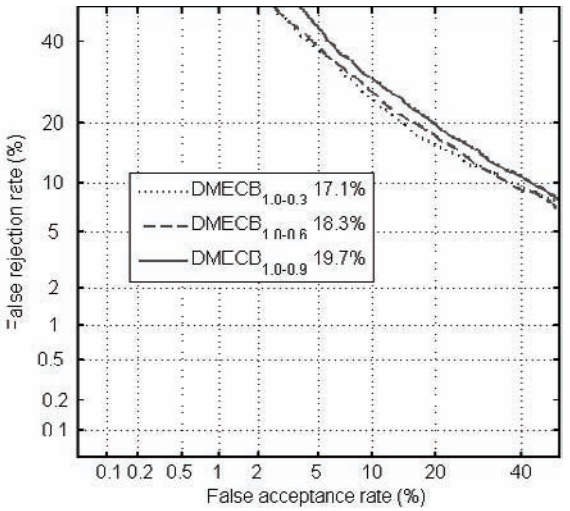
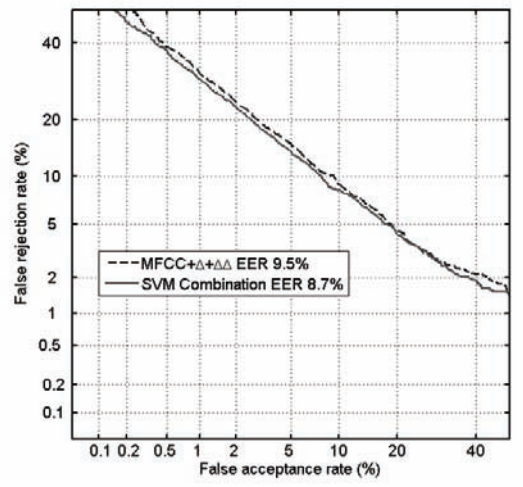


Figure 18. DET curve for combination of acoustic features using SVM



Summary of Studied Features

Several acoustic speech features for speaker recognition have been studied in this chapter.

Hurst parameter features (pH) are features of low dimensionality compared to other acoustic features. Yet they result in a good accuracy of speaker recognition, which suggests they may be attractive in applications where the training data are small. These features were also reported to be less sensitive to channel variation (Sant'Ana et al., 2006), but this has not been investigated in our study.

Mel linear spectrum frequencies (MLSF), mean energy within critical bands (MECB), and DMECB result in speaker recognition accuracy that is comparable to that of MFCC features.

LP residual phase features carry additional information about the speaker compared to other LP-derived features and thus are useful in combination with other acoustic features that depend on a person's vocal tract.

A combination of several acoustic features resulted in higher accuracy of speaker recognition than that achieved using each single acoustic feature type. We conclude, therefore, that the novel features do carry additional information compared to the MFCC features.

Thus, even though it is possible to reduce errors in speaker verification using a number of features and combining scores obtained from UBM-GMM classifiers, the improvement in accuracy is not dramatic. One issue for future investigation is the classifier. It is possible that a GMM classifier is good for MFCC features, but for other features, either a different classifier or different classifier parameters may be needed. For example, Sant'Ana, et al. (2006) used a new classifier for Hurst parameter features, which, they argue, is more suitable for this type of feature than GMM.

There are also other ways that should be investigated to improve the speaker verification system. It would be helpful if there were new novel and efficient features investigated. A good technique of selecting features among existing features is also another interesting area in speaker verification research. To make the system robust, there are also other problems that people have to consider, such as noise and channel attributes.

CONCLUSION

In this chapter, we have examined a number of features from handwriting and speech that can be used to extract forensic evidence about the identity of an individual based on matching his or her speech or handwriting against known samples.

For handwriting, it has been shown that a number of the structural features that document examiners extract and present as evidence in an objective but qualitative manner do indeed possess discriminating power, and thus, use of these features for the purpose of writer identification is justified. It was demonstrated that different characters have different discriminative power, and there exists a noticeable difference in discriminative power between characters and graphemes. This was demonstrated by the observation that the features of grapheme "th" resulted in significantly more accurate identification of writers than the use of any of the features of the three single characters studied.

The most important (indispensable), partially relevant, and irrelevant features were identified under the assumption that the data used were all genuine unconstrained handwriting. The identification of indispensable features provides the information about which features should be given priority in handwriting analysis when the aim is to identify writers. It is possible, however, that under conditions different from those used in the current study (normal unconstrained handwriting), the distribution of the features into the three categories according to their relevance may be different. For example, height-to-width ratio of a character of grapheme is thought to be a more useful feature than both height and width when handwriting is constrained, as in cases where handwriting is extracted from forms. Also, when the grapheme "th" is extracted not from the beginning of words, the presence of loop at the top of the t-stem may more often be included in best feature subsets.

For speech, it was found that even though it is possible to reduce errors in speaker verification using a number of traditional speech and speaker recognition features and combining scores obtained from UBM-GMM classifiers, the improvement in accuracy is not dramatic. It is evident that while existing features provide a reasonable level of accuracy for speaker recognition and verification, they are far from perfect.

One issue for future investigation is the classifier. It is possible that the GMM classifier is good for MFCC features, but for different features, either a different classifier or different classifier parameters may be needed. There are also other ways that should be investigated to improve the speaker verification performance. It would be particularly useful if novel and efficient features were investigated. A good technique of selecting features among existing features is also another interesting area in speaker verification research. To make the system robust, there are also other problems that people have to consider, such as noise and channel.

While some progress has been made in understanding the nuances of both handwriting and speech, and in particular detecting those features that are unique to an individual, it is apparent that there is still considerable improvement and further work to be carried out.

Mechanisms by which speech and handwriting changes depending on emotion are not understood. Neither is it currently possible to determine whether a person is disguising their handwriting or their voice or attempting to forge/imitate the handwriting or voice of another person. Determining whether there exist handwriting and speech features that remain unique even during attempts at forgery and disguise remains an unsolved and largely uninvestigated problem. It is likely that investigation of such features will remain the subject of future research for many years to come.

REFERENCES

Almeida, L.B. (1994). The fractional Fourier transform and time-frequency representations. *IEEE Transactions on Signal Processing, 42*(11), 3084–3091.

Broeders, A.P.A. (2001). Forensic speech and audio analysis forensic linguistics. *Proc. 13th INTERPOL Forensic Science Symposium*. Lyon, France, October 16-19, D2.51-D2.84.

Brummer, N., & du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech and Language, 20*, 230–275.

Campbell, J.P. (1997). Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(8).

Campbell, J.P. (1999). Speaker recognition. In A. Jain, R. Bolle, & S. Pankanti (Eds.), *Biometrics: Personal identification in networked society* (pp. 165–189). Kluwer Academic Publishers.

Campbell, W.M., Brady, K.J., Campbell, J.P., Granville, R., & Reynolds, D.A. (2006). Understanding scores in forensic speaker recognition. *Proceedings of the Odyssey 2006: The Speaker and Language Recognition Workshop*, June 2006.

Campbell, W.M., Reynolds, D.A., Campbell, J.P., & Brady, K.J. (2005). Estimating and evaluating confidence for forensic speaker recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1*, 717–720.

Chong, S.W. (1996). *Tools for forensic questioned document examination* [unpublished project report]. Nanyang Technological University, School of Applied Science.

Cordeiro, H., & Ribeiro, C.M. (2006). Speaker characterization with MLSF. *Proceedings of the Odyssey 2006: The Speaker and Language Recognition Workshop*, San Juan, Puerto Rico.

Daubert, et al., vs. Merrell Dow Pharmaceuticals. (1993). 509 U.S. 579.

Doddington, G.R. (1985). Speaker recognition—Identifying people by their voices. *Proceedings of the IEEE*, 73(11), 1651–1664.

Eldridge, M.A., Nimmo-Smith, I.,Wing, A.M., & Totty, R.N. (1984). The variability of selected features in cursive handwriting—categorical measures. *Journal of Forensic Science Society, 24*, 179–219.

Found, B., & Rogers, D. (1995). Contemporary issues in forensic handwriting examination: A discussion of key issues in the wake of the Starzecpyzel decision. *Journal of Forensic Document Examination, 8*, 1–31.

Found, B., & Rogers, D. (1998). A consideration of the theoretical basis of forensic handwriting examination. *Journal of Forensic Document Examination, 4* 2), 109–118.

Govindaraju, V., Srihari, S.N., & Shin, Y.-C. (1999). Use of handwriting recognition features in handwriting identification. *Proceedings of the 9th Biennial Conference of the International Graphonomics Society*, (pp. 73–78). Singapore.

Guo, Z., & Hall, R.W. (1989). Parallel thinning with two-subiteration algorithms. *Comm. ACM, 32* 3), 359–373.

Harrison, W.R. (1981). *Suspect documents, their scientific examinations.* Illinois: Nelson-Hall.

Hilton, O. (1993). *Scientific examination of questioned documents.* Florida: CRC Hall.

Holcombe, G., & Leedham, C.G. (1995). An experimental imaging environment for examination of questioned documents. *Proceedings of the 7th International Graphonomics Conference* (pp. 80–81). London, Ontario.

Huber, R.A., & Headrick, A.M. (1999). *Handwriting identification: Facts and fundamentals.* CRC Press.

Kam, M., Fielding, G., & Conn, R. (1997). Writer identification by professional document examiners. *Journal of Forensic Sciences, 42*(5), 778–786.

Kam, M., Fielding, G., & Conn, R. (1998). Effects of monetary incentives on performance in document examination proficiency tests. *Journal of Forensic Science, 43*(5), 1000–1004.

Kam, M., Gummadidala, K., Fielding, G., & Conn, R. (2001). Signature authentication by forensic document examiners. *Journal of Forensic Science, 46*(6), 884–888.

Kam, M., Wetstein, J., & Conn, R. (1994). Proficiency of professional document examiners in writer identification. *Journal of Forensic Sciences, 39*(1), 5–14.

Kunzel, H.J. (1994). Current approaches to forensic speaker recognition. *Proceedings of the ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, *48*, 135–141.

Leedham, C.G., Holcombe, G., & Sagar, V.K. (1995). Image processing tools for the interactive forensic examination of questioned document. *Proceedings of the European Convention on Security and Detection*, (pp. 225-228). Brighton, UK.

Matsui, T., & Furui, S. (1994). Comparison of text-independent speaker recognition methods using VQ-distortion and discrete/continuous HMMs. *IEEE Transactions on Speech and Audio Processing, 2*, 456–459.

Murty, K.S R., & Yegnanarayana, B. (2006). Combining evidence from residual phase and MFCC features for speaker recognition. *IEEE Signal Processing Letters, 13*(1), 52–55.

Oppenheim, A.V., & Schafer, R.W. (2004). From frequency to quefrency: A history of the cepstrum. *Signal Processing Magazine, IEEE, 21*, 95–106.

Pervouchine, V., Leedham, C.G., & Melikhov, K. (2005a). Handwritten character skeletonisation for forensic document analysis. *Proceedings of the 20th Annual ACM Symposium on Applied Computing*, (pp. 754–758). Santa Fe, New Mexico.

Pervouchine, V., Leedham, C.G., & Melikhov, K. (2005b). Three-stage handwriting stroke extraction method with hidden loop recovery. *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR'2005).* Seoul, Korea.

Poritz, A. (1982). Linear predictive hidden Markov models and the speech signal. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 7*, 1291–1294.

Reynolds, D.A., & Rose, R.C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing, 3*, 72–83.

Robertson, E.W. (1991). *Fundamentals of document examination*. IL: Nelson Hall.

Sant'Ana, R., Coelho, R., & Alcaim, A. (2006). Text-independent speaker recognition based on the Hurst parameter and the multidimensional fractional Brownian motion model. *IEEE Transactions on Audio, Speech, and Language Processing, 14*(3), 931–940.

Solihin, Y. (1997). *A toolset of image processing algorithms for forensic document examination* [unpublished master's thesis]. Nanyang Technological University, School of Applied Science.

Srihari, S.N., Cha, S.-H., Arora, H., & Lee, S. (2002). Individuality of handwriting. *Journal of Forensic Sciences, 47*(4), 1–17.

Srihari, S.N., Cha, S.-H., & Lee, S. (2001). Establishing handwriting individuality using pattern recognition techniques. *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR'2001),* (pp. 1195–1204). Seattle, Washington.

Srihari, S.N., Tomai, C.I., Zhang, B., & Lee, S. (2003). Individuality of numerals. *Proceedings of the 7th Internatiional Conference on Document Analysis and Recognition (ICDAR'2003),* (pp. 1096–1100). Edinburgh, UK.

Suen, C.Y., & Wang, P.S.P. (Eds.). (1994). *Thinning methodologies for pattern recognition* (Vol. 8). Singapore: World Scientific.

United States vs. Starzecpyzel. (1995). 880 F. Supp. 1027, 1046 (S.D.N.Y.)

Veith, D., & Abry, P. (1994). A wavelet-based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory, 45*(3), 878–897.

Wang, J., & Wang, J. (2005). Speaker recognition using features derived from fractional Fourier transform. *Proceedings of the Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, (pp. 95–100).

Weiss, S.M., & Kulikowski, C.A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems.* Morgan Kaufmann.

Yang, J., Parekh, R., & Honavar, V. (1997). *Distal: An inter-pattern distance-based constructive learning algorithm* (Technical Report No. ISU-CS-TR 97-06). Iowa State University, Department of Computer Science.

Zhang, B., Srihari, S.N., & Lee, S. (2003). Individuality of handwritten characters. *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'2003),* (pp. 1086–1090). Edinburgh, UK.

Zhang, T.Y., & Suen, C.Y. (1984). A fast parallel algorithm for thinning digital patterns. *Comm. ACM, 27*(3), 236–239.

Zheng, N., & Ching, P.C. (2004). Using Haar transformed vocal source information for automatic speaker recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1,* 77–80.

Chapter VII
Image Pattern Recognition–Based Morphological Structure and Applications

Donggang Yu
Bioinformatics Applications Research Centre, James Cook University, Australia

Tuan D. Pham
Bioinformatics Applications Research Centre, James Cook University, Australia

Hong Yan
City University of Hong Kong, Hong Kong

ABSTRACT

This chapter describes a new pattern recognition method: pattern recognition-based morphological structure. First, smooth following and linearization are introduced based on difference chain codes. Second, morphological structural points are described in terms of smooth followed contours and linearized lines, and then the patterns of morphological structural points and their properties are given. Morphological structural points are basic tools for pattern recognition-based morphological structure. Furthermore, we discuss how the morphological structure can be used to recognize and classify images. One application is document image processing and recognition, analysis and recognition of broken handwritten digits. Another one is dynamic analysis and recognition of cell-cycle screening based on morphological structures. Finally, a conclusion is given, including advantages, disadvantages, and future research.

INTRODUCTION

In intelligent information systems such as document and medical image processing, pattern recognition of images play an important role. The origin of character and shape recognition can be found as early as 1870, although it became a reality in the 1950s when computers were commonly used. For example, pattern recognition has wide applications in modern society: document reading and sorting, postal address reading, bank check reading, form recognition, writer recognition, signature verification, digital bar code reading, engineering drawing recognition, analysis and recognition of cells, face recognition, and shape recognition of various objects (e.g., ships, airplanes, etc.).

If postal codes can be recognized, then mail can be separated automatically in a processing system. In banks, a lot of checks need to be processed every day. One very hard task is that the dollar amount has to be input into computers by people. If these handwritten digits can be processed and recognized, one check processing system instead of people can do the job automatically.

One important problem is that there are broken and spurious segments caused by segmentation and threshold errors of noisy digits, the tools used or the writing style in handwritten digits in document images. In this case, it is difficult for most recognition methods to deal with them, including both structural and statistical approaches (Hu, 1998; Lee, 1996; Malaviya & Klette, 1996; Shi, 2002; Yan, 1993, 1994). In structural recognition method (Hu, 1998), it is difficult to describe the structure of the parts of broken segments of handwritten digits and spurious segments. If optimized nearest neighbor classifier (ONNC) (Yan, 1993) is used, the parts of broken segments of handwritten digits and spurious segments can influence training result. If the handwritten digits with broken and spurious parts belong to the test set, it is not easily recognized because there is a big difference between the digit shape and the normal digits.

In fact, it is based on the morphology structure of an object for a human to recognize an image. For example, if there are only two right-concave changes of one object skeleton, and the object is a digit based on prior knowledge, then the object is digit 3 for most of handwritten digit 3.

Morphological structure includes the morphology of lines, arcs, contours, and shapes. Also, all convex and concave changes have direction for each morphology change. Therefore, for the same shape of arcs, there are two directions: convex arc and concave arc. This chapter has made a new breakthrough in pattern recognition of images based on morphological structure. In this chapter, many new concepts of morphology description features are proposed; that is, contour smoothing following, smoothing of skeleton, linearization based on difference codes, structural points of contours, and description of morphological structures. Two applications, reconstruction and recognition of broken handwritten digits in document images and dynamic analysis and recognition of cell images, are described in detail.

SMOOTH FOLLOWING

Accurate representation and processing of the contour of a binary image play an important role in processing and recognition of images. For example, the result of thinning, curve fitting, contour following, polygon clipping, and mathematical morphology operations may depend on the contour shape of the image (Gonzalez & Woods, 1993; Freeman, 1961; Pavlidis, 1982; Rosenfeld, 1973).

In practical applications, the contour of a binary image is often corrupted by noise, which makes the recognition of the image unreliable. We propose an efficient method to smooth and linearize the contour of an image. Our method is developed based on a set of rules implemented

for contour tracing, which does not require any float point average operations.

The contour of a binary image can be represented with the Freeman code (Freeman, 1961). Many binary image processing algorithms based on contour following have been developed in the past (Pavlidis, 1982; Wang, Qi, Yu & Xu, 1990). The contour direction chain code can be used to represent a contour. For pixels sampled on rectangular grids, the Freeman code is shown in Figure 1(1).

For a pixel $p \epsilon P$, we call $NB_p=(b_7b_6b_5b_4b_3b_2b_1b_0)$ the 8-neighborhood byte of p shown in Figure 1(2) (Pavlidis, 1982; Wang et al., 1990). Contour following is a procedure to determine the sequence of all contour pixels of an object. If p is a contour pixel, then one neighborhood pixel in NB_p must be the next contour pixel in the sequence. If b_i $(0 \leq b_i \leq 7)$ is the next contour pixel, then the index number i is the direction chain code of the pixel p. We can make use of two types of binary image contour connections. Suppose P is a binary image, S is an object with pixel value 1, $\bar{S}$ is the background with pixel value 0, and S_k is the set of pixels of contour k in the image. If S is 8-connected, then both S and $\bar{S}$ are 4-connected, and if $\bar{S}$ is 4-connected then S and S_k are 8-connected. Here, we assume that S_k is 8-connected.

In a binary image such as a handwritten character, there can be many spurious pixels that make the contour description of an object difficult. For example, Figures 2(1-3) contain several handwritten-connected numerals taken from the U.S. National Institute of Science and Technology (NIST) database.

The results of contour following are shown in Figures 2(1-3). Since there are many spurious points, it is not easy to describe the contour of handwritten numerals 23, 01, and 58 with the direction chain code.

For example, it is difficult to represent the outer contour of numeral 23, the internal contour of numeral 01, and both the outer and the internal contours of numeral 58. Thus, it is necessary to modify the followed pixels, delete noisy ones, and set the starting pixel at the upper-left corner so the character contour can be represented reliably for further processing. Our algorithm for contour smooth following is developed based on difference codes. The algorithm is able to delete noisy pixels, modify followed ones, and determine the starting pixel of a contour chain. If the absolute value of the difference code of each pair of neighborhood pixels is less than 2, the procedure produces the desired result.

Contour Smooth Following with Difference Codes

The k-th contour, S_k, of a binary image P can be represented as:

Figure 1. Freeman codes and neighboring bytes

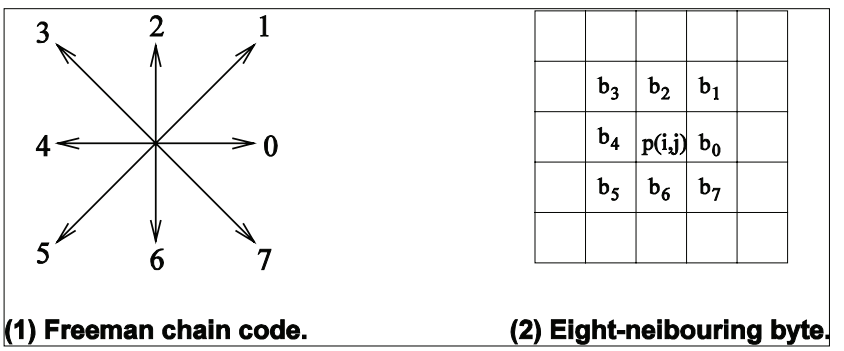


Figure 2. Original images of handwritten connected digits, 01, 23, and 58

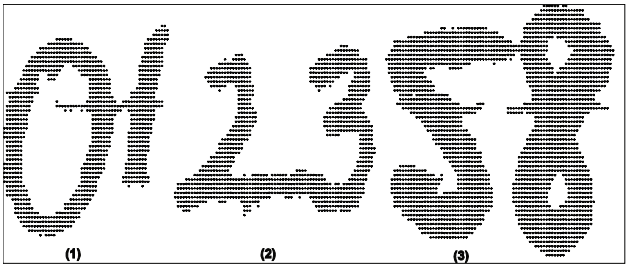
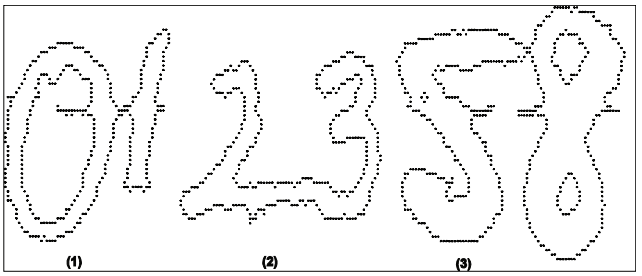


Figure 3. Contours of handwritten connected digits, 01, 23, and 58



$$C_K = \{c_0, c_1 ... c_i, ... c_{n-1}, c_n\} \quad (1)$$

$$X_K = \{x_0, x_1 ... x_i, ... x_{n-1}, x_n\} \quad (2)$$

$$Y_K = \{y_0, y_1 ... y_i, ... y_{n-1}, y_n\} \quad (3)$$

where C_k is the direction chain code set of contour *k*, *i* is the pixel index of the contour, c_0 is the starting direction code pointing from the first pixel to the second pixel of the contour, c_i is the direction code of pixel *i* pointing from pixel *i* to pixel (*i*+1), c_{n-1} is the direction code of pixel (*n*-1) (the last pixel in the sequence) pointing from pixel (*n*-1) to the first pixel of the contour, and c_n is same as c_0. X_k and Y_k are the *x* and *y* coordinate sets of contour *k*, respectively.

The difference code of a contour is defined as follows:

$$d_i = c_{i+1} - c_i \quad (4)$$

It can be calculated as:

$$d_i = \begin{cases} c_{i+1} - c_i & if \mid c_{i+1} - c_i \mid < 4 \\ c_{i+1} - c_i - 8 & if \mid c_{i+1} - c \mid > 4 \\ 4 & if \mid c_{i+1} - c_i \mid = 4 \end{cases} \quad (5)$$

Therefore, the difference code value is 0, ±1, ±2, ±3, or 4. If d_i is equal to ±1, ±2, or ±3, then the current direction chain code of pixel *i* is changed with an anticlockwise or clockwise rotation of 45°, 90°, or 135°. If d_i is equal to ±4, the current direction chain code of pixel *i* is changed with a rotation of 180°. In general, most contour pixel direction chain codes are 0 or ±1. Otherwise, there should be some noisy contour pixels.

In terms of Equation 5, if the difference code value is ±2, ±3, or 4, all difference code groups are shown in Figures 4-8. It is necessary to note that some difference codes of these groups do not appear if contour following is carried out with an 8-connected contour.

The algorithm for smoothing a contour is developed based on the following criteria, corresponding to the transformations of the contour codes shown with the dashed lines in Figures 4-8.

Figure 4. The contour smoothing procedure with d_i = 2, c_i being even and c_{i+1} being even

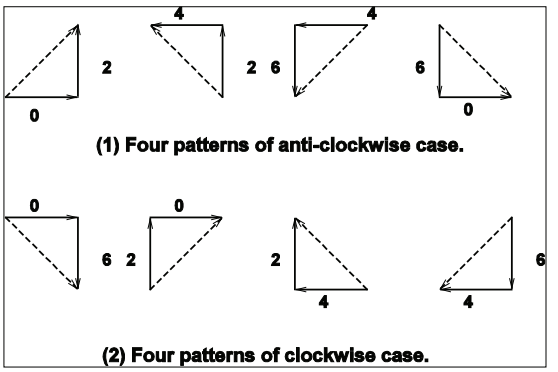


Figure 5. The contour smoothing procedure with d_i = 2, c_i being odd and c_{i+1} being odd

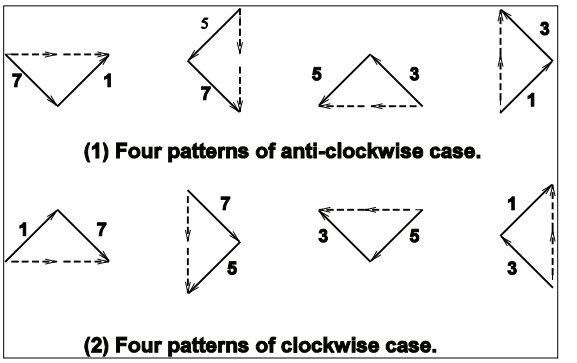


Figure 6. The contour smoothing procedure with d_i = 3, c_i being even and c_{i+1} being odd

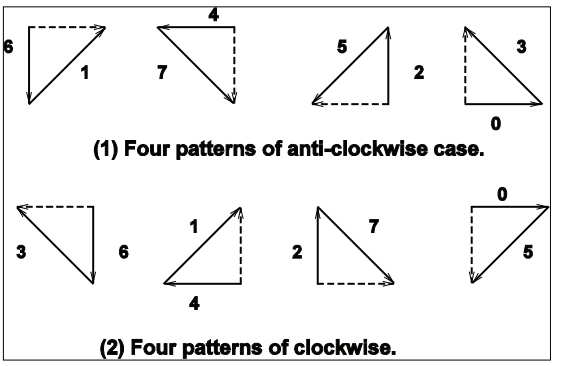


Figure 7. The contour smoothing procedure with d_i = 3, c_i being odd and c_{i+1} being even

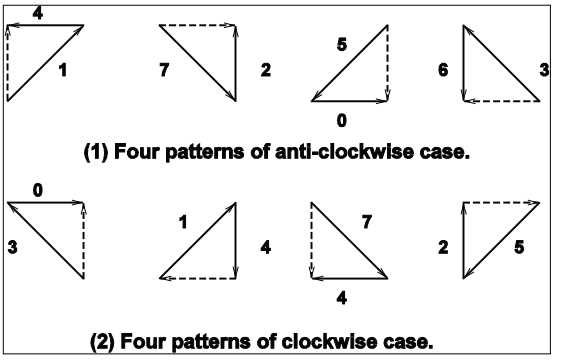
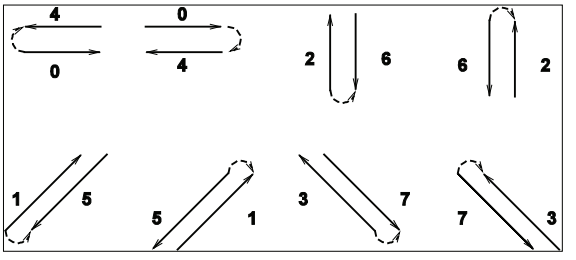


Figure 8. The contour smoothing procedure with d_i = 4



Here, we assume that c_{sj}, x_{sj}, and y_{sj} are the new direction chain code, x and y coordinate of pixel i on the image contour, respectively.

Criterion 2.1.1: If $|d_i| \leq 1$, then:

- $c_{sj} = c_i$
- $x_{sj} = x_i$
- $y_{sj} = y_i$
- The next pixel is the pixel (i+1) in the original contour chain.

Criterion 2.1.2: If $|d_i|$=2, c_i is even and c_{i+1} is even, then:

- $c_{sj} = c_i$-1 (mod 8)**;** if d_i=6 or -2 (clockwise)
- $c_{sj} = c_i$+1 (mod 8)**;** if d_i=-6 or 2 (anticlockwise)
- $x_{sj} = x_i$
- $y_{sj} = y_i$
- The next pixel is the pixel (i+2) in the original contour chain.

Criterion 2.1.3: If $|d_i|$=2, c_i is odd and c_{i+1} is odd, then:

- $c_{sj} = c_i$-1 (mod 8)**;** if d_i=6 or -2 (clockwise)
- $c_{sj} = c_i$+1 (mod 8)**;** if d_i=-6 or 2 (anticlockwise)
- $x_{sj} = x_i$
- $y_{sj} = y_i$
- Set:
  1. $c_{i+1} = c_i$-1 (mod 8)**;** if d_i=6 or -2 (clockwise)
  2. $c_{i+1} = c_i$+1 (mod 8)**;** if d_i=-6 or 2 (clockwise)
  3. x and y coordinates of the pixel (i+1) based on Figure 5
  4. The next pixel is the pixel (i+1) in the original contour chain.

Criterion 2.1.4: If $|d_i|$=3, c_i is even and c_{i+1} is odd, then:

- $c_{i+1} = c_i$-1 (mod 8)**;** if d_i=5 or -3 (clockwise)
- $c_{i+1} = c_i$-1 (mod 8)**;** if d_i=-5 or 3 (clockwise)
- $x_{sj} = x_i$
- $y_{sj} = y_i$
- The next pixel is the pixel (i+2) in the original contour chain.

Criterion 2.1.5: If $|d_i|$=3, c_i is odd and c_{i+1} is even, then:

- $c_{i+1} = c_i$-1 (mod 8)**;** if d_i=5 or -3 (clockwise)
- $c_{i+1} = c_i$-1 (mod 8)**;** if d_i=-5 or 3 (clockwise)
- $x_{sj} = x_i$
- $y_{sj} = y_i$
- The next pixel is the pixel (i+2) in the original contour chain.

Criterion 2.1.6: If $|d_i|$=4, then:

- Remove pixels i and (i+1)
- The next pixel is the pixel (i+2) in the original contour chain.

It is possible that the starting pixel of the new chain is not the same as that of the original chain, because the starting pixel of the original chain may be processed based on the previous criteria. Therefore, it is necessary to redetermine the starting pixel of the new contour chain according to the following criterion. We assume that the origin is at the upper-left corner of the image.

Criterion 2.1.7: Assume:

$$y_{mi} = \{y_{s0}, y_{s1} \ldots y_{si}, \ldots y_{s(n-1)}, y_{sn}\}$$

where y_{mi} is the y coordinate of pixel i, which has the minimum y coordinate in the new contour chain:

$$y_m = \{y_{m0}, y_{m1} \ldots y_{mt}\} \quad (6)$$

where $y_{m0} = y_{m1} = \ldots = y_{mt}$, y_m is the y coordinate set of the corresponding minimum y coordinate. The x coordinate set corresponding to the minimum y is:

$$x_m = \{x_{m0}, x_{m1} \ldots x_{mt}\} \quad (7)$$

Then the x and y coordinates of the starting pixel in the new contour chain are:

$$y_{sp} = y_m$$

$$x_{sp} = \min\{x_{m0}, x_{m1} \ldots x_{mt}\}$$

After the new starting pixel is determined, all pixels of the new contour chain need to be reordered based on the new starting pixel. In terms of the previous algorithm, contour chain can be smoothed several times until $|d_i| = 1$ for all pixels in the contour chain. Suppose the direction chain code set of the final result produced by the smoothing algorithm is:

$$C_{lsk} = \{c_{ls0}, c_{ls1} \ldots c_{lsi}, \ldots c_{ls(l-1)}, c_{lsl}\} \quad (8)$$

Then:

$$d_{si} = c_{ls(i+1)} - c_{lsi} \quad (9)$$

The smoothed contour chain has the following properties:

- Its difference code $|d_{lsi}| \leq 1$ ($i = 0,1,\ldots l$)
- The starting direction chain code c_{ls0} = code 5
- The last direction chain code $c_{ls(l-1)}$ = code 4

For the binary images in Figures 2(1-3), the processed results made by the smoothing algorithm are shown in Figures 9(1-3). We can see that the noisy pixels of the contour have been removed. Also, smoothing makes the linearization procedure possible, which will be discussed in the next section.

The Smoothing Following Based on Removing Spurious Point Groups

A followed chain based on the algorithm introduced in the previous section can be processed in terms of following procedure, which can remove some spurious point groups in a chain. The modified results of these pattern models with a spurious point group that has two convex or concave points in the direction of a special chain code are demonstrated in Figure 10. In this figure, *i* represents the *i*-th point of a chain, a circle is represented as a point of the previous chain, and a circle dot is represented as a modified result point. Figure 10 demonstrates that there are two convex and concave spurious points in the direction of code 0. In these cases, two criteria described next are used.

Criterion 2.2.1: $(c_i=0) \wedge (c_{i+1}=1) \wedge (c_{i+2}=0) \wedge (c_{i+3}=7) \wedge (c_{i+4}=0)$, then:

- $c_{s(i+1)}=0,\ x_{s(i+1)} = x_{(i+1)},\ y_{s(i+1)} = y_{(i+1)}$
- $c_{s(i+2)}=0,\ x_{s(i+2)} = x_{(i+2)},\ y_{s(i+2)} = y_{(i+2)+1}$
- $c_{s(i+3)}=0,\ x_{s(i+3)} = x_{(i+3)},\ y_{s(i+3)} = y_{(i+3)+1}$

Criterion 2.2.2: $(c_i=0) \wedge (c_{i+1}=7) \wedge (c_{i+2}=0) \wedge (c_{i+3}=1) \wedge (c_{i+4}=0)$, then:

- $c_{s(i+1)}=0,\ x_{s(i+1)} = x_{(i+1)},\ y_{s(i+1)} = y_{(i+1)}$
- $c_{s(i+2)}=0,\ x_{s(i+2)} = x_{(i+2)},\ y_{s(i+2)} = y_{(i+2)-1}$
- $c_{s(i+3)}=0,\ x_{s(i+3)} = x_{(i+3)},\ y_{s(i+3)} = y_{(i+3)-1}$

Figure 9. The contours of smooth following of handwritten connected digits, 01, 23, and 58

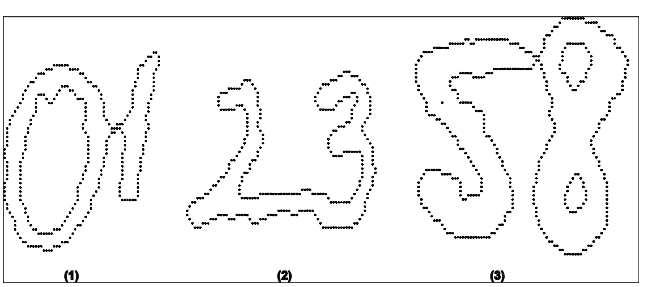
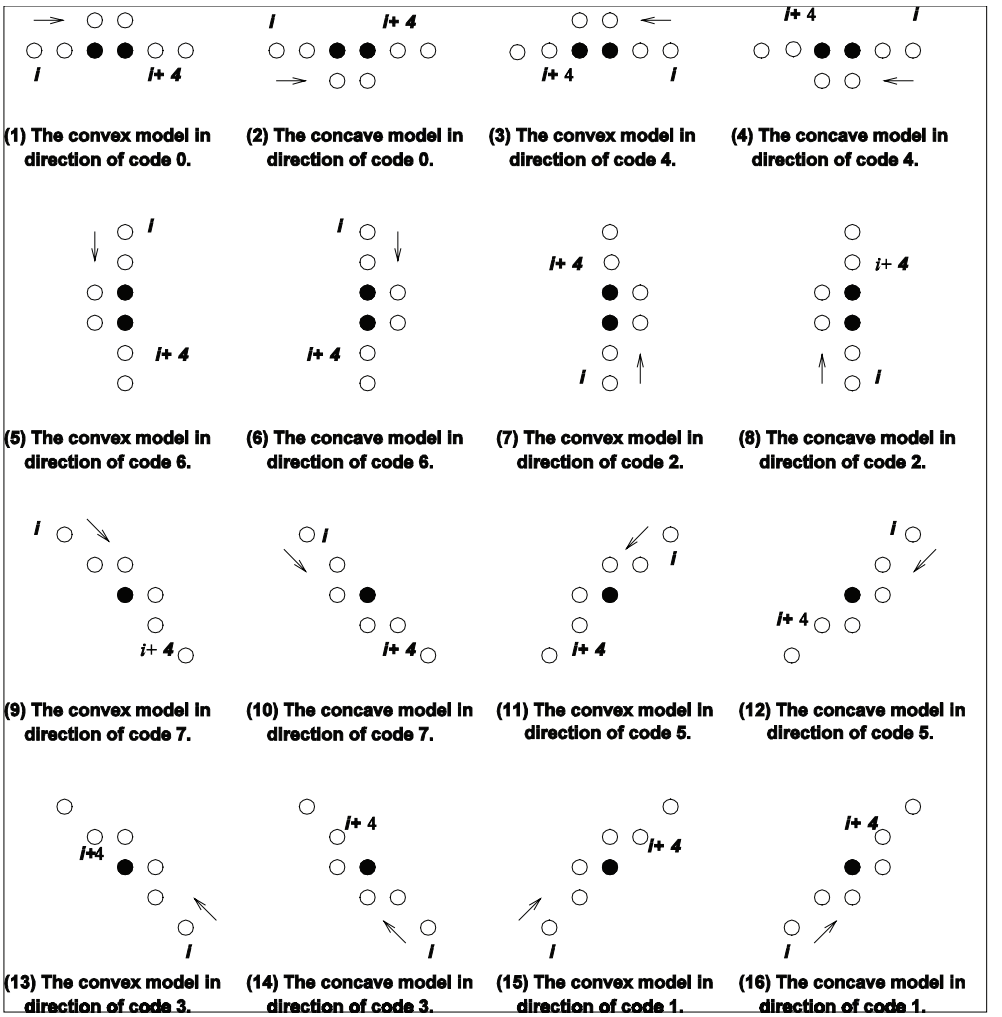


Figure 10. The pattern models of spurious point group (two points)



Similar to Criteria 2.2.1 and 2.2.2, we can introduce Criteria 2.2.3 to 2.2.8 based on Figures 10(3-8).

Figures 10(9-10) demonstrate that there are two convex and concave spurious points in the direction of code 7. In these cases, two criteria described next are used.

Criterion 2.2.9: $(c_i=7) \Lambda (c_{i+1}=0) \Lambda (c_{i+2}=7) \Lambda (c_{i+3}=6) \Lambda (c_{i+4}=7)$, then:

- $c_{s(i+1)}=7$, $x_{s(i+1)}=x_{(i+1)}, y_{s(i+1)}=y_{(i+1)}$
- $c_{s(i+2)}=7$, $x_{s(i+2)}=x_{(i+2)}, y_{s(i+2)}=y_{(i+2)+1}$
- Remove (i+3)th point of the smoothed chain and reorder points of the smoothed chain.

Criterion 2.2.10: $(c_i=7) \Lambda (c_{i+1}=6) \Lambda (c_{i+2}=7) \Lambda (c_{i+3}=0) \Lambda (c_{i+4}=7)$, then:

- $c_{s(i+1)}=7$, $x_{s(i+1)}=x_{(i+1)}, y_{s(i+1)}=y_{(i+1)}$
- $c_{s(i+2)}=7$, $x_{s(i+2)}=x_{(i+2)+1}, y_{s(i+2)}=y_{(i+2)}$
- Remove (i+3)th point of the smoothed chain and reorder points of the smoothed chain.

Similar to Criteria 2.2.9 and 2.2.10, we can introduce Criteria 2.2.11 to 2.2.16 based on Figures 10(11-16).

Based on the property of smoothed contour chain, the only remaining task is to select the first point and the last point of the chain based on the pattern model of Figure 10. In this case, it is necessary to redetermine the starting pixel of the new contour chain based on Criterion 2.1.7. The previous processing is finished until there are no spurious point groups like those shown in Figure 10 in the new contour. For the images of Figures 9(1-3), the processed results are shown in Figures 11(1-3), respectively. For example, Figure 11(3) demonstrates that the spurious point groups (two points) of Figure 11(3) have been removed (see the upper-left part of the digit). We can see that the noisy pixels of the contour have been removed based on the models of the spurious points. Similarly, the pattern models with a spurious point group that has more than two points and modifying results can also be constructed.

Skeleton and Its Smoothing

Many methods of image processing and recognition are based on the skeleton, which is obtained by a thinning algorithm. Generally, skeletons of binary images are not smooth because there are some spurious points, which make it difficult to extract skeleton features. Therefore, it is useful to remove spurious points by smoothing skeletons. Also, the connection of the skeleton should be retained. An example image of a thinned broken digit is shown in Figure 12.

The original image and its smoothed contours are shown in Figures 12(1-2), respectively, and the skeleton is shown in Figure 12(3). The skeleton can further be smoothed based on some patterns as shown in Figure 13(1), where "0" represents background pixel, "l" represents a skeleton pixel, and "x" stands for "don't care" point. In this way,

Figure 11. The contours of smooth following of handwritten connected digits 01, 23, and 58 based on the pattern models of spurious point group (two points)

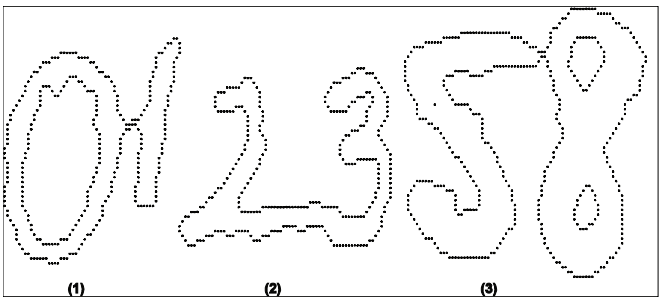
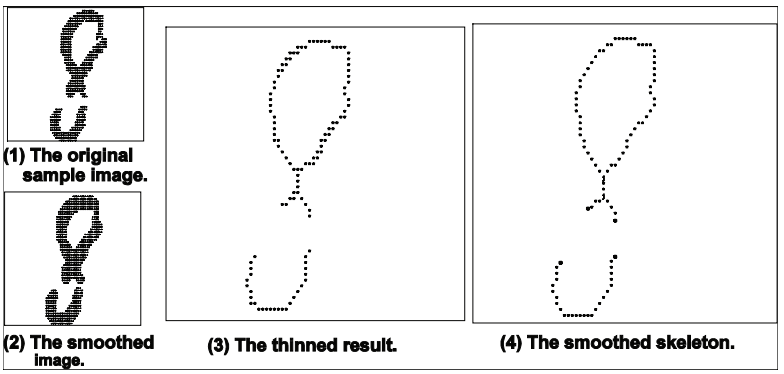


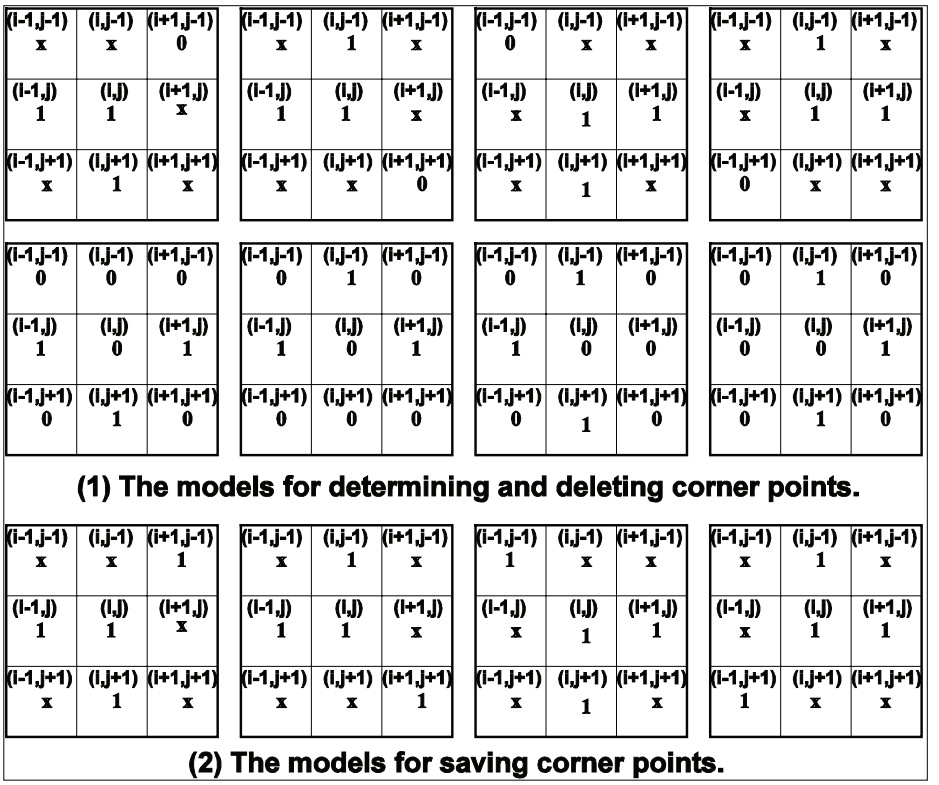
Figure 12. Thinning and smoothed skeleton



some "corner" points of the skeletons are removed. In order to retain the continuity of the skeletons, if the pixels of the skeleton belong to one of the patterns in Figure 13(2), they are saved. Based on our algorithm, the skeleton (as shown in Figure 12(3)) is smoothed, and the final skeleton is shown in Figure 12(4). Also, the "end" and "junction" points of the smoothed skeletons can be extracted (shown in Figure 12(4)), where the character "e" represents the "end" point of the skeleton, and character "j" represents the "junction" point of the skeletons. If we calculate the difference code of two neighboring pixels between one "e" point and its neighboring "j" point of smoothed skeleton, then $|d_i|$equals 0 or 1. If there is no "junction" point on the smoothed skeleton, $|d_i|$ between two "e" points of the skeleton equals 0 or 1. Therefore, they can be linearized, and the structural points can be extracted based on the algorithm described in Sections-Linearization and Structural Point, and then we can detect the morphological change of the smoothed skeleton. The geometrical location and direction change of the "end" points and the structure of the smoothed skeletons are useful to determine spurious segments and broken points of broken handwritten digits.

For example, the original binary image of typhoon is shown in Figure 14(1). After some preprocessing such as dilating, contour following, reconstructing, filling, and smooth following, the reconstructed typhoon image is shown in Figures 14(2-3). The filled image is shown in Figure 14(4). Its skeleton can be obtained in Figure 14(5). Based on our algorithm of smoothing skeleton (see the patterns in Figure 13), its smoothing skeleton is shown in Figure 14(6). Furthermore, the "end" and "junction" points are found, which are shown in Figures 14(7-8).

Figure 13. The pattern models for the smoothing skeleton



Summary

Some efficient algorithms of smooth following are developed based on smoothing structure patterns. Two types of patterns are constructed based on the structures of difference chain codes. Using these algorithms, some spurious points or spurious point group (two points) of contours are removed from the starting point to the last point of contours. For the skeletons of images, the corresponding smooth structural patterns are described. Based on these smooth patterns, the spurious points of each skeleton segment are removed. After smooth following of contours, the difference codes between the neighboring points of contours are less than 2. For the skeletons of images, the difference codes between the neighboring points of each skeleton segment are also less than 2. For each skeleton segment, there are three cases: (a) the skeleton segment is between neighboring "end" and "junction" points; (b) the skeleton segment is between two neighboring "junction" points of skeletons; (c) the skeleton segment is between two "end" points, if there are only two "end" points in the skeleton. In this way, the results of smooth following of contours make the linearization of contours possible. Also, the smoothing of skeleton of image makes the linearization of skeleton possible. Therefore, smooth following of contours and the smoothing of skeleton overcome the question that makes Freeman codes to describe contour and skeleton difficult because of spurious points of contours and skeleton.

LINEARIZATION OF CONTOURS-BASED DIFFERENCE CHAIN CODES

The description of binary image contour plays an important role for the shape description and recognition of images. One useful description of binary image contour should contain a suitable series of lines (critical points) and their series of description features, which are linearity, convexity or concavity of lines, curvature angle, bend angle, and convexity or concavity of bend angle. The method proposed in this section can make it possible.

Some original binary images are shown in Figures 15(1,4), 16(1). Their contour following results are shown in Figures 15(2,5), 16(2).

The difference code is defined as:

$$d_i = c_{i+1} - c_i \tag{10}$$

In the smoothed contour, $|d_i|$ equals 0 or 1 based on the algorithm described in Section-Smooth Following. The smooth following results

Figure 14. The preprocessing, smoothing, skeleton and smoothing skeleton of a typhoon object image from GIS image

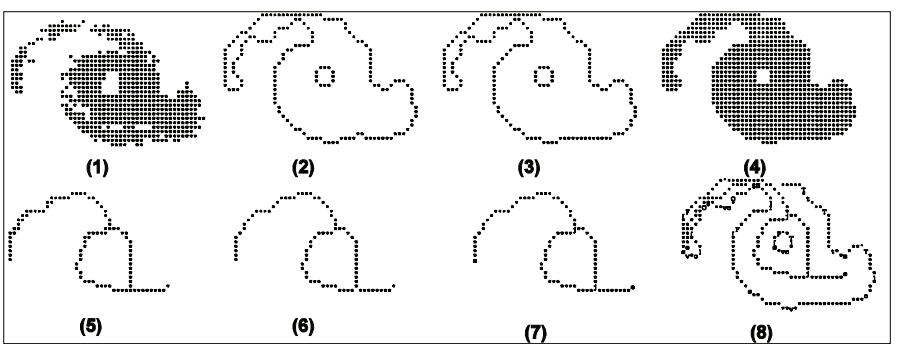


Figure 15. Handwritten digit 7 and its processing results

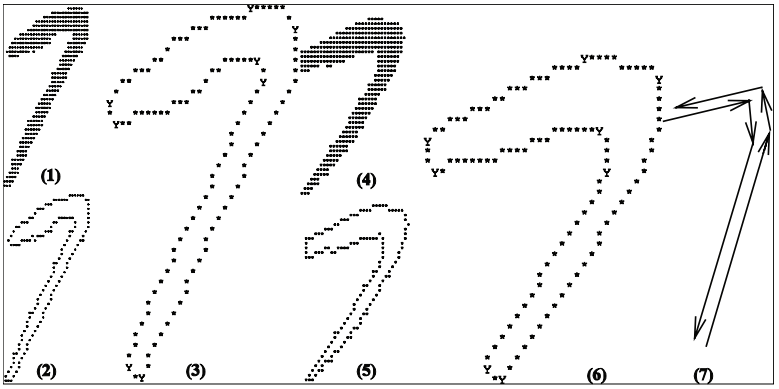
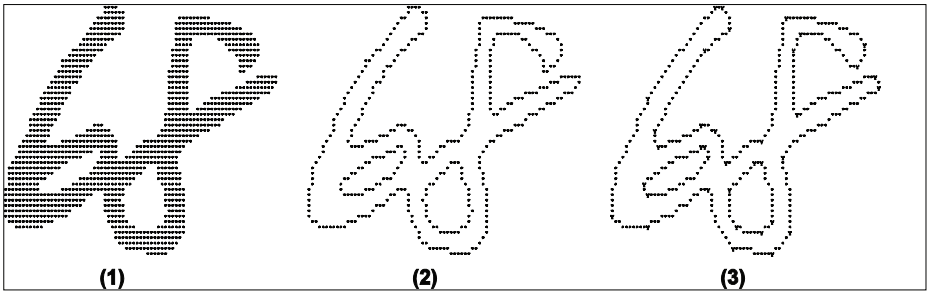


Figure 16. Handwritten digit 68 and its processing results



of four examples are shown in Figures 15(3,6) and 16(3).

The smoothed contour can be converted to a set of lines that consist of ordered pixels. Our task here is to approximate the contour of an image using straight lines. We call this procedure linearization of a contour.

Definition of Linearizing Line Based on Difference Codes

Suppose that a linearized line consists of some ordered pixels and its chain code set is:

$$c_k^{\ln} = \{c_k^{\ln}[0], c_k^{\ln}[1] \ldots c_k^{\ln}[i], \ldots c_k^{\ln}[n_k^{\ln} - 1]\}, \quad (11)$$

where k is represented as contour k of an image, ln as line ln of contour k, and $n_k^{\ln}$ as the total number of pixels contained in the line ln. A linearized line consists of ordered pixels that have the following property. If:

$$d_{ij} = c_k^{\ln}[i] - c_k^{\ln}[j] \quad (i = 0, \ldots . [n_k^{\ln} - 1]),$$
$$(j = 0, \ldots . [n_k^{\ln} - 1]), \quad (12)$$

Then:

$$| d_{ij} | \leq 1 \bmod 8 \ (i = 0, \ldots . [n_k^{\ln} - 1]),$$
$$(j = 0, \ldots . [n_k^{\ln} - 1]) \quad (13)$$

Therefore, a linearized line contains only two elements whose chain codes meet Equation 13; they are chain codes 0, 1, 2, 3, 4, 5, 6, or 7, respectively. We call two elements the element code; they are represented by *cdir1* and *cdir2*, respectively. For example, if *cdir1* is chain code 6, then *cdir2* is only chain code 7 or 5 based on Equation 13. In terms of the previous linearization definition and property of smooth following,

the algorithm of linearizing lines of a smoothed contour can be described as follows.

Criterion 3.1: A line includes two element codes so the first pixel of a new line is defined as the first element code, *cdir1.*

Criterion 3.2: If *cdir1* is found, the difference code, d_{cn}, between *cdir1* and the chain code of the next contour pixel, c_n, is calculated as:

$$d_{cn} = c_n - cdr1 \tag{14}$$

If $|d_{cn}| = 0$, then c_n is not the second element code of the new line and the next search is needed.

If $|d_{cn}| = 1$, then c_n is the second element code of the new line, *cdir2.*

Criterion 3.3:

1. If two element codes, *cdir1* and *cdir2,* are found, the difference code d_{c1n} between *cdr1* and c_n and the difference code d_{c2n} between *cdir2* and c_n are calculated as:

   $$d_{c1n} = c_n - cdr1 \tag{15}$$

   and

   $$d_{c2n} = c_n - cdr2 \tag{16}$$

2. If $|d_{c1n}| = 2$ or $|d_{c2n}| = 2$, then a new line is found. In this case, the chain code c_n of pixel *n* in the smoothed contour is the first element code of the next new line, and the contour pixel is the first pixel of the new line.
3. If both $|d_{c1n}| < 2$ and $|d_{c2n}| < 2$, then the pixel *n* belongs to the same line and next search is needed.

The starting pixel of a contour should be the first pixel, which belongs to the first linearized line. Its chain code is the first element code of the first linearized line, and the *cdr1* is chain code 5 based on Criterion 3.1 and the property of the smoothed contour described in Section-Smooth Following. The second element code (*cdr2*) of the first linearized line is chain code 4 or code 6 based on Criterion 3.2. From the second line, the starting pixel of other linearized lines can be determined based on Criterion 3.3. Other linearized lines of the smoothed contour can be found based on the previous criteria in order of pixels of the smoothed contour except the last pixel of the smoothed contour.

Criterion 3.4:

1. Based on the property of a smoothed contour, the chain code of the last pixel in the smoothed contour is chain code 4 based on Section-Smooth Following. When the last pixel is searched, the following three cases should be considered. If *cdir1* of the current line is chain code 3, then chain code 4 of the last contour pixel should be the same as the second element code (*cdr2*) of the current line or be the second element code of the current line. The last pixel of the smoothed contour belongs to the current line. The current line is the last line of a smoothed contour.
2. If *cdir1* of the current line is chain code 4, the last pixel of the smoothed contour belongs to the current line. The current line is the last line of the smoothed contour.
3. If *cdir1* or *cdir2* of the current line is chain code 2, then the last pixel of the smoothed contour belongs to a new line. The new line only consists of the last pixel and has one element code, chain code 4. Therefore, the current line is not the last line of the smoothed contour.

A smoothed contour can be linearized based on the previous algorithm, and its related chain code set $c_k^{\ln}$ and *x* and *y* coordinate sets of linearized lines can be found. Two handwritten digits 7 in Figure 15 are taken from the NIST database. Their linearization result is shown in Figures 15(1,4), and the starting point of each linearized line is represented by character "Y." The element codes *cdir1* and *cdir2* of the first line of Figure

15(3) are chain codes 5 and 4 based on Criterion 3.3, respectively. It contains 22 points. The element codes *cdir1* and *cdir2* of the second line of its contour are chain codes 6 and 7, respectively. It contains two points. They can be found in Figure 15(3). For the handwritten digit 68 in Figure 16(1), its linearizing result is shown in Figure 16(3).

STRUCTURAL POINTS OF BINARY IMAGE CONTOURS

For description and recognition of contour shape, one important feature is the morphological structures change (structure segment) of contours. In fact, humans recognize object image based on a series of morphological structures change of contours in the object image and some prior information. Some methods have been developed in the past (Fu, Yan & Huang, 1997; Moktarian & Mackworth, 1992; Suters & Yan, 1994). There are three problems in these algorithms: (a) extracted features (structure segments) are uncompleted; (b) there is no direction change of features; and (c) it is not easy to form a series of feature vectors. The reason is that linearization method-based difference codes are not developed.

Morphological Structural Point of Contours

Linearized lines and feature description are described in the last section based on difference codes. The linearized lines make morphological structural points of contours possible. There is a morphological change between two neighboring linearized lines (structure segments). Some special points are defined to represent these morphological changes along the contours. These special points are called the structural point of contours. The structural points not only describe the convex or concave change of line segments but also determine the change is in the direction of which chain code along the image contour. Therefore, these structural points can be used to represent convex or concave segments along the contour and the relation between the feature points and the object *S*. Their definition and detection are based on the structure patterns of element codes of two lines, which are shown in Figure 17. Assume that *line[ln]* is the current line, that *line[ln-1]* is the previous line, and that *line[ln+1]* is the next line along a contour. These lines are produced in the contour linearization procedure.

Definition 1. The convex point in the direction of code 4 (represented with "Λ").

If the element codes 3, 4, and 5 occur successively as a group of neighborhood linearized lines, then the point is a convex point in the direction of code 4. Two cases of the convex points in the direction of code 4 can be defined (represented with a small circle) in Figure 17(1):

1. If the first element code *cdir1* of *line[ln]* is code 4, the second element code *cdir2* is code 5, and the direction chain code of the last pixel of *line[ln-1]* is code 3; then the first pixel of the current line *line[ln]* is a convex point in the direction of code 4.
2. If the element code *cdir1* or *cdir2* of *line[ln]* is code 3, the direction chain code of the last pixel of *line[ln]* is code 4, and the first element code *cdir1* of *line[ln+1]* is code 5; then the last pixel of the current line *line[ln]* is a convex point in the direction of code 4.

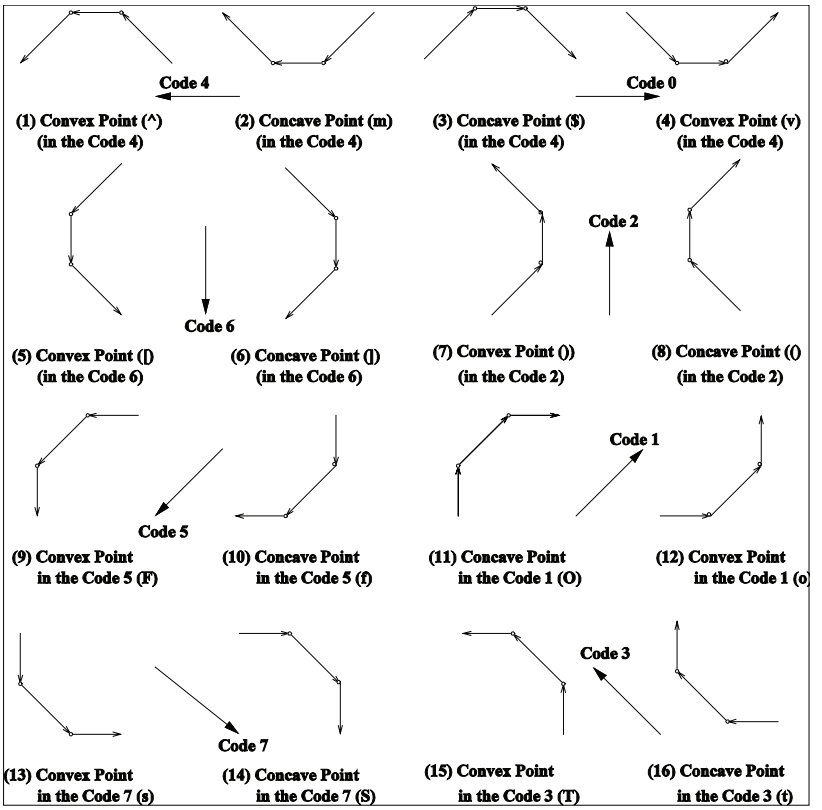
Definition 2. The concave point in the direction of code 4 (represented with the character "m").

If the element codes 5, 4, and 3 occur successively as a group of neighborhood linearized lines, then the point corresponding to code 4 is a concave point in the direction of code 4.

Two cases of the concave points in the direction of code 4 can be defined (represented with a small circle) in Figure 17(2):

1. If the first element code *cdir1* of *line[ln]* is code 4, the second element code *cdir2* is

Figure 17. Structural patterns of structural points



code 3 and the direction chain code of the last pixel of *line[ln-1]* is code 5; then the first pixel of the current line, *line[ln]*, is a concave point in the direction of code 4.

2. If the element code *cdir1* or *cdir2* of *line[ln]* is code 5, the direction chain code of the last pixel of *line[ln]* is code 4, and the first element code *cdir1* of *line[ln+1]* is code 3; then the last pixel of the current line *line[ln]* is a concave point in the direction of code 4.

Definition 3. The convex point in the direction of code 0 (represented with the character "v").

If the element codes 7, 0, and 1 occur successively as a group of neighborhood linearized lines, then the point is a convex point in the direction of code 0.

Two cases of the convex points in the direction of code 0 can be defined (represented with a small circle) in Figure 17(3):

1. If the first element code *cdir1* of *line[ln]* is code 0, the second element code *cdir2* is code 1, and the direction chain code of the last pixel of *line[ln-1]* is code 7; then the first pixel of the current line *line[ln]* is a convex point in the direction of code 0.
2. If the element code *cdir1* or *cdir2* of *line[ln]* is code 7, the direction chain code of the last pixel of *line[ln]* is code 0, and the first element code *cdir1* of *line[ln+1]* is code 1; then the last pixel of the current line *line[ln]* is a convex point in the direction of code 0.

Definition 4. The concave point in the direction of code 0 (represented with the character "$").

If the element codes 1, 0, and 7 occur successively as a group of neighborhood linearized lines, then the point is a concave point in the direction of code 0.

Two cases of the concave points in the direction of code 0 can be defined (represented with a small circle) in Figure 17(4):

1. If the first element code *cdir1* of *line[ln]* is code 0, the second element code *cdir2* is code 7, and the direction chain code of the last pixel of *line[ln-1]* is code 1; then the first pixel of the current line *line[ln]* is a concave point in the direction of code 0.
2. If the element code *cdir1* or *cdir2* of *line[ln]* is code 1, the direction chain code of the last pixel of *line[ln]* is code 0, and the first element code *cdir1* of *line[ln+1]* is code 7; then the last pixel of the current line *line[ln]* is a concave point in the direction of code 0.

Similar to definitions 1- 4, other structural points can be defined and found as follows: the convex point in code 6 ("["), the concave point in code 6 ("]"), the concave point in code 2 ("("), the convex point in code 2 (")"), the convex point in code 5 ("F"), the concave point in code 5 ("f"), the convex point in code 1 ("o"), the concave point in code 1 ("O"), the convex point in code 3 ("T"), the concave point in code 3 ("t"), the convex point of code 7 ("s"), and the concave point in code 7 ("S") (see Figure 17).

These structural points describe the convex or concave change in different chain code directions along the contour of a binary image. Sixteen different characters in these figures are used to represent different structural points. It is necessary to note that if the processed line is the last line, then there must be an upper convex point. It is the last pixel of the last line. This is because the direction chain code of the last point in a smoothed contour is code 4, and the direction chain code of the first pixel in a smoothed contour is code 5 based on the property of the smooth following contour.

Also, although the represented contour shape of point "Λ" is the same as that of concave point "$", two types of structural points have different morphological structural properties. Point "Λ" is convex, but point "$" is concave. Similarly, the represented contour shape of point "m" is the same as that of point "v," but the morphology of point "m" is concave, and that of point "v" is convex. There is similar property for structural points "[" and "(", points "]" and ")", points "F" and "O," points "f" and "o," points "T" and "S," and points "t" and "s" (see Figure 17). Therefore, structural points not only describe the shape of contour segments but also determine the convexity or concavity of contour segments. It is clear that the series of structural points of a contour can describe the shape of the contour.

Experiment Results

If the contours of an image are smooth followed and linearized, then all structural points of these linearized contours can be extracted based on the previous definitions and algorithms. The processing results of four sample images are shown in Figures 18-21.

For the outer contour in Figure 18, there is a series of structural points:

"Λ"→ "F" → "[" →"s" →"v" (convex)→ "$" →"S"→ "]" → "f" (concave) → "F"→ "["→ "s"→ "v" (convex)→ "$" → "S" → "]" → "f" → "m"(concave) → "Λ"→ "F"→ "[" → "s"→ "v"→ "o"→ ")" → "T"→ "Λ" (convex).

Each convex or concave change consists of a group of convex or concave structural points, respectively. For example, the first convex change of the previous series consists of convex structural points "Λ", "F", "[", "s", and "v".

For the outer contour in Figure 19, the series of structural points is:

"Λ"→"F"→"["→"s"(convex)→"S"→"]"→"f" (concave)→"F"→"["→"s"(convex)→"S"→"]"→

Figure 18. Original image, contour, smooth following, linearization, and structural points of one handwritten digit 3

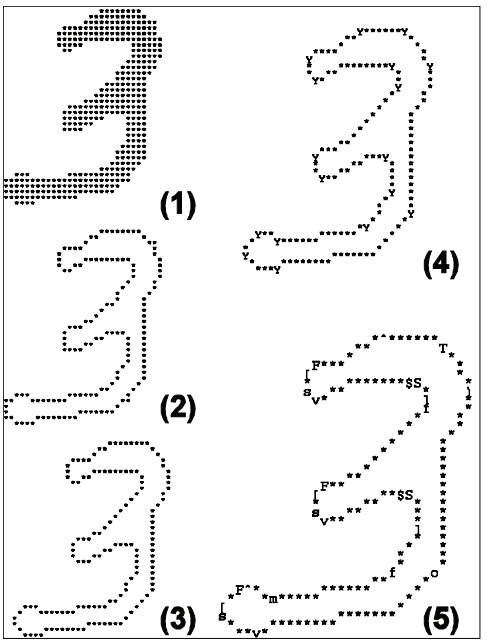


Figure 19. Original image, contour, smooth following, linearization, and structural points of another handwritten digit 3

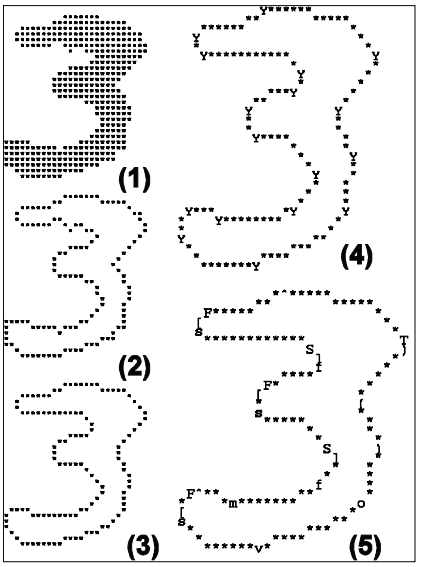
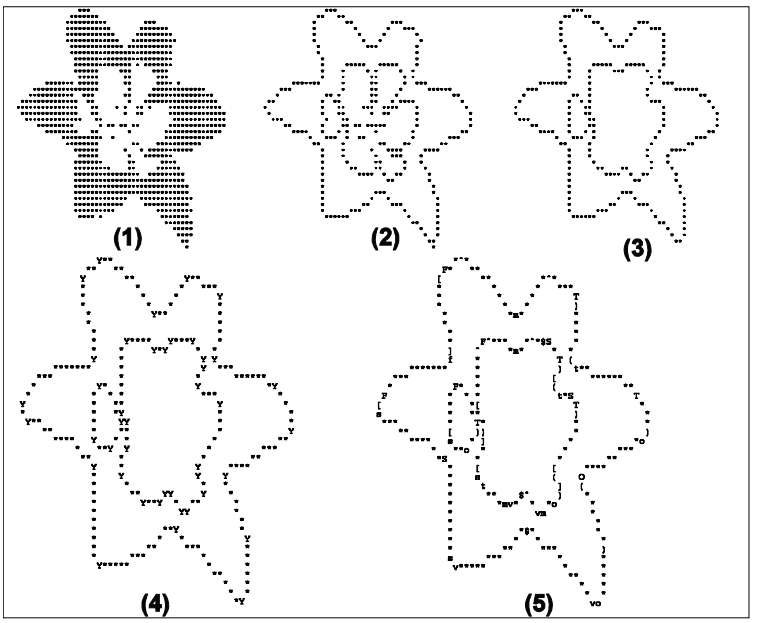


Figure 20. Binary image, contour and smooth following, linearization, and structural points of one lily flower image



"f"→"m"(concave)→"Λ"→"F"→"["→"s"→"v" →"o"→")"(convex)→"("(concave)→ "T'→"Λ" (convex).

There are similar concave changes that contain a group of points "S" and "]" in both of the previous series of structural points. If recognized object images are handwritten digits based on prior information, then two handwritten digits are recognized as digit 3. The reason is that there is such a morphological structure pattern (two groups of structural points "S" and "]") on the smoothed contours of all types of digits 3 (both of printed and handwritten digits 3). For handwritten digits, handwritten digit 7 (with one horizontal line) contains two groups of structural points "S" and "]." However, digit 7 can be distinguished with handwritten digits 3 based on other location features of structural points.

For the outer contour in Figure 20, the series of structural points is:

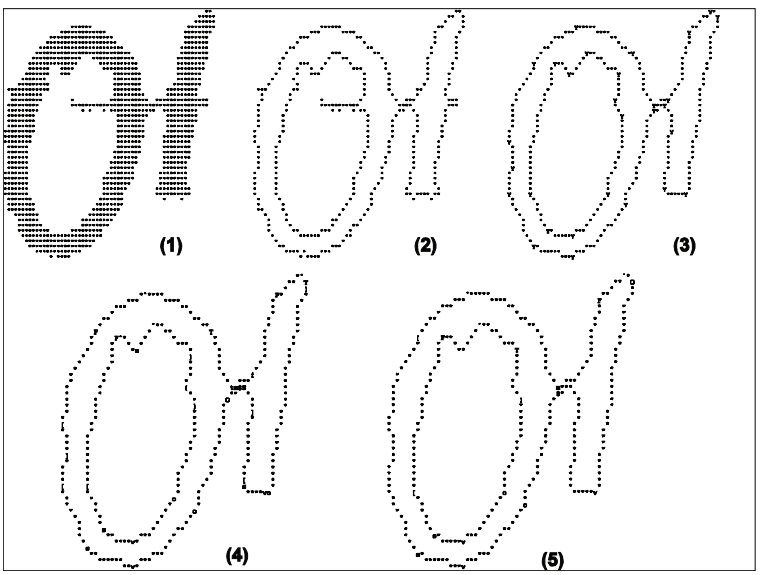
"Λ"→"F"→"["(convex)→"]"→"f"(concave)→ "F"→"["→"s"(convex)→"S"(concave)→"s"→ "v"(convex)→ "$" (concave)→"v"→"o"→")" (convex)→"("→"O"(concave)→"o"→")"→"T" (convex)→"t"→"("(concave)→")"→"T"→"Λ" (convex)→ "m" (concave)→ "Λ" (convex).

It is clear that the outer contour is six angles, because there are six pairs of convex and concave change. If the flower image is recognized, it can be recognized as a lily flower. For most sorts of lily flowers, there are six petals that are constructed of six angles.

Another example is shown in Figure 21. Its original image and contour and smooth contour images are shown in Figures 21(1-3), respectively. For the outer contour in Figure 21(4), the series of structural points is:

"Λ"→ "F" (convex)→ "f""→ "m" →"t" (concave)→ "T"→ "Λ"→ "F"→"["(convex)→"]" (concave)→"s"→ "v"→"o"(convex)→"O"→"$" →"S"→"]"(concave)→"["→ "s"→ "v"→ "o"→ ")"→"T"→ "Λ"(convex).

Figure 21. Binary image, contour and smooth following, linearization, and structural points of connected handwritten digits 0 and 1



The outer contour mainly consists of two convex shapes between which there are two concave shapes. One concave shape is mainly in the direction of chain code 4, and another concave shape is in the direction of chain code 0.

ANALYSIS AND RECOGNITION OF BROKEN HANDWRITTEN DIGITS

Machine recognition of handwritten digits plays an important role in intelligent document processing systems. When handwritten digits contain broken strokes and spurious segments in their fields, it is difficult for most recognition methods to deal with them, including both structural and statistical approaches (Hu, 1998; Lee, 1996; Malaviya & Klette, 1996; Shi, 2002; Yan, 1993, 1994). Broken handwritten digits are caused by segmentation and threshold errors of noisy digits, the writing style, or the tools used. Some examples of broken handwritten digits are shown in Figures 22(1-4). They are taken from the U.S. National Institute of Science and Technology (NIST) database (Garris & Wilkinson, 1992).

Some methods have been developed to resolve the problem in the past. The general stroke extension procedure for reconstructing broken digits may create links that actually do not exist, thereby creating an additional problem (Whichello & Yan, 1996). Another approach is to use variable sized masks to increase the size of the link (Whichello & Yan, 1996). However, with this algorithm, if the distance between two points of a broken digit (without a link) is less than that between two points (with a link), it will result in a bad reconstruction of the broken digit.

This section presents an efficient method of reconstructing and recognizing broken handwritten digits. First, we describe some algorithms for preprocessing broken digits before introducing the set of structural rules. Second, the broken points of broken digits are preselected based on the minimum distance between two "end" points of skeletons that belong to the main and adjunctive segments, respectively, and some correction rules of the preselected broken points are discussed based on the structure analysis and comparison of digit fields. The diagram of reconstructing broken digits is shown in Figure 23. Finally, a summary is given.

Preprocessing Algorithms of Broken Digits

In this section, a set of preprocessing algorithms is introduced. Here, suppose the starting point for the process is the upper-left corner point (the

Figure 22. Sample images of broken handwritten digits and final processed results

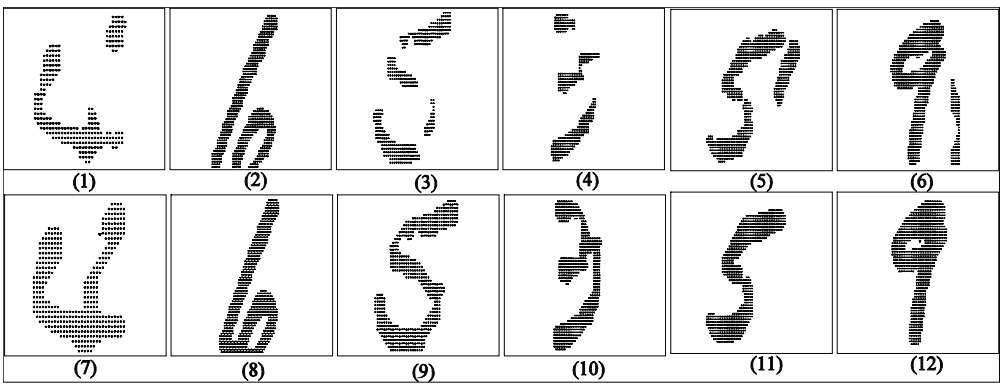
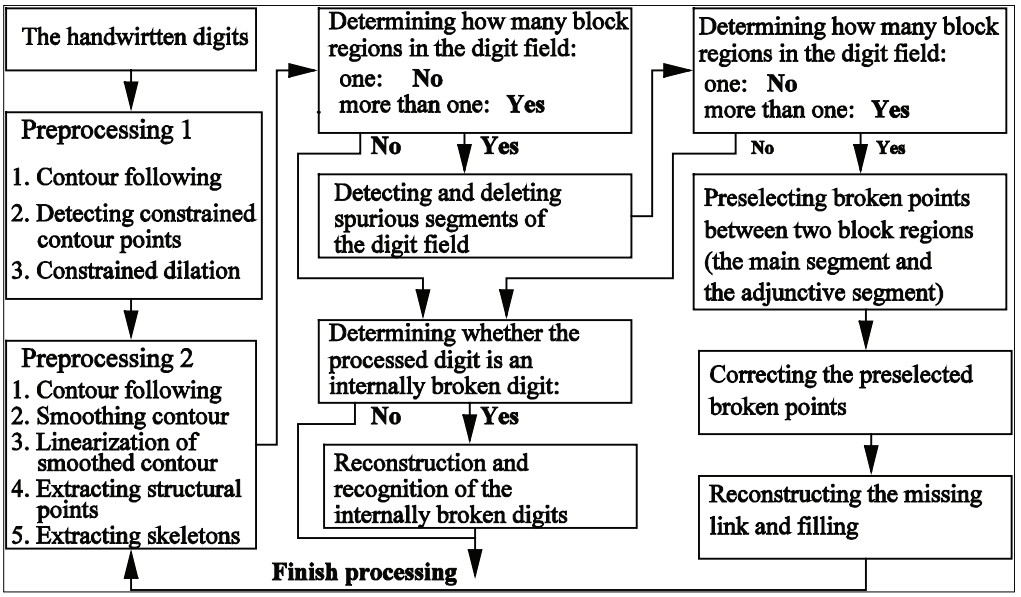


Figure 23. Block diagram of the algorithm for the reconstruction of handwritten broken digits



upper-left point) of the object, Freeman codes are used, and the contour of image is 8-connected.

Filling and Constrained Dilation

In many cases, there are some small gaps and spurious holes in a broken handwritten digit. The dilation process can be used to fill the small gaps, and it can be interpreted as follows:

$$A \oplus B = \left\{ x \,|\, [(\hat{B})_x \cap A] \subseteq A \right\} , \qquad (17)$$

where A is the processed image, B is the structural element in dilation, and x is the displacement (Garris & Wilkinson, 1992). The structural element for dilation is shown in Figure 24(1).

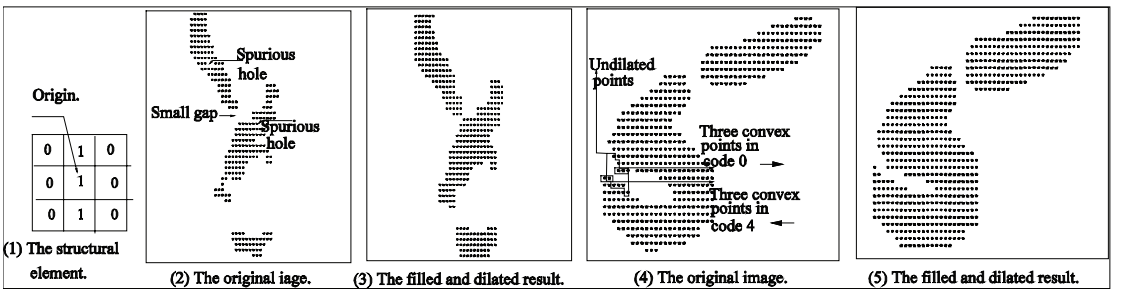
The structural element for dilation is shown in Figure 24(1). If the input image is processed by such a dilation once, then the small gaps between the two regions and the small holes can be filled. However, there is a danger of forming links that do not exist. Such an example is shown in Figure 24(4), which shows that the small gap of the lower block region in Figure 24(4) should not be linked. We therefore need to impose some conditions on the dilation operation to avoid such cases. Two constrained conditions are as follows.

Condition 1. If a candidate contour point is a convex point in the direction of code 4, calculate the distance between it and other contour points whose y coordinate is less than the candidate's y coordinate. If these distances are less than or equal to 3, then this candidate point and corresponding contour points found are the points that should not be dilated.

Condition 2. If a candidate contour point is a convex point in the direction of code 0, calculate the distance between it and other contour points whose y coordinate is greater than the candidate's y coordinate. If these distances are less than or equal to 3, then this candidate point and its corresponding contour points found are not the dilated points.

For example, there are three convex points in the direction of code 0 and two convex points in the direction of code 4 in Figure 24(4). Based

Figure 24. Constrained dilation and two examples



on the previous conditions, the undilated points in Figure 24(4) can be determined. The dilation operation with the previous constraining conditions can be represented as follows:

$$A \oplus B = \left\{ x \mid [(\hat{B})_x \cap A] \subseteq A \right\}_c , \qquad (18)$$

where *c* represents the conditions. The processed results of the original sample images in Figures 24(2,4) are shown in Figures 24(3,5) based on the previous filling and constrained dilation algorithm, respectively. We can see that spurious holes in Figure 24(2) are filled, and some spurious spaces in Figures 24(2,4) are smoothed, but the small gap of the lower region in Figure 24(4) is not linked.

Preprocessing

Contour following and smooth following, linearization and description features of contour, and detection of structural points are used to preprocess the broken handwritten digits. Also, the project filling method is used to fill the smooth following contours, and then the skeletons of the filled images are extracted. In general, broken points are located between two skeletons that belong to two neighboring regions. Therefore, it is necessary to extract the skeleton of broken digits. The smoothed contours are filled and then thinned by a parallel thinning algorithm (Zhang & Suen, 1984). An example image of a thinned broken digit is shown in Figure 12. The original image and its smoothed contours are shown in Figures 12(1-2), respectively, and its skeleton is shown in Figure 12(3). The skeleton can further be smoothed based on some patterns as shown in Figure 13(1). Based on our algorithm, the skeleton (see Figure 12(3)) is smoothed, and the final skeleton is shown in Figure 12(4).

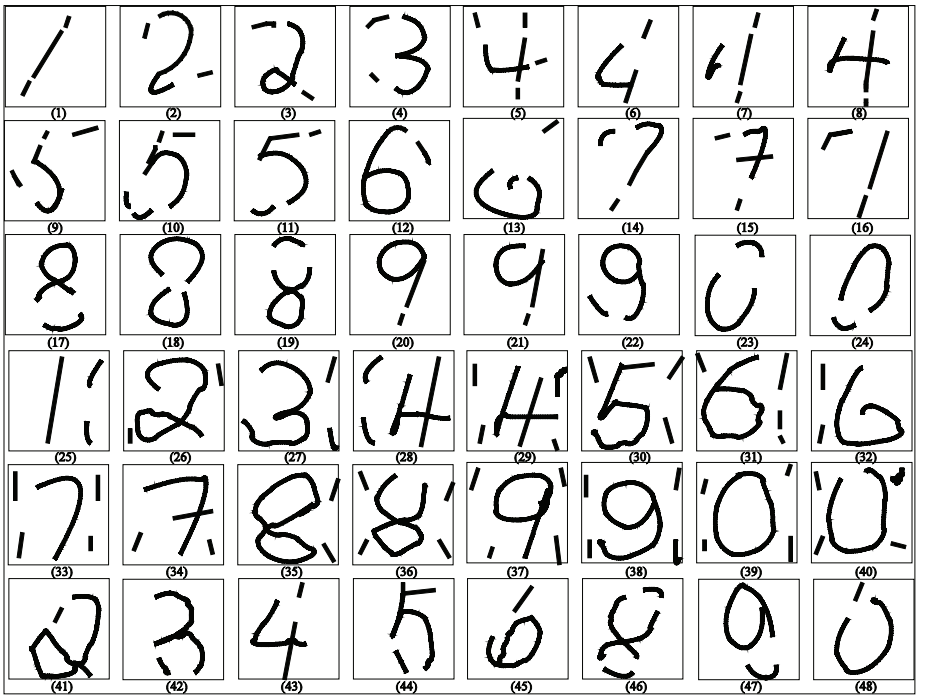
Broken Points of Broken Digits

If all possible morphological structural patterns of broken digits are compared and analyzed, then all related detection rules can be constructed and described. Some structure patterns are shown in Figure 25.

If the contour length of a segment is largest, then the segment is defined as the main segment of the digit field, and the other segments are defined as the adjunctive segment. In general, if a segment is spurious, it is an adjunctive segment, and there are no crossing points between the extension lines of the main and adjunctive segments. We can determine whether an adjunctive segment is spurious or not based on the morphological structure of main and adjunctive segments. Here, suppose that the adjunctive segment is not spurious.

Basically, a large gap is caused by the broken stroke of a digit. Broken points are defined as those region points between which there is a gap.

Figure 25. Some structures of relation between the main and adjunctive segments



If we can correctly determine the broken points of a digit, we can bridge the gap between them and reconstruct the broken digit. A new approach for finding broken points is based on the analysis and comparison of structure and skeleton of the digit region.

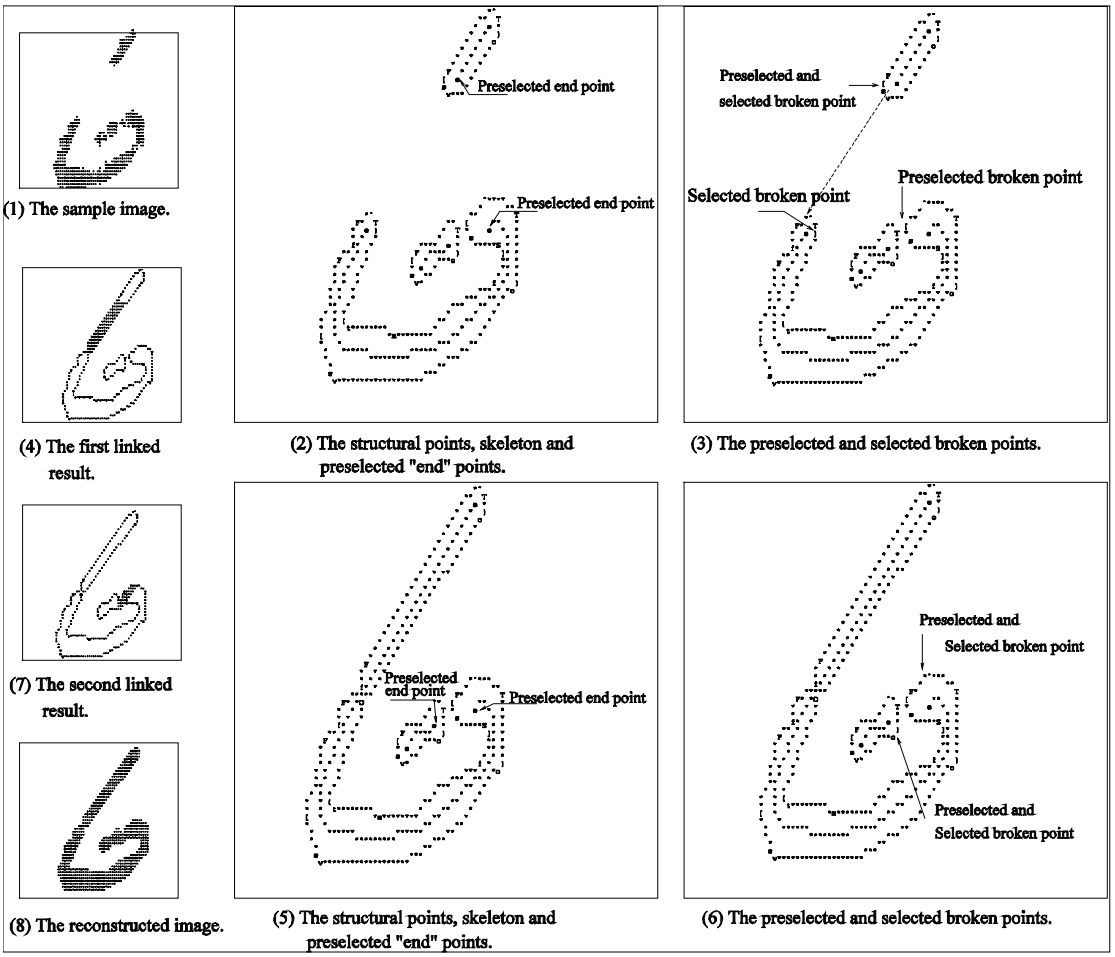
Preselection of Broken Points

In most cases, if one gap can be correctly determined, then there is a minimum distance between the two "end" points that belong to two different skeletons. The two skeletons are neighboring; one belongs to the main segment, and another one belongs to the adjunctive segment. We call these two "end" points the preselected "end" point. Furthermore, if there is a minimum distance between one structural point (only structural points "[," "v," "Λ," or ")" being considered) and the found "end" point, then such a structural point is a broken point. Two found "end" points correspond to two broken points. Based on this description, the preselection criteria for broken points can be summarized as follows:

1. Extract the structural points and skeleton of digit regions by algorithms in Section-Structural Points.
2. Calculate the distance between the two "end" points that belong to different block regions (the main and adjunctive segments, respectively).
3. Select the two "end" points between which there is minimum distance.
4. For each "end" point selected by Step 3, if there is minimum distance between the "end" point and a structural point (only structural points "[,', "v," "Λ," or ")" being considered) that belongs to its corresponding block region, then the structural point is a preselected broken point.

For example, one original image is shown in Figure 31(1), the extracted structural points

Figure 26. The selection of broken points and correction rule 4



and skeleton are shown in Figure 31(2) based on Step 1, two preselected "end" points are selected based on Steps 2, 3 (shown in Figure 31(2)), and the preselected broken points are selected based on Step 4 (see Figure 31(3)). It can be seen that the preselected broken points in Figure 31(4) are correct.

Two segments of a broken digit are selected to preselect the broken points. One is the main segment, and another one is its neighboring adjunctive segment. If the adjunctive segment is above the main segment, it is chosen to preselect the broken point. Otherwise, this lower adjunctive segment is chosen to preselect the broken point. When a gap is bridged, the number of segments of a broken digit can be decreased by one. For example, selected broken points are shown in Figure 31(3), the linked image is shown in Figure 31(4), and we can see that the number of segments is changed from 4 to 3. After all gaps are linked, it only contains one block region, and the final result is shown in Figure 31(12). If there is only one block region in the processed digit, the digit does not contain an outer gap. In this case, there is only one segment, and its recognition can be done based on the algorithm described in Figure 36.

Correction of Preselected Broken Points

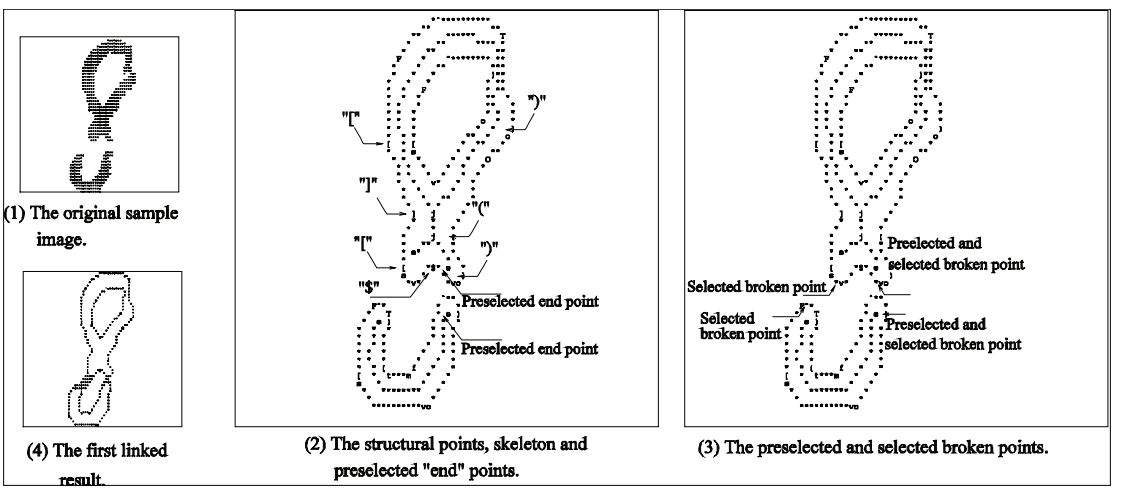
In most cases, the preselected broken points are correct. However, some false selection of broken points may occur, especially if the distance between the two preselected broken points is large. For example, an original image is shown in Figure 26(1).

Its preselected "end" points and preselected broken points (see Figures 26(2-3)) are based on the preselection algorithm of broken digits. We can see that the distance between the two preselected broken points (shown in Figure 26(3)) is large compared to the height of the whole digit. In this example, the preselected broken points are false. For the sample image in Figure 27, only one pair of broken points is preselected (see Figures 27(2-3)), but in fact, there is another pair of broken points. The other three examples are shown in Figure 28, and the preselected broken points are also false (see Figures 28(2-3,6-7,10-11)). In these cases, it is necessary to correct the preselected broken points. The available prior knowledge is that the processed object is handwritten digits. We can construct all possible morphological structure patterns of broken handwritten digits whose preselected broken points are not correct. Some of examples are shown in Figures 25(41-48).

In fact, although a digit is broken, its segments contain the morphological structural information of the digit, especially in its main segment. Correction rules are constructed based on the description analysis and comparison of these morphological structure patterns and on the shape recognition of the main segment, and then determined whether the preselected broken points are reasonable. If not, the correction methods are proposed. Some correction rules are described as follows.

Correction rule 1. If (1) the main segment is the uppermost block region and is the shape of digit 3; (2) the neighboring adjunctive segment is at the lower-left corner of the main segment, then (a) one broken point is a structural point (points "Λ," "[,""v," or ")") of the main segment, and there is a minimum distance between the broken point and the lower-right "end" point of the skeleton of the main segment; (b) another broken point is a structural point (points "Λ," "[," "v," or ")") of the neighboring adjunctive segment, and there is a minimum distance between the broken point and a selected "end" point of the skeleton of the adjunctive segment. This "end" point is selected as follows: if the outer contour of the adjunctive

Figure 27. The selection of broken points and Correction rule 5



segment contains point "(" (concave change in code 2), the selected "end" point is the lowermost "end" point of its skeleton. Otherwise, the selected "end" point is the rightmost "end" point of its skeleton.

One example is shown Figure 28 (2). It meets correction rule 1(1-2), and the preselected broken points need to be corrected. The selected broken points are determined based on correction rule 1(2)(a-b) (shown in Figure 28(3)). Based on the selected broken points, the broken digit is reconstructed and shown in Figure 28 (4).

Correction rule 2. If (1) the main segment is over the adjunctive segment and there is one point "S" in the series of structural points of the main segment; (2) there is no internal contour in the main segment; and (3) the skeleton of the main segment contains one "junction" point which is at the bottom of the skeleton, then (a) one broken point is a structural point (points "Λ," "[," "v," or ")") of the main segment, and there is a minimum distance between the broken point and the right one of the two lower "end" points of the skeleton of the main segment; (b) another broken point selected is the same as that of correction rule 1(2)(b).

One sample is shown in Figure 28(6); it meets correction rule 2(1-3), and the preselected broken points need to be corrected. The selected broken points can be found based on correction rule 2(3)(a-b) and shown in Figure 28(7). Based on the selected broken points, the broken digits reconstructed and shown in Figure 28(8).

Correction rule 3. If (1) the main segment is over the adjunctive segment, and there are one point "S" and point "(" in the series of structural points of the main segment; (2) there is no "junction" point on the skeleton of its neighboring adjunctive segment, and there is no point "t" (concave change in code 3); (3) the right "end" point of the skeleton of the adjunctive segment is on the right of the lower "end" point of the skeleton of the main segment, then two broken points can be selected as follows: (a) one broken point is a structural point (points "Λ," "[," "v," or ")") of the main segment, and there is a minimum distance between the broken point and the right "end" point of the skeleton of the adjunctive segment; (b) another broken point is a structural point (points "Λ," "[," "v," or ")") of the adjunctive segment, and there is a minimum distance between the broken point and the right "end" point of the skeleton of the adjunctive segment.

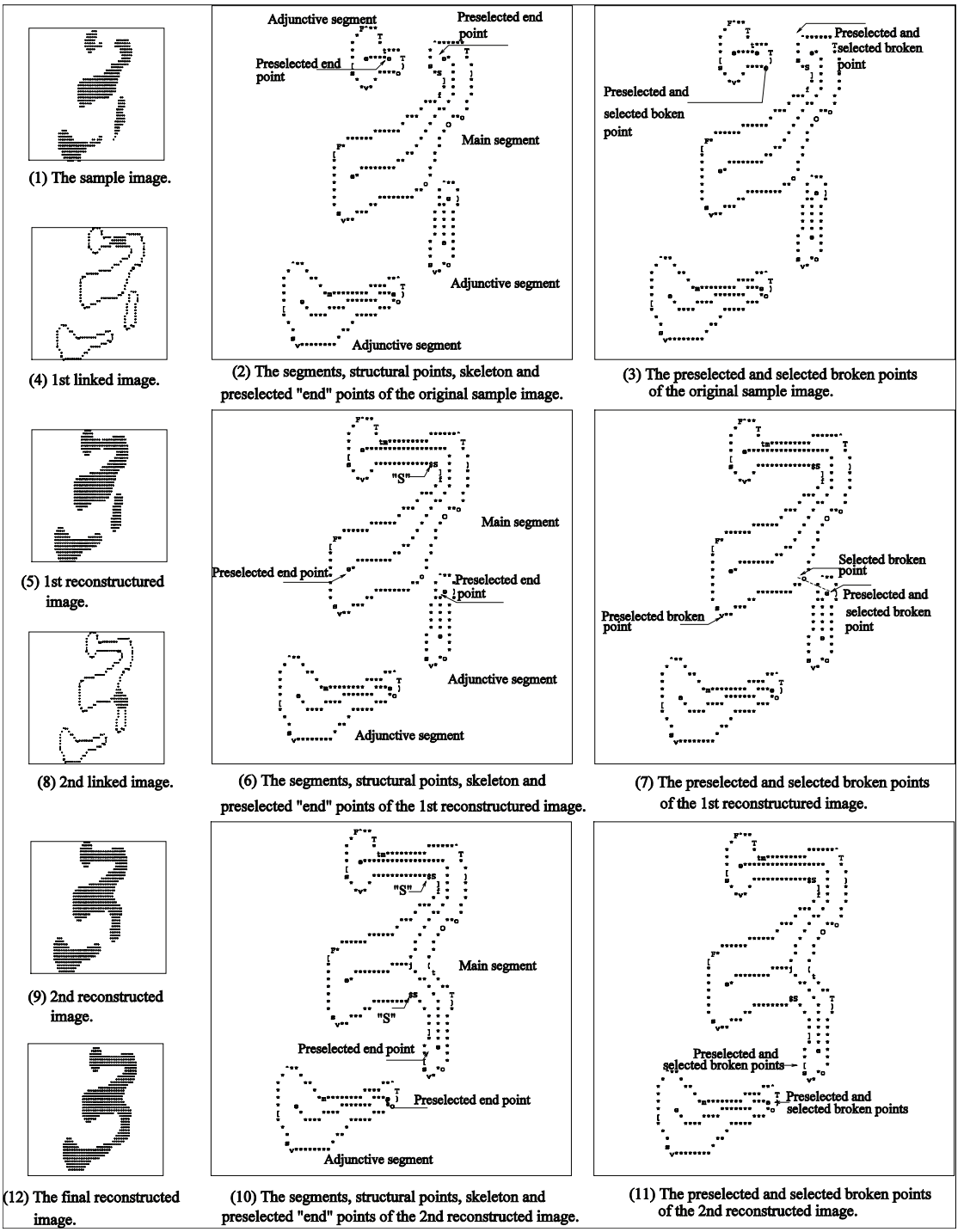
One sample is shown in Figure 28(10); it meets correction rule 3(1-3), and the preselected broken points need to be corrected. The selected broken points can be found based on correction rule 3(3)(a-b) and shown in Figure 28(11). Based on the selected broken points, the broken digit is reconstructed and shown in Figure 28(12).

Sometimes, false broken points may occur, especially if the distance between the two preselected broken points is large. Such an example is shown in Figure 26. In this case, it is necessary to correct the preselected broken points. Determining whether the preselected broken points are correct can be based on the structural and skeleton extending analysis of the digit.

Correction rule 4. If (1) the main segment is under the adjunctive segment and is the one shape of digit 0; (2) the adjunctive segment is the shape of digit 1; (3) there exist crossing points between the down extension lines of the adjunctive segment and up extension lines of the left "end" part of the main segment, then two broken points can be selected as follows: (a) one broken point is a structural point (points "Λ," "[," "v," or ")") of the main segment, and there is a minimum distance between the broken point and the lower "end" point of the skeleton of the main segment; (b) another broken point is a structural point (points "Λ," "[," "v," or ")") of the adjunctive segment, and there is a minimum distance between the broken point and "end" point of the skeleton of the adjunctive segment.

One sample is shown in Figure 27(2); it meets correction rule 4(1-3), and the preselected broken points need to be corrected. The selected broken

Figure 28. The selection of broken points and correction rules 1, 2 and 3



points can be found based on correction rule 4(3)(a-b) and shown in Figure 27(3). Based on the selected broken points, the broken digit is reconstructed and shown in Figure 27(5).

In some cases, it is possible that there are two missing links. Some examples are shown in Figures 25(17-19, 23-24). A practical example is shown in Figure 27. Its structural points and skeleton are shown in Figure 27(2). The related correction rule is based on the morphological structure analysis and shape recognition of the main and adjunctive segments. Here, the main segment is one shape of digit 8 because: (a) there exists a series of structural points on its outer contour: "[" → "]" → "[" → "$" → ")" → "(" → ")"; (b) there is one upper internal contour. The related correction rule can be described as follows.

Correction rule 5. If (1) the main segment is over the adjunctive segment and is the shape of digit 8 (see the previous description); (2) the skeleton of the main and adjunctive segments all contain two "end" points, then two pairs of broken points can be selected as follows: (a) one pair of selected broken points is two preselected broken points; (b) another pair of selected broken points is: one broken point is a structural point (points "Λ," "[," "v," or ")") of the main segment, and there is a minimum distance between the broken point and another lower "end" point of the skeleton of the main segment; another broken point is a structural point (points "Λ," "[," "v," or ")") of the adjunctive segment, and there is a minimum distance between the broken point and another upper "end" point of the skeleton of the adjunctive segment. One sample is shown in Figure 27(2); it meets correction rule 5(1-2). The selected pair of broken points is two preselected broken points based on correction rule 5(2)(a), and another pair of broken points is based on correction rule 5(2)(b) and corresponds to the left pair of "end" points of the main and adjunctive segments. They are shown in Figure 27(3). Based on the selected broken points, the broken digit is reconstructed and shown in Figure 27(4).

One morphological structure pattern of broken digit is shown in Figure 25(7), and a practical sample is shown in Figure 29. In this case, the preselected "end" and broken points are shown in Figures 29(2-3) based on the algorithms described in the preselection algorithm of broken digits. The preselected broken points are incorrect.

The corresponding correction rule can be constructed as follows.

Correction rule 6. If (1) the main segment is the shape of digit 1; (2) there are structural point "O" (concave change in code 1) but no structural points "S" (concave change in code 7), "m" (concave change in code 4), or "$" (concave change in code 0) on the outer contour of the adjunctive segment; (3) the adjunctive segment is on the middle left side of the main segment, then two broken points can be selected as follows: (a) one broken point is the lower-right point ")" of the adjunctive segment; (b) another broken point is on the outer contour of the main segment, between which and the selected broken point (by the previous Step (a)), there is a minimum horizontal distance.

One sample is shown in Figure 29; it meets correction rule 6(1-3), and the preselected broken points need to be corrected (see Figure 29 (2)). The selected broken points can be found based on correction rule 6(a-b) and shown in Figure 29(3). Based on the selected broken points, the broken digit is reconstructed and shown in Figure 29(4).

Other structural patterns of unreasonable preselected broken points can be found from possible broken structural patterns. Some of examples are shown in Figures 25(6,18-19,23-24,41-48).

Similar to the previous six correction rules, another 22 corresponding correction rules can be set up based on the analysis of morphological structural patterns, shape recognition, skeleton analysis, and geometrical features of broken digits. Based on these correction rules, unreasonable preselected broken points can be corrected.

Based on these procedures, the broken digits can be reconstructed until there is only one block

Figure 29. The selection of broken points and Correction rule 6

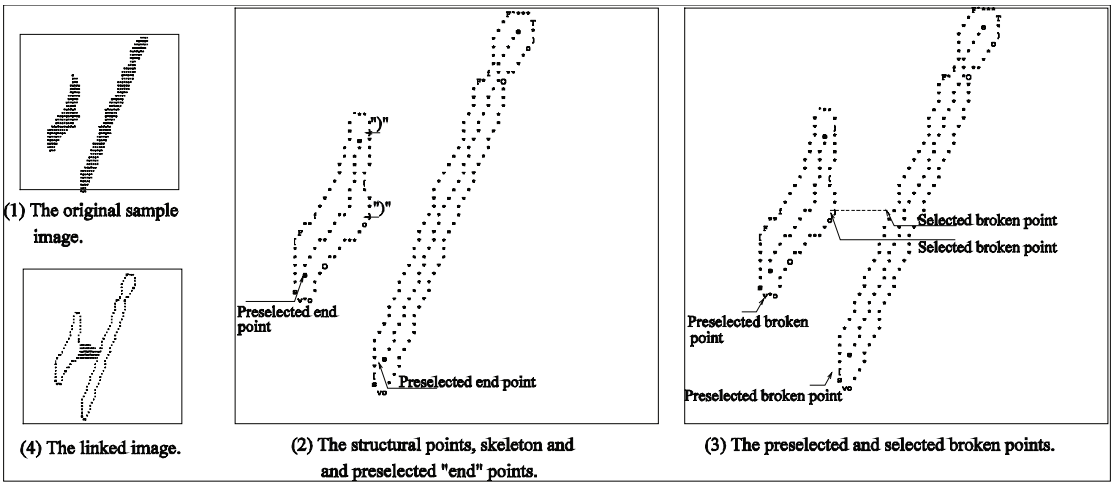


Figure 30. Example 1 of multi-broken handwritten digits

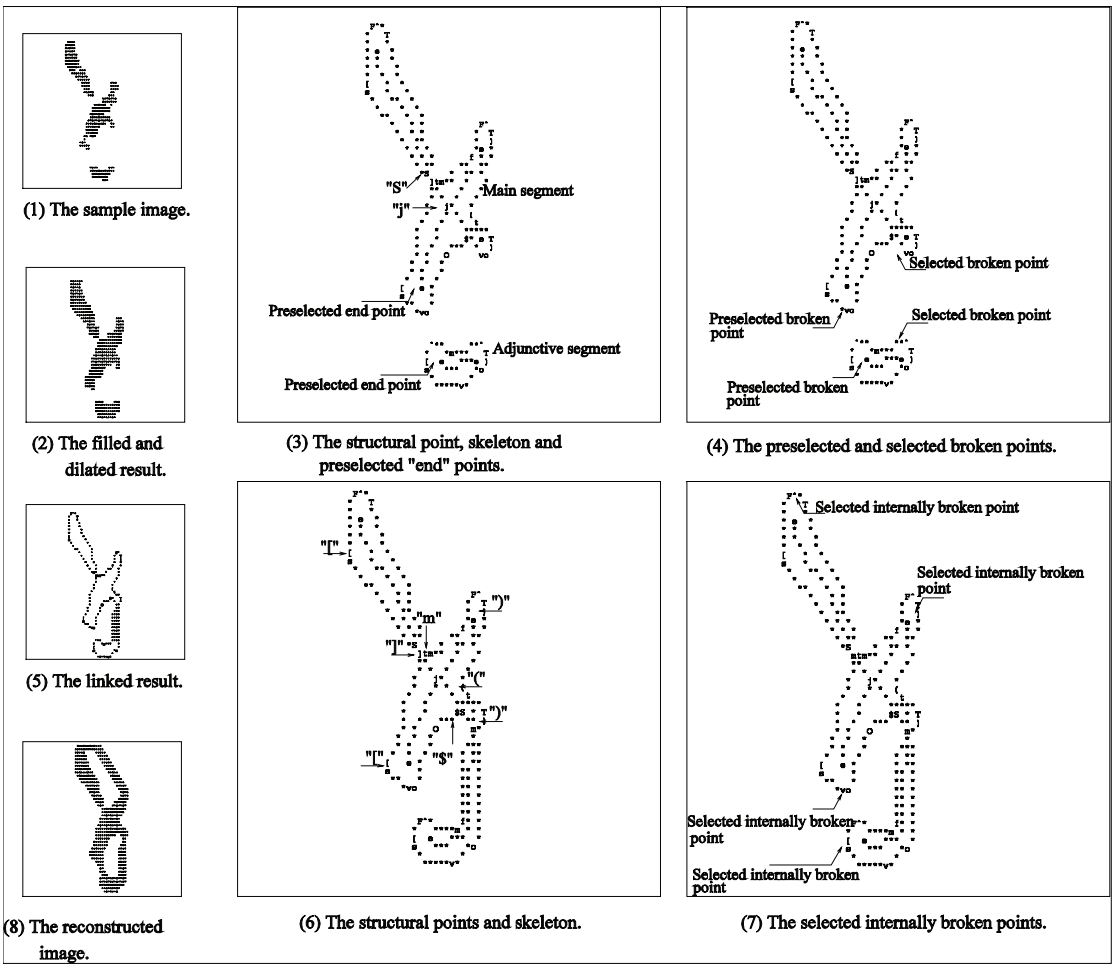
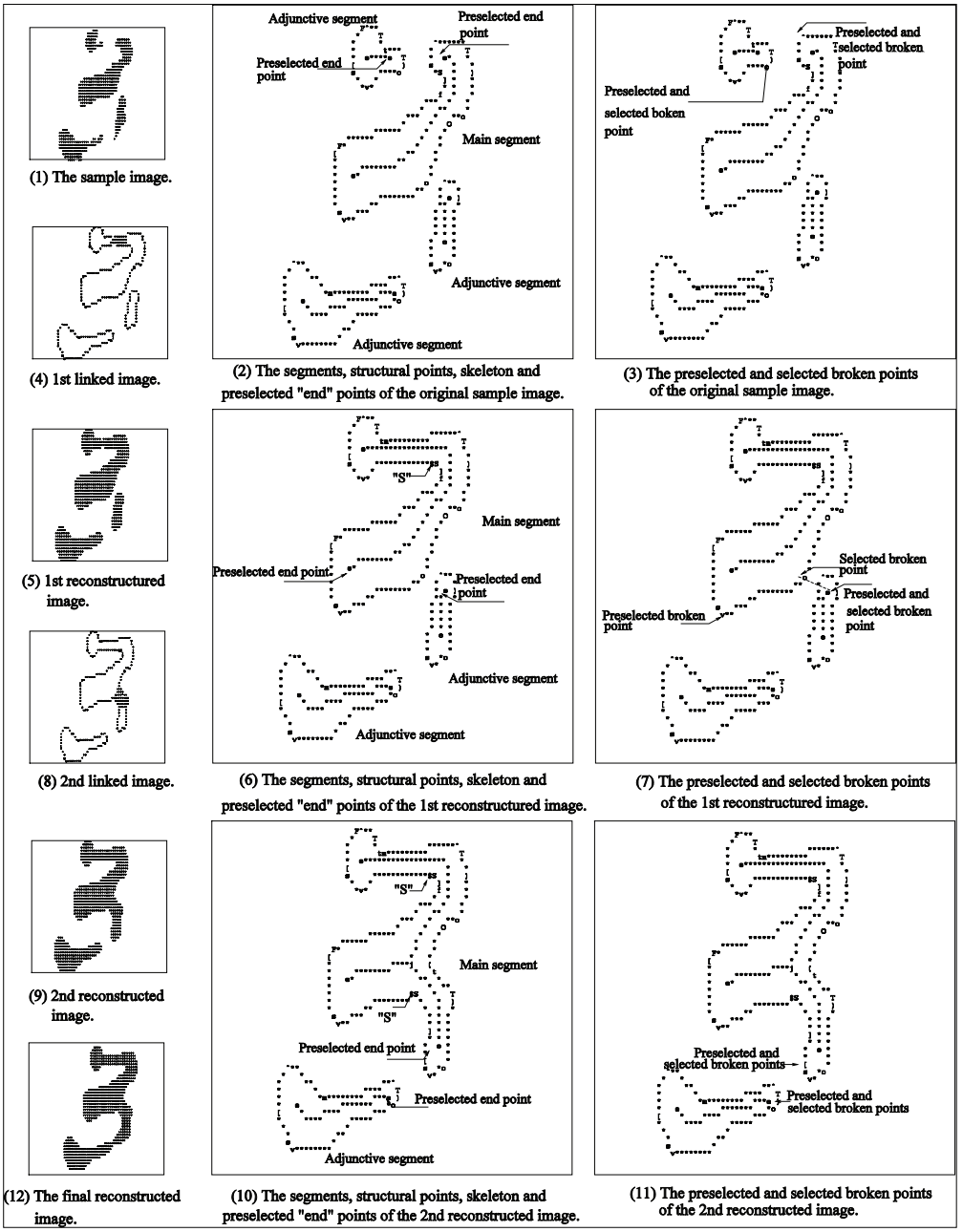


Figure 31. Example 2 of multibroken handwritten digits



region for these broken digits. Two multibroken digits are shown in Figures 30(1) and 31(1).

Based on the constrained dilation algorithm, the upper segment is linked with its neighboring segment, and the processed result is shown in Figure 30(2). We can determine that the newly formed segment (upper block region) is the main segment, and another segment is the adjunctive segment and a nonspurious segment based on the structure of segments. The preselected "end" points and broken points can be selected based on the algorithms described in this section; they

are shown in Figures 30(3-4). However, based on correction rule 1, the preselected broken points need to be corrected. Here, the main segment is recognized as the shape of digit 3. The selected broken points are found, and the first link result is shown in Figure 30(5). Furthermore, the broken points can be found and shown in Figure 30(7) based on the reconstruction algorithm of internally broken digits, which is introduced in the section Analysis and Recognition. The final reconstruction result is shown in Figure 30(8).

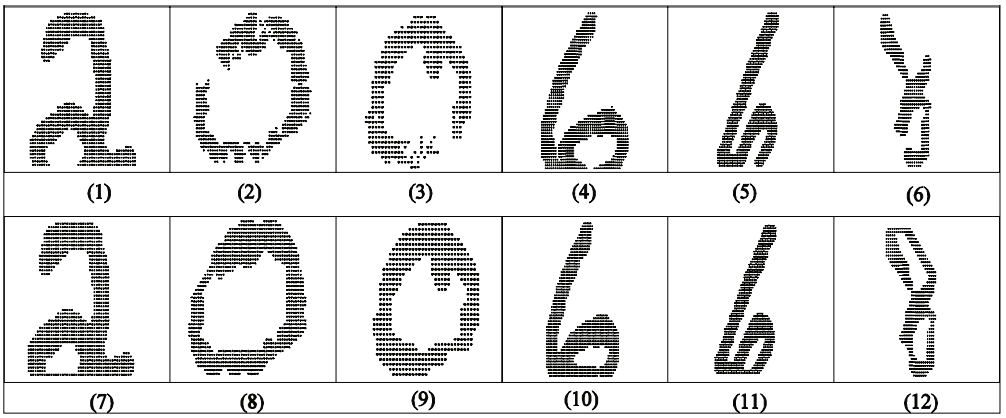
The original image of another sample that has four segments is shown in Figure 31(1). Based on our algorithms, the upper segment is an adjunctive segment, and its neighboring segment is the main segment. The preselected broken points between the main and adjunctive segments are correct, and the missing link can be bridged (see Figure 31(4)). Using our procedures repeatedly for the image in Figure 31(5), we can determine that the newly formed upper segment is the new main segment, and its neighboring segment is the adjunctive segment. The preselected "end" points and broken points can be determined (see Figures 31(6-7)) based on the preselection algorithm of broken digits. Based on correction rule 3, the preselected broken points need to be corrected, and the selected broken points can be determined (see Figure 31(7)). The linked result is shown in Figure 31(8). Using our procedures repeatedly for the image in Figure (9), we can determine that the second-formed upper segment is the main segment, and its neighboring segment is the adjunctive segment. The preselected "end" points and broken points can be determined (shown in Figures 31(10-11)) based on the preselection algorithm of broken digits. In this case, the preselected broken points are the same as the selected broken points (see Figures 31(10-11)). The final reconstructed result is shown in Figure 31(12).

Reconstruction and Recognition of Internally Broken Handwritten Digits

The handwritten broken digit, shown in Figure 30(5), contains internally broken strokes. An internal gap structure is unavoidably caused by a broken internal contour during extraction of the digit fields. Some example images are shown in Figure 32.

These internally broken digits can be reconstructed and recognized based on some structural patterns. The internally broken digits are mainly made by the broken internal contours of handwritten digits 0, 2, 4, 6, 8, and 9. In some cases, a digit is broken internally because of writing styles. Such examples are shown in Figures 33(5,14,17,21). We call the location where a digit is broken a "cut."

Figure 32. Sample images and their processed results of internally broken digits



There are some internally broken handwritten digits in the NIST database. Two sample images are shown in Figures 34(1,5). Most recognition methods are unable to classify them. In order to reconstruct and recognize these digits, we need to analyze and describe the morphological structure of internally broken digits. If we can distinguish between different structures in Figure 33, we can recognize these internally broken digits and reconstruct them. The shape description of internally broken digits can be represented by a structural feature vector (SFV), which is a series of ordered structural points, the geometrical locations of some structural points, and skeleton structure.

The extraction of SFV is from the first line to the last line along the processed outer contour. Suppose the height and width of a processed digit is *h* and *w*, respectively.

The algorithm for the reconstruction and recognition of internally broken digits is described next.

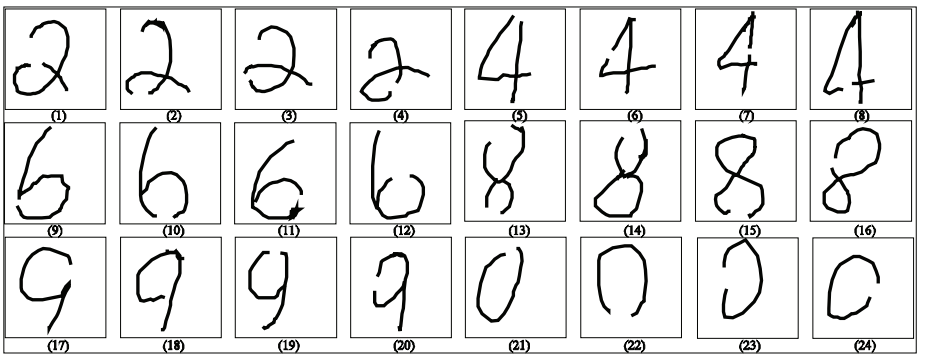
Case 1

One pattern model of digit 8, which includes two "cuts," is shown in Figure 33(13). A real sample is shown in Figure 34. Its original image is shown in Figure 34(1). It is preprocessed by the algorithms described in the section Smooth Following, Linearization, and Structural Points, and its extracted structural points and skeleton are shown in Figure 34(2). Its morphological structure belongs to one model of Figure 33(13). The reconstruction and recognition of this model are based on the following algorithm:

1. There exists the following SFV in the image (see Figure 34(2)): "["→"]" →"[" →"$"→ "v"→ "O" →"Λ"→ "m" →"Λ."
2. Suppose the *y* coordinate of the point "m" in the selected SFV is h_m, then $h_m > h/4$. This means that the selected point "m" (the concave change in code 2) is near the middle of the digit.
3. Suppose the *y* coordinate of the point "$" in the selected SFV is $h_\$$, then $h_\$ < 3h/4$. This means that the selected point "$" (the convex change in code is near the middle of the digit.
4. There are only four "end" points in the skeleton of the digit.
5. The *y* coordinates of two lower "end" points are greater than 3h/4. That means that the two "end" points are near the bottom of the digit.

Steps 1–5 describe the morphological structure of the model in Figure 34(2). Therefore, if Steps a–e are true, this digit is digit 8, which has two "cuts," and two pairs of broken points can be selected as follows:

Figure 33. Pattern models of internally broken digits



1. Two lower "end" points of the skeleton are extended in the direction of their chain codes (see Figure 34(2)).
2. If there is a minimum distance between the extension point and the outer contour, the found point of the outer contour is a selected broken point.
3. In this way, two broken points can be selected based on the two lower "end" points of the skeleton (see Figure 34(2)).
4. Similarly, another pair of broken points can be selected based on the two upper "end" points of the skeleton.

Based on this algorithm, the selected broken points are shown in Figure 34(2) in our sample. The final reconstructed result is shown in Figure 34(4) based on the selected broken points. Another sample of digit 8 is shown in Figure 32(6). It belongs to another model of Figure 33(13). The difference between it and Case 1 is that its SFV is "["→"]" →"[" →"$" →")"→"("→ ")"→"m" →"Λ". Its reconstruction result is shown in Figure 32(12).

Case 2

The internally broken digit 6, which includes a right "cut," is shown in Figure 33(11). A real example is shown in Figure 34. Its original image is shown in Figure 34(5). It is preprocessed by the algorithm described in the sections Smooth Following, Linearization and Structural Points, and the structural points and the skeleton can be found and shown in Figure 34(6). Its morphological structure belongs to one model of Figure 33(11). The reconstruction and recognition of this model are based on the following algorithm:

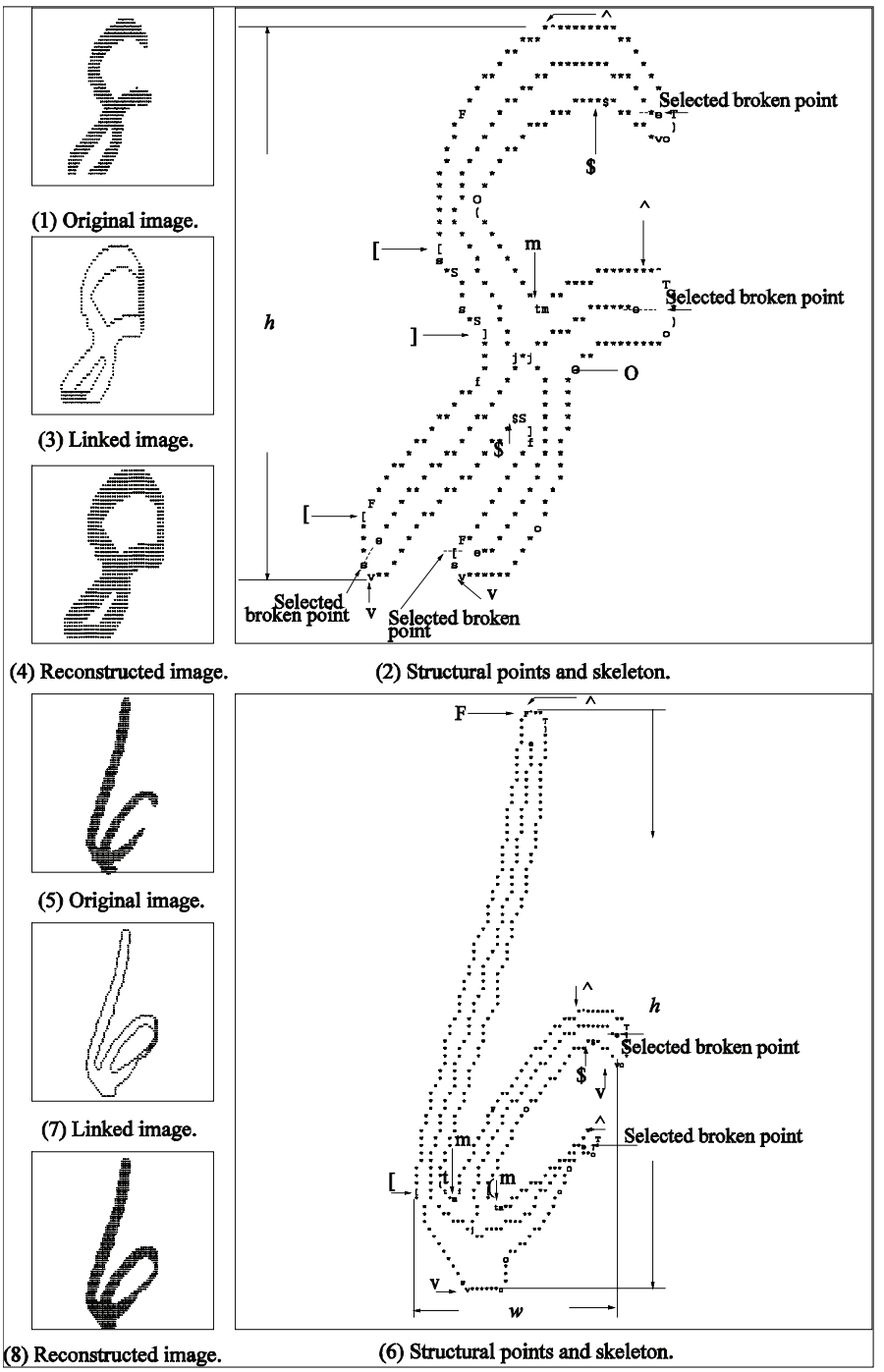
1. There exists the following SFV in the image (see Figure 34(6)): "F"→ "["→ "v"→ "Λ"→ "m"→ "("→"$"→ "v"→ "Λ"→ "m"→ "t"→ "Λ."
2. Suppose the *y* coordinate of the first point "m" in the selected SFV is h_{m1}, then h_{m1}>3h/4. This means that the first selected point "m" (the concave change in code 2) should be near the bottom of the digit.
3. Suppose the *y* coordinate of the point "$" in the selected SFV is $h_{\$}$, then $h_{\$}$>*h*/3. This means that the selected point "$" (the convex change in code 6) is near the middle of the digit.
4. Suppose the *y* coordinate of the second point "m" in the selected SFV is h_{m2}, then h_{m2}>*h*/2. This means that the second selected point "m" (the concave change in code 2) is below the middle of the digit.
5. There are only three "end" points in the skeleton of the digit.
6. The *x* coordinates of two lower "end" points are greater than *w*/2, where *w* is the width of the digit. This means that the two lower "end" points are near the right-hand side of the digit.

These steps describe the morphological structure of the model in Figure 34(6). Therefore, if steps 1-6 are true, this digit is digit 6, which has a right "cut." Its broken points are the extension points of the two lower "end" points in the direction of their chain codes, which can be selected (see Figure 34(6)) based on the similar algorithm described in Case 1. The final reconstructed image is shown in Figure 34(8). Similarly, we can deal with other models in Figure 33. Based on our algorithm, the internally broken digits can be reconstructed and recognized. Other samples are shown in Figures 32(1-6), and their reconstructed results are shown in Figures 32(7-12).

Experimental Results of Reconstruction

All image data are taken from the NIST database. In order to set up all structural patterns, 210 patterns (sample images) of broken digits were

Figure 34. Two sample models of the internally broken handwritten digits



selected from the broken digit set (1,236 broken handwritten digits) of the training set (11,639 handwritten digits of SD3). Two hundred ten sample images are selected based on all possible structural patterns, which are used to recognize the shape of segments, to determine whether a segment is spurious, and to detect whether the preselected broken points need to be corrected. Some structural patterns are constructed based on possible shape structure analysis and comparison of broken digits, and these patterns are shown in Figure 25. For the training data set (1,236 broken handwritten digits), a reconstruction rate of 97.35% with 0.33% substitution and 2.32% rejection is achieved. For the testing dataset, 3,631 broken handwritten digits are extracted from 42,698 extracted handwritten digits of SD3, and there is a reconstruction rate of 96% with 0.36% substitution and 3.14% rejection.

The method of linking broken digits with variable sized masks (Whichello & Yan, 1996) cannot be used to link digits with large gaps. Such examples are shown in Figures 28(5), 26(1), 29(1), and 22(1,4). Also, if a digit is broken, there is at least one gap. However, if there is a gap, it is possible that the gap is the normal structure of the digit and should not be linked. After a mask size is selected, it is still possible for both true and normal gaps to be linked. If a large sized mask is selected for a large gap, then the reconstructed broken digit will be distorted (Whichello & Yan, 1996). If the general stroke extension method is used, it is difficult to select the threshold of the extending link length. If this threshold is small, then a big gap cannot be linked. Such examples are shown in Figures 28(5), 26(1), 29(1), and 22(1,4). If the threshold is large, then it is possible that the normal gaps of broken digits are linked incorrectly. Such an example is shown in Figure 28(5). Also, the two methods cannot deal with the problem of spurious segments of a digit field.

Recognition of Handwritten Digits Based on Reconstruction and ONNC

The input images of optimized nearest neighbor classifier (ONNC) (Yan, 1993) are original and normalized images in the NIST database. If the recognition algorithm of ONNC is used, a total of 11,639 extracted handwritten digits of SD3 (taken from the NIST database) are used as the training dataset, a total of 42,698 extracted handwritten digits of SD3 are used as the testing data set, and the recognition rate is 96.1%. The improved recognition algorithm of handwritten digits is that the input images of ONNC are preprocessed based on the methods described in the sections Smooth Following, Linearization and Structural Points, and this section. Therefore, the quality of the input images of ONNC is improved. After removing the spurious segments of handwritten digits and reconstructing broken handwritten digits, the algorithm, ONNC, is used to recognize these handwritten digits. For the same testing dataset (42,698 extracted handwritten digits of SD3), the recognition rate of 99.7% is achieved. This is because the quality of the input images is improved after the input images are preprocessed, spurious segments of the input images are removed, and the broken input images are reconstructed. Another testing dataset totally consists of 11,495 extracted handwritten digits of SD7. For the testing dataset from SD7, if ONNC is used, the reconstruction rate is 94.87%. After the reconstruction of the testing dataset, if the classifier ONNC is used, the recognition rate is 97.62%. For the new test dataset, the reconstruction rate is decreased and the rejection rate increased because some new broken morphological structure patterns are not considered in our methods for the testing data. In order to improve the processing result, more training data need to be used for the classifier ONNC, and more new structure patterns need to be used to construct some new reconstruction rules.

One efficient recognition method is to use gradient and curvature of gray scale image (Shi, 2002). By using this method, its recognition rate

is 99.49% and 98.25% for 223,124 handwritten digits of SD3 and 58,646 handwritten digits of SD7, respectively. Our recognition rate of 99.7% is a little higher than 99.49%, but only 42,698 extracted handwritten digits of SD3 are used in our algorithm. For testing the dataset of SD7, the method (Shi, 2002) is more efficient. One possible approach is to use our method to process the dataset, and then the method (Shi, 2002) is used for recognition.

If the recognition algorithm is an ONNC on a Sun Sparc 2 workstation, the average time of recognizing one handwritten digit is 0.02 seconds where the handwritten digits mainly consist of one segment. However, if the reconstruction of broken handwritten digits and processing of spurious segments are used, the average time of recognizing one broken handwritten digit with spurious segments is 0.19 seconds. Therefore, although recognition rate is improved, total recognition time is increased after using our reconstruction algorithm.

Recognition of Handwritten Digits Based on Morphological Structures

The structure recognition is used to recognize handwritten digits (Hu, 1998). However, most structure recognition algorithms cannot recognize those handwritten digits that contain the spurious segments or are the broken digits. One new structure recognition algorithm is based on all possible morphological structures of handwritten digits that may contain the spurious segments or be the broken digits. The new structural recognition method is based on the following procedures: (1) preprocessing (smooth following, linearization-based chain codes, extraction of skeleton, and filling); (2) feature extraction (structural points, curvature, bend angles, etc.); (3) analysis and recognition of handwritten digits with spurious segments; (4) reconstruction and recognition of broken handwritten digits; and (5) recognition of handwritten digits with one segment.

The block diagram of the recognition method is shown in Figure 35. Some morphological structure patterns are shown in Figures 25(1-32). The segments in these images are analyzed and then determined whether they are spurious. If a segment is spurious, then it is removed.

After separation of spurious segments, there are three cases:

1. If the digit contains more than one segment, then it is a broken digit. Such examples are shown in Figures 25(1-16). In this case, the broken digit is reconstructed based on the algorithm described in this section.
2. If the digit only contains one segment and it is an internally broken digit, the digit is recognized based on the structure patterns of internally broken digits, which are shown in

Figure 35. The block diagram of recognition based on morphological structures

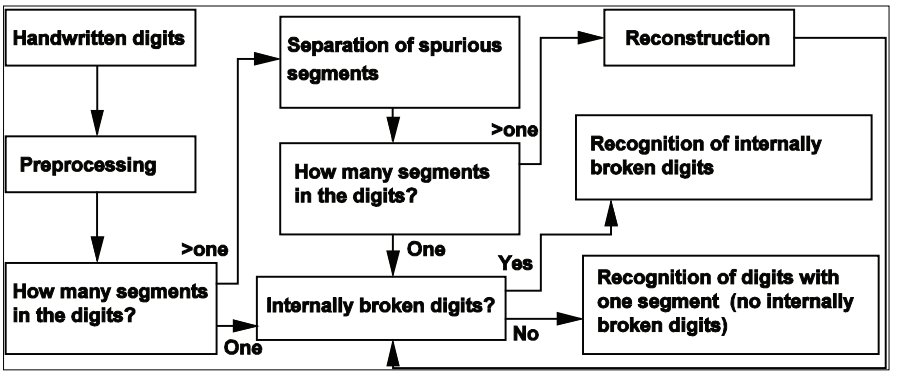


Figure 34. Based on these structure patterns, the relative recognition patterns can be constructed in terms of their series of structural points, geometric features (i.e., geometric location of some special structural points), and the number of internal contours.

3. If the digit only contains one segment and it is not an internally broken digit, the case is general. In fact, handwritten digits only contain one segment (except digit 5) in most cases.
4. The recognition of handwritten digits with one segment is based on all possible morphological structure patterns. The recognition of handwritten digits containing only one segment is based on the description of their morphological structure patterns.

The description of morphological structure patterns is based on:

- Series of structural points of outer contour (from starting point to end point of the contour)
- Geometrical location of structural points; the number of internal contours and their properties (geometrical location)
- The property and geometrical location of lines

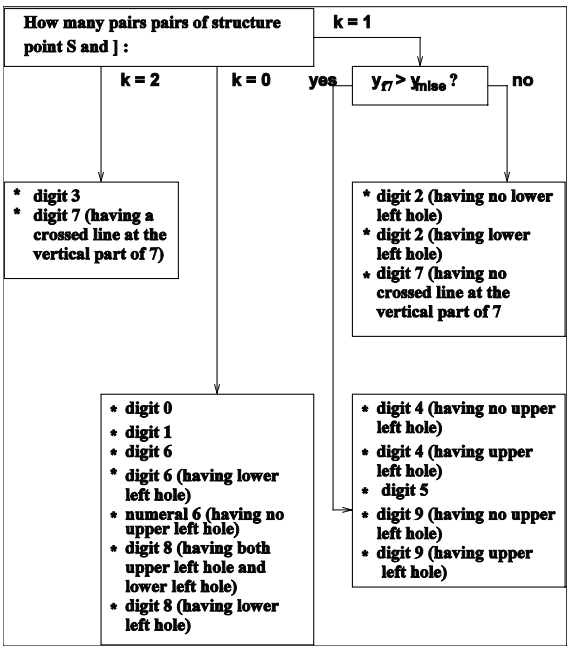
The classification is based on these descriptions (how many pairs of structural points "S" and "]," how many internal contours, etc.). For example, the shape recognition of digits 3 and 7 has been described, where two handwritten digits 3 in Figures 18-19 contain two pairs of structural points, "S" and "]." Most of handwritten digits can be considered as those cases in which the digit field only consists of one segment, and their morphological structure patterns can be constructed to recognize these handwritten digits based on the above algorithm description.

After reconstruction of broken handwritten digits and separation of spurious segments, the processed digits only consist of one segment. If the processed digits do not belong to internally broken digit structure, then the flow diagram of their recognition is shown in Figure 36. In Figure 36, k is the number of pairs of structural points "S," "[" (the concave change in the direction of codes 7 and 6), y_{f7} is the y coordinate of the first structural point "S" of series of structural points, and $y_{mlse}=(y_{led}-y_{lsu})/4 + y_{lsu}$ (y_{lsu} and y_{led} being the y coordinate of the uppermost and lowest position of the digit respectively). The inequality, $y_{f7} < y_{mlse}$, means that point "S" (the concave change in the direction of code 7) is at the upper of the digit. Based on all possible recognition rules and the previous processing procedures, general handwritten digits (with one segment) can be recognized.

Summary

A new and efficient method of reconstructing and recognizing broken handwritten digits has been developed. The preprocessing algorithms can be

Figure 36. The flow diagram of shape recognition of handwritten digits with one segment



used to fill and smooth some spurious holes and spaces and make the morphological structural analysis of broken digits possible. All algorithms of preselecting broken points and correcting some preselected broken points are described. Our new method is more efficient because it not only uses some features such as geometry and stroke extension information but also uses morphological structure and skeleton information of broken digits. Therefore, both small and large true gaps of broken digits are linked, and the normal gaps of broken digits are not based on our method. These newly developed algorithms of structural analysis are invariant to broken digit size, but not rotation in variant.

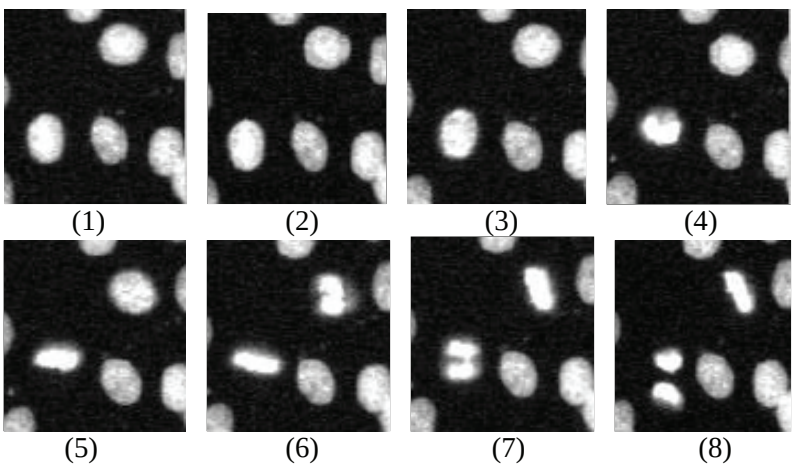
DYNAMIC ANALYSIS AND RECOGNITION OF HIGH CONTENT CELL-CYCLE SCREENING BASED ON MORPHOLOGICAL STRUCTURES

In order to assist scientists to understand the complex process of cell division or mitosis (Dunkle, 2003; Feng, 2002; Fox, 2003; Hiraoka & Haraguchi, 1996; Yarrow et al., 2003), analysis of cell-cycle screening by automated fluorescence microscopy is used. An essential task for cell-cycle screening is to measure cell-cycle progression (interphase, prophase, metaphase, and telophase), which can be identified by measuring nuclear changes. The most difficult task of such analysis (Chen, Zhou & Wong, 2007), (Pham, Tran, Zhou & Wong, 2005), (Pham, Tran, Zhou & Wong, 2006) is finding different stages that can be presented by nuclear size and shape changes during cell mitosis. One example, the nuclear migration during cell division (eight series of times), is shown in Figure 37, which is the section of large original microscope images.

The Morphological Structure of Various Cell Phases

In order to find the morphological structure of various cell phases, we can use the methods described in the sections Smooth Following, Linearization, and Structural Points to process and analyze these original microscope cell images. The description of image contour plays an important role in the shape analysis and recognition of image. Line segment, critical points, and their convexity and concavity are useful features to analyze the shape of image contour. Many methods and algorithms are developed for the description of contours in

Figure 37. One example of cell-cycle screening



the past (Fu et al., 1997; Moktarian & Mackworth, 1992; Yang & Yan, 2000). However, these descriptions cannot form series of sets, or the intercontour of a binary image cannot be processed based on these algorithms, which make the analysis and understanding of contour shape difficult. Also, no one uses difference code to describe and extract these series of features. The methods proposed can make it possible.

Binarization

The Ostu (1978) thresholding method is used to separate binary cell images in cell-cycle screening and is shown in Figure 38.

Smooth Following and Linearization

The contours in Figures 38(1-8) are followed smoothly and linearized based on the algorithms described in the section Smooth Following and Linearization. For example, the processed results of the bottom-left cell in the cell screen (Figures 38(1-8)) are shown in Figure 39, where the spurious points in contours are removed and character "Y" is the first point of each linearized line.

The Series of Structural Points of Cell Images

The series of structural points of cell images in Figure 39 can be found in Figure 40.

Figure 40 is based on the algorithm described in the section Structural Points. For the outer contour in Figure 40(1), the series of structural points is:

"Λ"→ "F"→ "["→"s"→ "v"→ "o"→ ")"→"T."

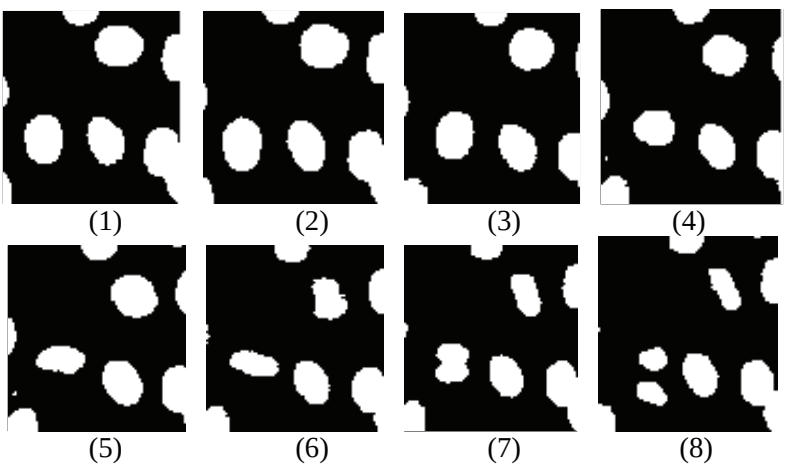
It is clear that the contour shape is convex polygon, where it can be approximately defined as an ellipse shape.

For the outer contour in Figure 40(7), the series of structural points is:

"Λ"→ "F" →"[" →"s"(convex)→ "S"→ "]" →"f" (concave)→ "F"→ "["→"s" → "v"→ "o"→ ")"(convex)→ "(" (concave)→ ")"→"T" (convex).

It is clear, that this contour shape consists of two pairs of convex and concave change. It can be approximately defined as a barbell shape.

Figure 38. Binarization of cell images in one cell-cycle screening (Figure 37)



Separation and Reconstruction of Touching Cells

There are some images of touching cells in a frame of microscope image. We can see that there are nine groups of touching cell images in Figure. 41(1-2), respectively.

Therefore, these groups of touching cell images make analysis and recognition of the cell images of cell-cycle screening automatically very difficult because the size and shape of some cells cannot be found. In order to resolve the problem, we need to determine which image is a group of touching cell images, and then find all separation points of the group of touching cell images. Finally, each touched cell image needs to be reconstructed.

Morphology Structures of Touching Cell Images

Based on the prior knowledge, the cell shape of cell-cycle screening images can approximate as an ellipse before it is divided. Therefore, if two or more cells are touched, there is one concave structural point at least on its outer contour. Also, its size is larger than that of one cell image, as a touching cell image consists of two or more cells. Some binarization images of touching cell images

Figure 39. Smooth following and linearization of contours of the bottom-left cell in the cell screen (Figure 37)

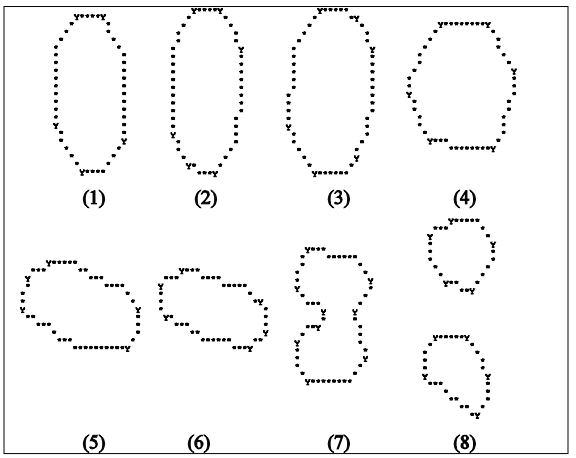


Figure 40. Cell morphological structures of different phases (the bottom-left cell in the cell screen) (Figure 39)

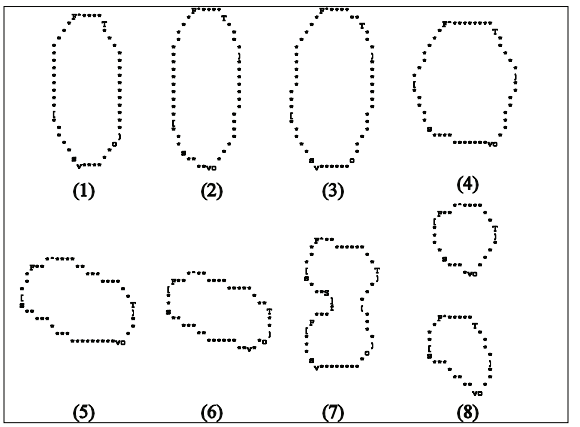
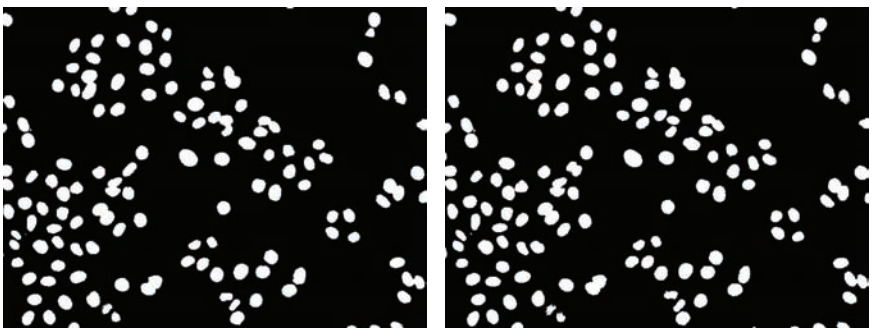


Figure 41. Binary images of two frames in one cell-cycle screening (frame time: 15 minutes)



(1) (2)

are shown in Figures 42(1-3), and the image in Figure 42(4) is not a touching cell image because of its small size.

The smooth following and linearization results of images in Figure 42 can be shown in Figures 43(1-4) based on algorithms of smooth following and linearization, which are described in the sections Smooth Following and Linearization.

Furthermore, the structural points of the images in Figures 43(1-4) can be extracted based on an algorithm of extracting structural points, as shown in Figure 44. We can see that there are some concave structural points on the contours of the images in Figures 44(1-4). Based on the definition of structural points, one concave point means a concave change in the direction of one

Figure 42. Binarization images of three touching cell images and one telophase cell image taken from Figure 41

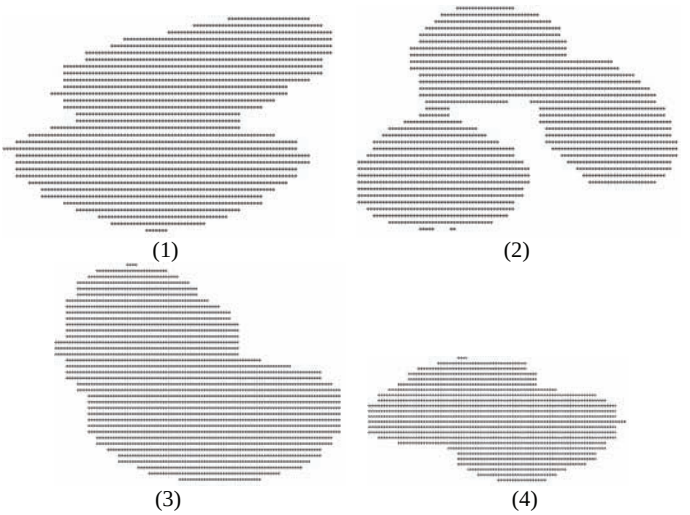
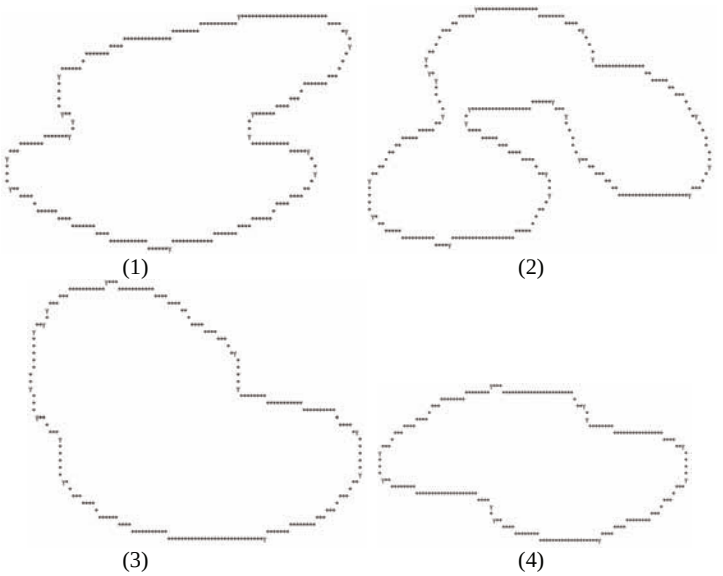


Figure 43. The results of smooth following and linearization for the images in Figures 42(1-4)



chain code on the contour. Generally, there is at least one concave change on the outer contour of an image if two or more ellipses are touching each other. Therefore, we can detect whether there is any concave on the image contour to determine if there is a touching cell image. Also, the separation points of a touching cell image can be found based on the concave points of a touching cell image.

Let a series of concave structural points on the outer contour of touching cell images be:

$$S_{cc} = s_{cc}(o), s_{cc}(1)...s_{cc}(i),...s_{cc}(n-1), s_{cc}(n) \quad (19)$$

where s_{cc} (i) is the structural point number of the ith concave structural point on the contour, and there are n concave structural points on the contour. It is clear that s_{cc} (i) < s_{cc} (i + 1). In fact, one concave change on the contour may consist of several closest concave structural points. For example, if there exists s_{cc} (i + 1) - s_{cc} (i) =1 and s_{cc} (i + 2) - s_{cc} (i + 1) =1, then that means one concave change consists of three concave structural points, s_{cc} (i), s_{cc} (i + 1) and s_{cc} (i + 2). In this case, these three concave structural points should be merged into one group of concave structural points. After this merging processing for S_{cc}, a series of groups of concave structural points represented as S_{cg}:

$$S_{cg} = s_{cg}(o), s_{cg}(1)...s_{cg}(i),...s_{cg}(k-1), s_{cg}(k) \quad (20)$$

can be found, where k is the number of groups and $k \leq n$.

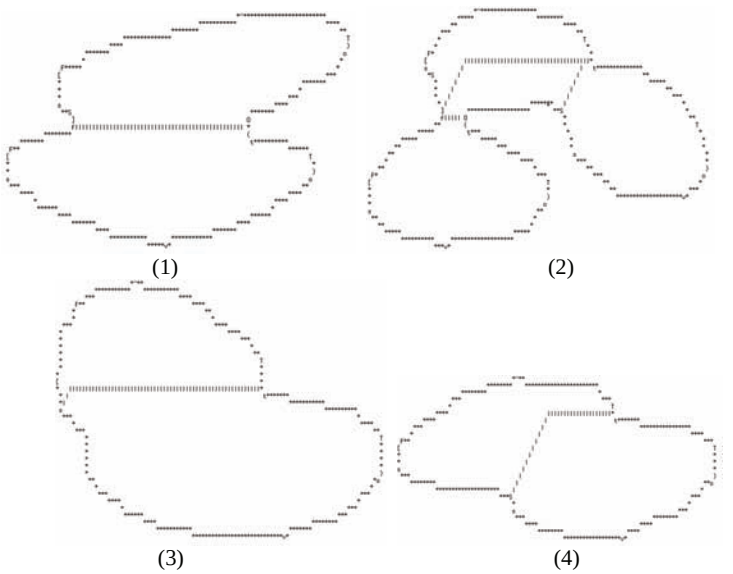
For example, nine concave structural points in Figure 44(2) are merged into three groups of concave structural points. The morphological patterns of touching cell images can be determined based on the number of groups of concave structural points.

- If k=1 or k=2, two cells are touched.
- If k=3, three cells are touched.
- If k=4, four cells are touched.

Separation Points of Touching Cell Images

The method of searching separation points can be described as follows.

Figure 44. The extracting structural points of the images in Figures 43(1-4)



Case 1 (*k*=1)

If k=1, there is one group of concave points, $s_{cg}(0)$. Suppose $s_{cg}(0)$ contains p concave points, $s_{cg}(0),\ldots s_{cg}(p-1)$ p<4. For each concave point, find its matching convex structural points, which are defined as its corresponding convex structural points in the approximate reverse direction of chain code. For example, if $s_{cg}(0)$ is concave structural point "Λ", then its match convex structural points are "s," "v," and "o." Let q be the number of the corresponding match convex structural points for all $s_{cg}(0),\ldots s_{cg}(p-1)$, and they are represented as $s_{cv}(0),\ldots s_{cv}(q-1)$. We can determine separation points that make minimum distance between one pair of one concave structural points in $s_{cg}(0),\ldots s_{cg}(p-1)$ and one convex structural point in $s_{cv}(0),\ldots s_{cv}(q-1)$. That is:

$$\left\{s_{cg0}(m), s_{cc}(n)\right\} = \min\left\{\left\| s_{cg0}(i), s_{cc}(j)\right| i < p, j < q\right\}, \quad (21)$$

where $s_{cg}(m)$ and $s_{cc}(n)$ are the point numbers of the selected separation points, respectively.

Case 2 (*k*=2)

If k=2, there are two groups of concave points, $s_{cg}(0)$ and $s_{cg}(1)$. Suppose the number of concave structural points in $s_{cg}(0)$ is p_0, and in $s_{cg}(1)$ is p_1, respectively. In this case, we can determine separation points that make minimum distance between one pair of one concave structural point in $s_{cg}(0),\ldots s_{cg}(p_0)$ and another one in $s_{cg}(1),\ldots s_{cg}(p_1)$. That is:

$$\left\{s_{cg0}(m), s_{cg1}(n)\right\} = \min\left\{\left\| s_{cg0}(i), s_{cg1}(j)\right| i < p_0, j < p_1\right\}, \quad (22)$$

where $s_{cg0}(m)$ and $s_{cg1}(n)$ are the point numbers of the selected separation points, respectively.

Case 3 (*k*>2)

If k>2, there are more than two groups of concave points, $s_{cg}(0),\ldots s_{cg}(l)$ l>2. In this case, we can determine each pair of separation points that make minimum distance between each pair of one concave structural point in $s_{cgx}(0),\ldots s_{cgx}(p_x)$ and another one in $s_{cgy}(0),\ldots s_{cgy}(p_y)$ where they are neighboring groups of concave structural points. That is:

$$\left\{s_{cgx}(m), s_{cgy}(n)\right\} = \min\left\{\left\| s_{cgx}(i), s_{cgy}(j)\right| i < p_x, j < p_y\right\}, \quad (23)$$

where $s_{cgx}(p_x)$ and $s_{cgy}(p_y)$ are the point number of the selected separation points.

For example, if k=3, there are three pairs of groups of concave structural points, $s_{cg}(0)$ and $s_{cg}(1)$, $s_{cg}(1)$ and $s_{cg}(2)$, and $s_{cg}(2)$ and $s_{cg}(0)$, respectively.

Based on this algorithm, we can find all separation points of images in Figures 44(1-4). We can find related separation lines (see Figures 44(1-4)) and the coordinate data of related arcs, which are shown in Figures 45(2,3), 46(2,3), and 47(2,3,4), based on these separation points and series of points of the contour. These contours of touching cell images are shown in Figures 45(1), 46(1), and 47(1).

Reconstruction of Touching Cell Images

We have found the coordinate data of all related arcs that are separated based on the previous algorithm. As all cell shapes are approximately as an ellipse, touching cell images can be reconstructed based on these separated arcs. The reconstruction method is direct least square fitting of ellipses (Fitzgibbon, Pilu & Fisher, 1999). The reconstructed cell images are shown in Figures 45(4,5), 46(5,6,7), and 47(4,5) based on the coordinate data of separated arcs, respectively.

Dynamic Analysis and Recognition of Different Cell Phases

In order to detect the morphological structures of cells, it is necessary to recognize the cell shape of different cell phases. We can see that there

Figure 45. The contour, separated arcs, and reconstructed ellipses of sample touching cell image 1 in Figure 44(1)

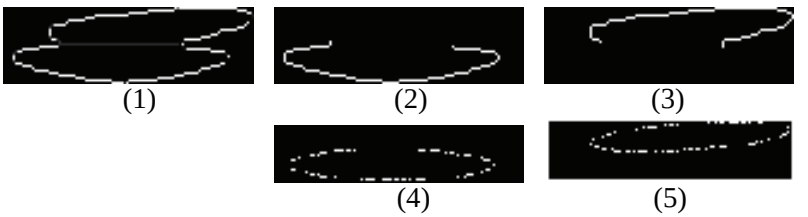


Figure 46. The contour, separated arcs, and reconstructed ellipses of sample touching cell image 2 in Figure 44(2)

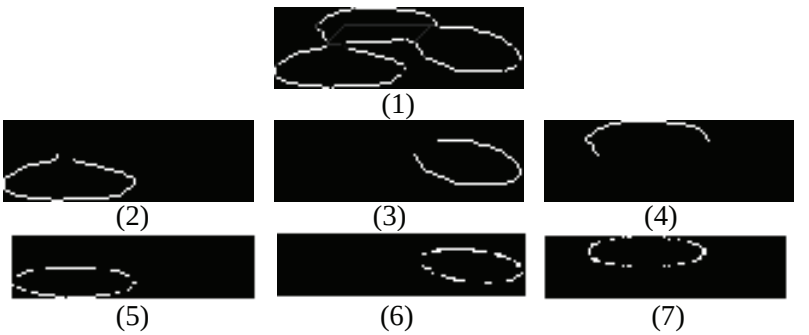
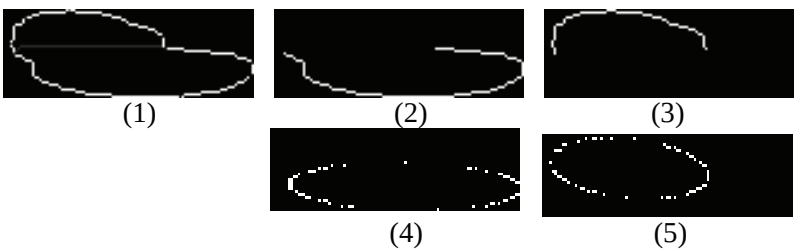


Figure 47. The contour, separated arcs, and reconstructed ellipses of sample touching cell image 3 in Figure 44(3)



are mainly two classes of cell shapes, ellipse and barbell, for our cell screening. The ellipse shapes can be three types—skew, horizontal, and vertical—based on the relation between their major and minor axes. It is clear here that "skew," "horizontal," and "vertical" are fuzzy. Some recognition models of cell shapes can be described as the following, based on their morphological structures.

If a cell shape is an ellipse, there are no concave structural points on the outer contour of the cell contour. Furthermore, four models of ellipse can be described as follows: Calculate the number of groups of codes (codes 0, 4, 5, and 1), (codes 5, 1, 2, and 6), (codes 6, 2, 7 and 3), (codes 7, 3, 0 and 4), on the outer contour of the cell image contour, respectively.

Morphological Model 1: Ellipse shapes $e_{(5,1,2,6)}$ and $e_{(6,2,1,5)}$

For these shapes, the number of groups of codes, (codes 5, 1, 2, and 6), is largest. Let $c_{5,6,1,2}$ be the

total number of codes 5, 6, 1, and 2, c_t be the total number of all codes, $c_{5,1}$ be the total number of codes 5 and 1, and $c_{6,2}$ be the total number of codes 6 and 2 on the outer contour of the cell image, respectively. If (1) the previous condition is met; (2) its outer contour mainly consists of chain codes 5, 6, 1, and 2 ($c_{5,6,1,2} \geq \frac{1}{2} c_t$); (3) the number of chain codes 5 and 1 is more than that of chain codes 6 and 2 ($c_{5,1} \geq c_{6,2}$), then the cell image shape is recognized as the shape $e_{(5,1,2,6)}$; otherwise, ($c_{5,1} < c_{6,2}$) the cell image is recognized as the shape $e_{(6,2,1,5)}$.

In this model, the cell image shape, $e_{(5,1,2,6)}$, is a skew ellipse in the direction of code 5 or 1, and the cell image shape, $e_{(6,2,1,5)}$, is mainly a vertical ellipse.

Morphological Model 2: Ellipse shapes $e_{(7,3,0,4)}$ and $e_{(0,4,3,7)}$

For these shapes, the number of groups of codes (codes 7, 3, 0, and 4) is largest. Let $c_{7,0,3,4}$ be the total number of codes 7, 0, 3, and 4, c_t be the total number of all codes, $c_{0,4}$ be the total number of codes 0 and 4, and $c_{7,3}$ be the total number of codes 7 and 3 on the outer contour of the cell image, respectively. If (1) the previous condition is met; (2) its outer contour mainly consists of chain codes 7, 0, 3, and 4 ($c_{7,3,0,4} \geq \frac{1}{2} c_t$); (3) the number of chain codes 7 and 3 is more than that of chain codes 0 and 4 ($c_{7,3} \geq c_{0,4}$), then the cell image is recognized as the ellipse shape $e_{(7,3,0,4)}$; otherwise, ($c_{7,3} < c_{0,4}$) the cell image is recognized as the shape $e_{(0,4,3,7)}$.

In this model, the cell image shape $e_{(7,3,0,4)}$ is considered as a skew ellipse in the direction of code 7 or 3, and the cell image shape, $e_{(0,4,3,7)}$ a horizontal ellipse.

Morphological Model 3: Ellipse shapes $e_{(6,2,7,3)}$ and $e_{(7,3,2,6)}$

For these shapes, the number of groups of codes (codes 6, 2, 7, and 3) is largest. Let $c_{6,7,2,3}$ be the total number of codes 6, 7, 2, and 3, c_t be the total number of all codes, $c_{6,2}$ be the total number of codes 6 and 2, and $c_{7,3}$ be the total number of codes 7 and 3 on the outer contour of the cell image, respectively. If (1) the previous condition is met; (2) its outer contour mainly consists of chain codes 6, 7, 2, and 3 ($c_{6,7,2,3} \geq \frac{1}{2} c_t$); (3) the number of chain codes 6 and 2 is more than that of chain codes 7 and 3 ($c_{6,2} \geq c_{7,3}$), then the adjunctive segment is recognized as the ellipse shape $e_{(6,2,7,3)}$; otherwise, ($c_{6,2} < c_{7,3}$) the cell image is recognized as the shape $e_{(7,3,2,6)}$.

Morphological Model 4: Ellipse shapes $e_{(5,1,0,4)}$ and $e_{(0,4,5,1)}$

For these shapes, the number of groups of codes, (codes 5, 1, 0, and 4) is largest. Let $c_{4,5,0,1}$ be the total number of codes 4, 5, 0, and 1, c_t be the total number of all codes, $c_{5,1}$ be the total number of codes 5 and 1, and $c_{4,0}$ be the total number of codes 4 and 0 on the outer contour of the cell image contour, respectively. If (1) the previous conditions are met; (2) its outer contour mainly consists of chain codes 4, 5, 0, and 1 ($c_{4,5,0,1} \geq \frac{1}{2} c_t$); (3) the number of chain codes 5 and 1 is more than that of chain codes 4 and 0 ($c_{5,1} \geq c_{4,0}$), then the cell image shape is recognized as ellipse shape $e_{(5,1,0,4)}$; otherwise, ($c_{5,1} < c_{4,0}$) the cell image shape is recognized as the shape $e_{(0,4,5,1)}$.

Based on the previous recognition models, the cell image in Figure 40(1) is recognized as shape $e_{(6,2,7,3)}$, the cell images in Figures 40(2-3) are recognized as shape $e_{(6,2,1,5)}$, and the cell images in Figures 40(5-6) are recognized as ellipse shape $e_{(0,4,3,7)}$.

Morphological Model 5: Barbell Shapes

If (1) there are two concave structural changes; (2) there is one pair of corresponding concave structural points, "Λ" and "$" (horizontal), "]" and "(" (vertical), "f" and "O" (skew), or "t" and "S" (skew), then cell image contour can be

recognized as the barbell shape. There are four types of barbell shapes that can be defined based on which pair of concave structural points is found. The cell image contour in Figure 40(7) can be recognized as a barbell shape, and its pair of corresponding concave structural points are structural points "]" and "(".

Cell Phases

The different phases of a cell are determined based on the shape and size of the cell image in the current frame and those in its neighboring frames of the cell screening. As the shape ellipse, its shape change is mainly based on the rate between major and minor axes, and area size of cell image.

Basically, if there are concave changes in the cell image and its size is large, then the cell image is touching cell image. First, if the cell image's size is not too large, then different cell phases can be traced and recognized as follows:

1. If (a) the cell shape is an ellipse and (b) the area size of the cell image is large enough (threshold can be found from statistical analysis of cell screening), then the cell is a normal changing one (metaphase). In this case, there are two phases: (1) if its changing trend is that the rate between major and minor axes of the cell image is decreased with time (compare these features between two neighboring frames), then the phase is changing from prophase to metaphase; (2) otherwise, from metaphase to telophase.
2. If the rate between major and minor axes of the cell image is large enough (threshold can be found from statistical analysis of cell screening), there are two phases: (1) if there are two new small cells at tracing location, then the cell has split; (2) otherwise, the cell will be split soon (telophase).
3. If the cell shapes are a barbell shape and the area size of the cell image is not large, then the cell will be split (telophase).
4. If (a) the cell shape is an ellipse and (b) the area size of the cell image is small enough, then the cell is a newly changing one, and its changing trend is that the area size of the cell image is increased with time (compare these features between two neighboring frames) (prophase).
5. If the cell shapes and area are not changed for a long time, it should be a dead cell.

Based on the previous analysis, the cells in Figures 40(1-5) are at metaphase, the cells in Figures 40(6-7) are at telophase soon, and the cells in Figure 40(8) are at prophase. In this way, cell screening can be analyzed and recognized dynamically and automatically.

Summary

An efficient and new method has been developed for analysis and recognition of cell-cycle screening based on morphological structures. The morphological structures of cells are described based on structural points. The different cell phases are determined in terms of cell shape recognition, cell size between neighboring frames of cell screening. The touching cell images are recognized, segmentation points are detected, and separated cell images are reconstructed. Our algorithms have been used to process the cell database, which consists of 480 frames. Average division time is 1,830 minutes (tracing the cell from prophase to telophase). All phases of cells of the database can be traced and recognized. The dynamic analysis and recognition of cell-cycle screening are approached based on morphological structure models and prior knowledge of cell images. The best useful contribution is that some series of structural features of linearized lines of contours are found, and morphological models of cell contour shapes are described based on our algorithm, but not other methods. Our method is efficient and new because morphological structure

models of cell images are constructed, and these models simulate artificial intelligence.

CONCLUSION

In this chapter, we have described morphological structure of images: smooth following, linearization-based difference codes, patterns, and extraction of structural points. Finally, we have discussed how to use morphological to analyze and recognize practical object images based on morphological structure by two applications. One is reconstruction and recognition of handwritten broken digits, and another one is dynamic analysis and recognition of cell-cycle screening based on morphological structures. Smooth following techniques are developed to smooth contours of images. The spurious pixels (points) of contours in images can be removed based on the structural patterns of difference codes and of spurious point groups. The algorithm can let the abstract value of difference codes between neighboring points be less than 2. This makes the linearization of contours possible.

The skeletons of images can be smoothed based on the structural patterns of skeleton smoothing, and some spurious points in the skeleton are removed. Based on the new algorithm, the abstract value of difference between neighboring "end" and "junction" points is less than 2. Smoothed skeleton can make the extraction of skeleton features possible.

A new and efficient algorithm about linearization-smoothed contours is discussed. A series of linearized lines in a contour is formed based on element chain codes, which are found in terms of difference chain codes. A series of description features of contours is formed, but not by other methods.

The morphological structural points are proposed. The morphological structural points are defined to describe morphological structures between neighboring linearized lines along contours. Extraction of structural points is based on structural patterns, which are determined by element chain codes. Two applications are used to discuss how to analyze and recognize object images. One application is reconstructing and recognizing broken handwritten digits based on their morphological structure and skeleton. The preselected broken points are made based on minimum distance between skeletons of neighboring segments. Correction rules of some preselected broken points are constructed based on the shape recognition and the geometrical location of the neighboring segments, the relation between neighboring segments, the shape structure of a digit, the geometrical property of segments, and the extending results of segments. The algorithm is more efficient because it not only uses some features such as geometry and stroke extension information, but also uses the morphological structure and skeleton information of broken digits. Therefore, based on his method, both small and large true gaps of broken digits are linked, but the normal gaps of broken digits are not.

Also, two methods of recognizing handwritten digits are discussed. One method combines the reconstruction of handwritten digits and ONNC. Another recognition method of handwritten digits is based on all possible morphological structures of handwritten digits. Experiments with large numbers of testing data show satisfactory results for these algorithms.

Another application is dynamic analysis and recognition of cell-cycle screening based on morphological structures. The algorithm of extracting structural features (structural points) is described based on smooth followed contour, linearized line, and difference chain codes. The morphological structures of cell contour shapes are constructed based on the structure points, difference chain code of linearized line, and geometry features. The dynamic analysis and recognition of cell-cycle screening are approached based on morphological structure models and prior knowledge of cell images. Comparing our algorithm with other

methods, the best useful contribution is that some series of structural features of linearized lines of contours are found, and morphological models of cell contour shapes are described based on our algorithm, but other methods do not. Our method is efficient and new because morphological structure models of cell images are constructed, and these models simulate artificial intelligence.

REFERENCES

Chen, X., Zhou, X., & Wong, S.T.C. (in press). Automated segmentation, classification, and tracking cancer cell nuclei in time-lapse microscopy. *IEEE Trans on Biomedical Engineering.*

Dunkle, R. (2003). Role of image informatics in accelerating drug discovery and development. *Drug Discovery World, 5,* 75–82.

Feng, Y. (2002). Practicing cell morphology based screen. *European Pharmaceutical Review, 7,* 7–11.

Fitzgibbon, A., Pilu, M., & Fisher, R.B. (1999). Direct least square fitting of ellipse. *Term Analysis and Machine Intelligence, 21,* 476–480.

Fox, S. (2003). Accommodating cells in HTS. *Drug Discovery World, 5,* 21–30.

Freeman, H. (1961). On the encoding of arbitrary geometric configuration. *Trans Electronics Computers, EC-10,* 264–268.

Fu, A.M.N., Yan, H., & Huang, K. (1997). A curvature angle bend function based method to characterize contour shapes. *Patt Recog, 30*(10), 1661–1671.

Garris, M.D., & Wilkinson, R.A. (1992). *NIST special database 3 handwritten segmented characters.* National Institute of Standards and Technology.

Gonzalez, R.C., & Woods, R.E. (1993). *Digital image processing.* 2nd ed. Adisson-Wesley.

Hiraoka, Y., & Haraguchi, T. (1996). Fluoresence imaging of mammalian living cells. *Chromosome Res, 4,* 173–176.

Hu, J., & Yan, H. (1998). Structural primitive extraction and coding for handwritten numeral recognition. *Patt Recog, 31*(5), 493–509.

Lee, S.W. (1996). Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural networks. *IEEE Trans Pattern Anal Mach Intell, 18*(6), 648–652.

Malaviya, A., & Klette, R. (1996). A fuzzy syntactic method for on-line handwriting recognition. In P. Pernet, P. Wang, & A. Rosenfeld, Eds., *Advance in structural and syntactical pattern recognition*. Springer.

Moktarian, F., & Mackworth, A.K. (1992). A theory of multiscale curvature-based shape representation for planer curvature angles. *IEEE Trans Pattern Analysis Mach Intell, 14(8),* 789–805.

Ostu, N. (1978). A thresholding selection method from graylevel histogram. *IEEE Trans Systems Man Cybernet, SMC8,* 62–66.

Pavlidis, T. (1982). *Algorithms for graphics and image processing.* Computer Science Press.

Pham, D.T.D., Tran, T., Zhou, X., & Wong, S.T.C. (2005). An automated procedure for cellphase imaging identification. *Proceedings of the AI-2005 Workshop on Learning Algorithms for Pattern Recognition,* 52–59.

Pham, T.D., Tran, D.T., Zhou, X., & Wong, S.T.C. (2006). Classification of cell phases in time-lapse images by vector quantization and Markov models. In E.V. Greer, Ed., *Neural stem cell research.* New York: Nova Science.

Rocha, J., & Pravlidis, T. (1994). A shape analysis model with application to a character recognition system. *IEEE Trans Pattern Anal Mach Intell, 16*(4), 393–404.

Rosenfeld, A. (1973). Arcs and curves in digital pictures, *JACM, 20*, 81–87.

Shi, M., Fujisawa, Y., Wakabayashi, T., & Kimura, F. (2002). Handwritten numeral recognition using gradient and curvature of gray scale image. *Patt Recog, 35*, 2051–2059.

Sonka, M., Hlavac, V., & Boyle, R. (1993). *Image processing, analysis and machine vision*. Cambridge: Chapman & Hall Computing.

Suters, M., & Yan, H. (1994). Connected handwritten digit separation using external boundary curvature. *Journal of Electronic Imaging, 3*(3), 251–256.

Wang, C.X., Qi, Y., Yu, D., & Xu, S. (1990). A fast algorithm for boundary tracing of binary image with neighborhood coding. *Proceedings of the International Conference on Signal Processing*, 1083–1085.

Whichello, A., & Yan, H. (1996). Lining broken character borders with variable sized masks to improve recognition. *Patt Recog, 29*(8), 1429–1435.

Yan, H. (1993). Prototype optimization of a nearest neighbor classifier using a multi-layer neural network. *Patt Recog, 26*, 317–324.

Yan, H. (1994). Handwritten digit recognition using optimized prototypes. *Patt Lett, 15*, 207–211.

Yang, Y., & Yan, H. (2000). An adaptive logical method for binarization of degraded document images. *Pattern Recognition, 33*(5), 787–807.

Yarrow, J.C., et al. (2003). Phenotypic screening of small molecule libraries by high throughput cell imaging. *Comb Chem High Throughput Screen, 6*, 279–286.

Yu, D., Hu, J., & Yan, H. (1999). Separation of single- and double-touching handwritten numeral strings based on structural analysis. *Optical Engineering, 38*(3), 514–522.

Yu, D., & Yan, H. (1997). An efficient algorithm for smoothing, linearization and detection of structure feature points of binary image contours, *Patt Recog, 30*(1), 57–69.

Yu, D., & Yan, H. (1998). Separation of single-touching handwritten numeral strings based on structural features. *Patt Recog, 31*(12), 1835–1847.

Zhang, T.Y., & Suen, C.Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of ACM, 27*, 236–239.

Chapter VIII
Robust Face Recognition Technique for a Real-Time Embedded Face Recognition System

Ting Shan
National ICT Australia, and The University of Queensland, Australia

Abbas Bigdeli
National ICT Australia, Australia

Brian C. Lovell
National ICT Australia, and The University of Queensland, Australia

Shaokang Chen
National ICT Australia, and The University of Queensland, Australia

ABSTRACT

In this chapter, we propose a pose variability compensation technique, which synthesizes realistic frontal face images from nonfrontal views. It is based on modeling the face via active appearance models and estimating the pose through a correlation model. The proposed technique is coupled with adaptive principal component analysis (APCA), which was previously shown to perform well in the presence of both lighting and expression variations. The proposed recognition techniques, though advanced, are not computationally intensive. So they are quite well suited to the embedded system environment. Indeed, the authors have implemented an early prototype of a face recognition module on a mobile camera phone so the camera can be used to identify the person holding the phone.

INTRODUCTION

Although face recognition has been researched for many years, promising laboratory systems developed with off-line face databases have not fared well when deployed in the real world. This is largely because of the deleterious effect of image acquisition problems such as lighting angle, facial expression, and head pose. Accuracy may drop to 10% or even lower under uncontrolled image acquisition conditions. Such conditions are often encountered in automatic identity capture for video surveillance and for identification of faces from a mobile camera phone.

Indeed, the mobile phone is a very interesting target platform for advanced pattern recognition. Many modern phones can reliably recognize speech in noisy environments. Some recognize handwriting, including Chinese characters. Newer phones are equipped with fingerprint reading sensors and software. At first glance, it seems strange that such advanced pattern recognition algorithms representing decades of research are found on garden-variety mobile handsets. Yet the small size of the mobile phone encourages the development of pattern recognition interfaces rather than bulky keyboards. The high cost of licensing these algorithms is defrayed by the sheer volume of mobile phones being sold every year. With this in mind, the authors have implemented an early prototype of a face recognition module on a mobile camera phone so the camera can be used to identify the person holding the phone and unlock the keyboard. Robust face recognition is a key requirement of such a system.

Our recent research has led to recognition systems that are much less sensitive to uncontrolled image capture. These methods are described in this chapter. Now the challenge is to incorporate these new algorithms into embedded platforms to help realize our dream of ubiquitous, low-cost face recognition.

BACKGROUND OF FACE RECOGNITION

Face recognition under uncontrolled image acquisition conditions is a challenging goal, not only because of the gross similarity of all faces, but also because of the vast differences between face images of the same person due to variations in lighting conditions, facial expression, and pose. Figure 1 shows example images of the same person with the lighting source positioned in front, above, bottom, right, and left of the person,

Figure 1. Example images of the same person with the lighting source from front, above, bottom, right, and left of the person, respectively, from Asian Face Database (© copyright 2007, I.M. Lab; used with permission)

Figure 2. Example images of the same person with neutral, smiling, surprising, and sad expressions, respectively, from Yale Face Database (Yale, 1997)

Figure 3. Example images of the same person with different presentation angles, from Feret Database (NIST, 2001)

respectively. Figure 2 shows example images of a person with neutral, smiling, surprising, and sad expressions, respectively. Finally Figure 3 shows example images of the same person with different presentation angles.

An ideal face recognition system should recognize new images of a known face from a novel image and be maximally insensitive to nuisance variations in image acquisition. However, the fact that differences between images of the same face due to these nuisance variations are normally greater than those between different faces (Adini, Moses & Ullman, 1997) makes this task exceedingly difficult. Variations in illumination, facial expression, and head pose significantly degrade the performance of a face recognition system. Among these three important variations, pose variation is the hardest one to model (Phillips, Grother, Micheals, Blackburn, Tabassi & Bone, 2003). Therefore, most of the existing face recognition systems only work well in well controlled environments (Chellappa, Wilson & Sirohey, 1995) with frontal images taken under constrained or laboratory conditions. However, this requirement is too strict to be met in many situations or when only a few gallery images are available, such as in recognizing people from images captured on mobile cameras.

Illumination-Insensitive Face Recognition

Many approaches have been proposed for illumination-invariant recognition (Yilmaz & Gokmen, 2000; Yongsheng & Leung, 2002) and expression-invariant recognition (Beymer & Poggio, 1995; Black, Fleet & Yacoob, 2000). Many of these methods suffer from the need to have large numbers of example images for training, or they are computationally too expensive to be used in mobile camera phone platforms. In 2004, we developed adaptive principal component analysis (APCA) and rotated APCA (Chen & Lovell, 2004; Lovell & Chen, 2005), which inherited merits from both PCA and FLD (Fisher linear discriminant) (Belhumeur, Hespanha & Kriegman, 1997) by warping the face subspace according to the within-class and between-class covariance to compensate for illumination and facial expression variations. We sometimes refer to the highly representative features used in these techniques as "eigenfisher-faces" to reflect the hybrid nature of their derivation.

We first apply principal component analysis (PCA) (Turk & Pentland, 1991a, 1991b) to project each face image into a face subspace with reduced dimension to form an m-dimension feature vector $s_{j,k}$ with k = 1, 2, …, K_j denoting the k^{th} sample of the class S_j. After constructing the face subspace for image representation, we warp this space to enhance class separability. According to Bayes rule, the conditional density function is:

$$p(s \mid S_j) = \frac{\exp[-\frac{1}{2}(s-u_j)^T \Sigma_j^{-1}(s-u_j)]}{(2\pi)^{\frac{m}{2}} \mid \Sigma_j \mid^{\frac{1}{2}}}$$

where u_j is the mean of class S_j and cov_j is the covariance matrix of S_j under the assumption of a Gaussian distributed random variables.

In order to compensate for the influence of between-class covariance on the estimation of the pdf, we use a whitening power parameter, *p*, to whiten the distribution:

$$\text{cov} = diag\{\lambda_1^{-2p}, \lambda_2^{-2p}, ... \lambda_m^{-2p}\}$$

where $\lambda_i (i = [1...m])$ are the eigenvalues extracted by PCA. Each eigenface u_i is whitened according to the corresponding eigenvalue λ_i using the power *p*. We then apply filtering to weigh each eigenface according to the corresponding identity-to-variation (ITV) ratio, which measures the correlation of a change in person vs. a change in variation for each of the eigenface. For an *M* class problem, assume that for each of the *M* classes (persons) we have examples under *K* standardized different lighting conditions. We denote the i^{th} element of the face vector of the k^{th} lighting sample for class (person) S_j by $s_{i,j,k}$. What follows is displayed in Box 1.

$\varpi_{i,k}$ represents the i^{th} element of the mean face vector for lighting condition k for all persons, and $\mu_{i,j}$ represents the i^{th} element of the mean face vector for person *j* under all lighting conditions.

Box 1.

$$ITV_i = \frac{BetweenClassCo\,\text{var}\,iance}{WithinClassCo\,\text{var}\,iance} = \frac{\frac{1}{M}\sum_{j=1}^{M}\frac{1}{K}\sum_{k=1}^{K} \mid s_{i,j,k} - \varpi_{i,k} \mid}{\frac{1}{M}\sum_{j=1}^{M}\frac{1}{K}\sum_{k=1}^{K} \mid s_{i,j,k} - \mu_{i,j} \mid}$$

$$\varpi_{i,k} = \frac{1}{M}\sum_{j=1}^{M} s_{i,j,k}$$

$$\mu_{i,j} = \frac{1}{K}\sum_{k=1}^{K} s_{i,j,k}$$

We then define the filtering matrix γ by:

$$\gamma = diag\{ITV_1^q, ITV_2^q, \ldots ITV_m^q\}$$

where q is an exponential scaling factor determined empirically.

We use the following cost function, which is a combination of error rate and the ratio of within-class distance to between-class distance, and optimize it empirically using an objective function defined by Box 2, where $d_{jj,k0}$ is the distance between the sample $s_{j,k}$ and $s_{j,0}$, which is the standard image reference (typically the normally illuminated neural image) for class S_j. Correspondingly, $d_{jm,k0}$ is the between-class distance between sample $s_{j,k}$ and the reference image $s_{m,0}$ for class S_m.

Expression Invariant Face Recognition

We rotate the feature space according to within-class covariance to enhance the representativeness of the features and to improve estimation of the conditional pdf. The rotation matrix R is a set of eigenvectors obtained by applying singular value decomposition to the within-class covariance matrix. Every face vector s is transformed into the new space by R:

$$r = R^T \text{ s}$$

We test RAPCA, APCA, and PCA on the Asian Face Image Database (I.M. Lab), which consists of 535 facial images under five different standardized illuminations and 428 images with four different facial expressions corresponding to 107 subjects. Each image is 171*171 pixels with 256 grey levels per pixel. We perform threefold cross-validation on the database in an attempt to obtain reliable estimates of accuracy on unknown faces. Figure 4 shows that our methods offer significant recognition performance gains over standard PCA when changes in lighting, expression, and both lighting and expression are present.

APCA and RAPCA perform well against both lighting variation and expression change, but they only perform well with frontal face images as registered in the database. In the mobile camera phone scenario, many face images captured by the cameras are not perfectly frontal, so the need for a pose-invariant face recognizer becomes crucial.

Background of Pose-Insensitive Face Recognition

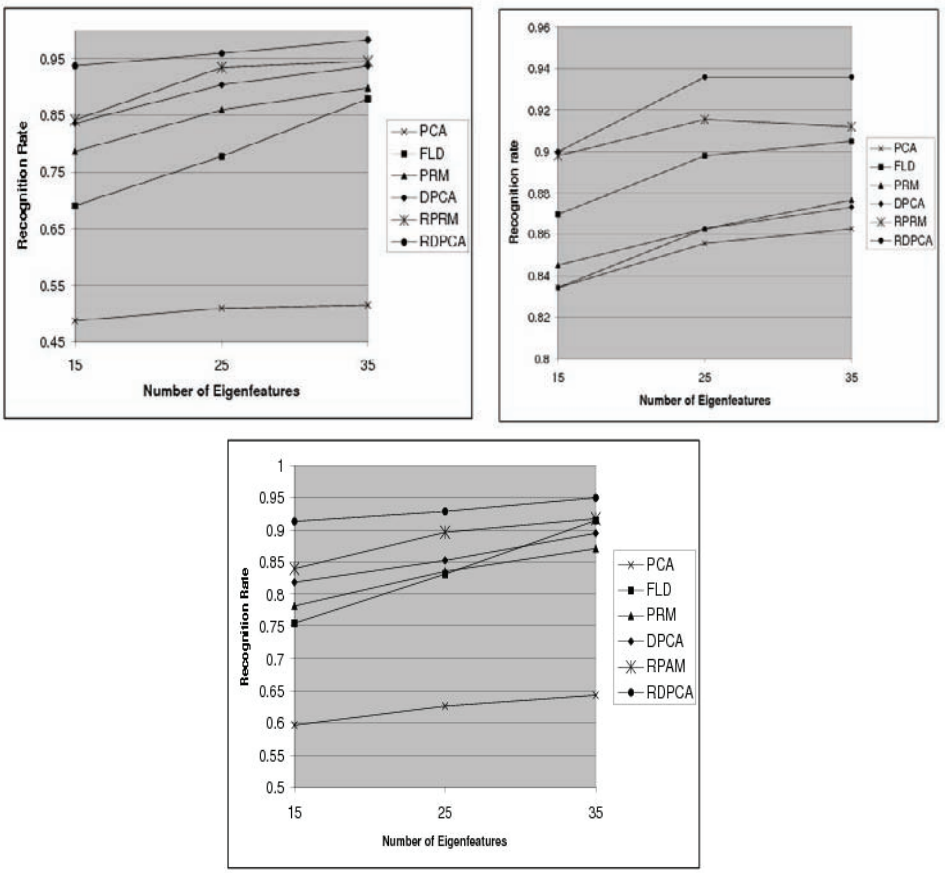
The performance of face recognition systems drops significantly when large pose variations are present (Phillips et al., 2003). Many approaches have been proposed to compensate for pose change.

Wiskott, Fellous, Kuiger, and von der Malsburg (1997) extend the DLA- (dynamic link architecture-) based face recognizer (Lades et al., 1993) to deal with larger pose variations. The face image is represented by a labeled graph called the face bunch graph (FBG), which consists of N nodes connected by E edges. The nodes are located at facial landmarks $\overrightarrow{x_n}, n = 1, \ldots, N$, which are called fiducial points. By using such a graph, the correspondences between face images across different viewpoints can be found. Matching an FBG on a new image is done by maximizing a graph

Box 2.

$$OPT = \sum_{j=1}^{M} \sum_{k=1}^{K} \sum_{m} \frac{d_{jj,k_0}}{d_{jm,k_0}}, \forall m \in d_{jm,k_0} < d_{jj,k_0}, m \in [1 \cdots m]$$

Figure 4. Recognition rates for APCA and RAPCA with variations in lighting (top left), expression (top right), and both lighting and expression (bottom) (Chen & Lovell, 2004; Lovell & Chen 2005)



similarity between an image graph and the FBG at that pose. DLA approaches are invariant to pose change with rotations in depth less than 30 degrees, but the computation is too expensive[1] to be used in a real-time face recognition system.

In 1994, Pentland, Moghaddam, and Starner (1994) proposed two methods for use in face recognition using variable pose. Given *N* individuals under *M* different views, face recognition and pose estimation is carried out in a universal eigenspace computed from the combination of all the NM images. In this way, a "parametric eigenspace" will encode both identity and view conditions (Murase & Nayar, 1993). The other method is to build a "view-based" set of M separate eigenspaces, each capturing the variation of the N individuals in a common view (Darrell & Pentland, 1993). In the view-based method, each view space's eigenvectors are used to compute the "distance-from-face-space" (DFFS) (Turk & Pentland, 1991a, 1991b); once the proper view space is determined, the input face image is encoded using the eigenvectors of that view space and then recognized.

In 2003, Chai, Shan, and Gao (Xiujuan, Shiguang & Wen, 2003) presented a framework for pose-invariant face recognition using a pose alignment method, which is based on a statistical transformation. Their algorithm partitions the face into three rectangular regions, and the

affine transformation parameters associated with various poses are learned from the one-to-one rectangle mapping relations. The affine transformation parameters associated with different poses can be used to align the input nonfrontal face image to frontal view face image, and the virtual frontal view can be generated through the polynomial warping. Experiments on the FERET dataset (NIST, 2001) show that their method can increase recognition rate by an average of 17.75% compared to face recognition without pose alignment. But their recognition rate was still quite low, only reaching an average of 58% under pose rotations of 30 degrees. Moreover, they did not develop an automatic way to mark the key points on facial structures.

In 2000, Cootes, Walker, and Taylor (2000) proposed "view-based active appearance models," which was based on the idea that a small number of 2D statistical models are sufficient to capture the shape and appearance of a face from any viewpoint. They demonstrated that to deal with pose angle change from left to right profile, only three distinct models were needed; each model is trained on the labeled images of a variety of people. They learn the relationship between model parameters and head orientation based on the assumption that the model parameters trace out an approximately elliptical path, which can be used to estimate both the orientation of any head pose and synthesize a new face at any orientation. They applied this method to face tracking but did not do any face recognition experiments.

Sanderson, Bengio, and Gao (2006) address the pose mismatch problem by extending each frontal face model with artificially synthesized models for nonfrontal views. The synthesized methods are based on several implementations of maximum likelihood linear regression (MLLR) and standard multivariate linear regression (LinReg). In the MLLR-based approaches, they used prior information to construct generic statistical face models for different views. A "generic" GMM was used to represent a population of faces. Each nonfrontal generic model is constructed by learning and applying an MLLR-based transformation to the frontal generic model. The LinReg approach is similar to the MLLR-based approach. The main difference is that it learns a common relation between two sets of feature vectors instead of learning the transformation between generic models. They evaluate these two approaches by applying it to two face verification systems: a holistic system based on PCA-derived features, and a local feature system based on DCT-derived features (Sanderson & Paliwal, 2002). Experiments on the FERET database show that for the holistic system, the LinReg-based technique is more suited than the MLLR-based technique. It also shows that the local feature system is less affected by view changes than the holistic system.

HEAD POSE COMPENSATION

Facial Feature Interpretation

Active shape models (ASMs) (Cootes, 1992; Cootes, Taylor, Cooper & Graham, 1995) and active appearance models (AAMs) (Cootes, Edwards & Taylor, 2001; Edwards, Taylor & Cootes, 1998), first introduced by Cootes and Taylor, is the most famous and robust approach to build such deformable models and used to interpret images.

Statistical Models of Shape and Appearance

The shape of an object is represented by a set of *n* landmarks, which can be in any dimension. Good landmarks are the points that lie at the corners of object boundaries or obvious biological landmarks. For instance, a face image can be labeled by N landmark points, $\{(x_i, y_i)\}$, which are located on key feature points such as the eyes, nose, mouth, chin, and so forth, and be represented by a 2N element vector x:

$$x = (x_1, x_2, \ldots, x_N, y_1, y_{2,\ldots,} y_N)^T$$

After shape normalization, we apply principal component analysis (PCA) to the sample shape to reduce the dimensionality. Any shape x in the training set can be represented by:

$$x = \bar{x} + P_s b_s$$

where $\bar{x}$ is the mean shape vector, P_s is a matrix containing the k eigenvectors with largest eigenvalues, and b_s is a vector representing the weighting of the eigen-components.

We can also generate a statistical appearance model to represent texture variations. We warp each training sample to the mean shape to form a texture vector g_{im}, which is the texture covered by the mean shape. We then apply PCA to these data and use similar methods to generate statistical shape models. Any texture g in the training set can be represented as:

$$g = \bar{g} + P_g b_g$$

where $\bar{g}$ is the mean appearance vector, P_g is the matrix describing the appearance variations matrix learned from the training sets, and b_g is the component vector.

Combination of Statistical Shape and Appearance Models

The shape and appearance parameters b_s and b_g can be used to describe the shape and appearance of any example. As there are correlations between the shape and appearance variations of the same person, we can apply PCA to these data:

$$b = \begin{pmatrix} W_s b_s \\ b_b \end{pmatrix} = \begin{pmatrix} W_s P_s^T (x - \bar{x}) \\ P_g^T (g - \bar{g}) \end{pmatrix}$$

where W_s is a diagonal matrix that represents the change between shape and texture. We apply PCA to these vectors to get:

$$b = P_c\, c$$

where P_c are eigenvectors and c is a vector of appearance parameters controlling both shape and texture of the model.

$$x = \bar{x} + Q_s c$$

where

$$Q_s = P_s W_s^{-1} P_{cs}$$
$$Q_g = P_g P_{cg}$$
$$P_c = \begin{pmatrix} P_{cs} \\ P_{cg} \end{pmatrix}$$

Here, $\bar{x}$ is the mean shape, $\bar{g}$ is the mean appearance, Q_s is the matrix describing the shape variations that have the shape eigenvectors at its columns, and Q_g is the matrix describing the texture that has the texture eigenvectors as its columns. The vector of components c is used to control the shape and texture change.

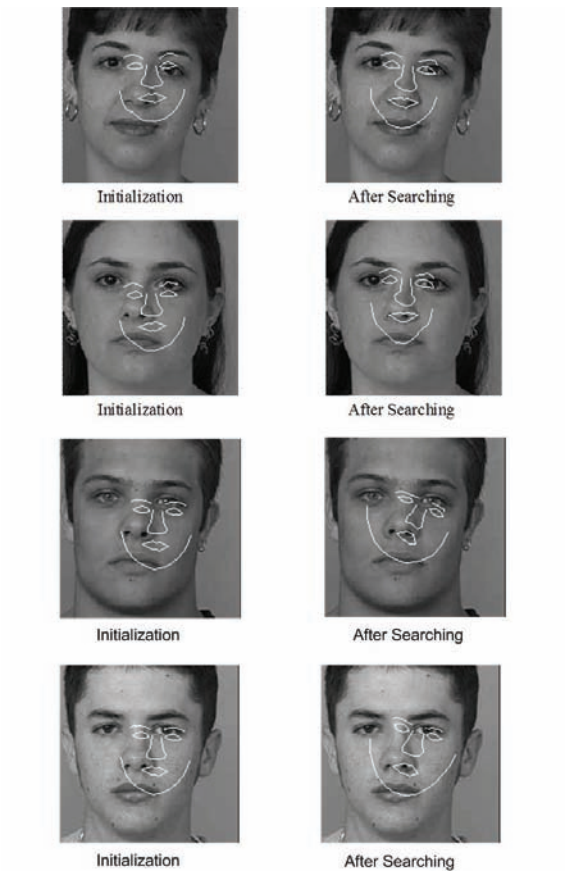
Interpreting Images Using Active Shape and Appearance Models

Interpreting images means we try to find a set of model parameters containing shape, orientation, scale, position, and texture information of this object to generate a sample of this model that can best match the input image. Given a novel image, active shape models are used to interpret the shape information, and active appearance models are used to interpret the texture information.

Combination of Cascade Face Detector with Active Appearance Model Search

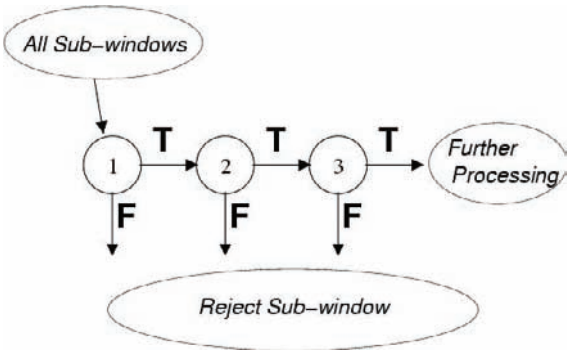
The initialization of the active appearance model search is a critical problem since the original AAM search is a local gradient ascent. Some failed AAM searches due to the poor initialization can be seen in Figure 5.

Figure 5. Failed AAM search due to poor initialization on images from Feret Database (NIST, 2001)



In 2001, Viola and Jones (2001) proposed an image-based face detection system that can achieve remarkably good performance in real time. The main idea of their method is to combine weak classifiers based on simple binary features, which can be computed extremely fast. Simple rectangular Haar-like features are extracted; face and nonface classification is done using a cascade of successively more complex classifiers that discards nonface regions and only sends facelike candidates to the next layer's classifier. Thus, it employs a "coarse-to-fine" strategy (Fleuret & Geman, 2001). Each layer's classifier is trained by the AdaBoost learning algorithm. AdaBoost

Figure 6. The cascade detection process



is a boosting learning algorithm that can fuse many weak classifiers into a single more powerful classifier. Our face detector is based on the Viola-Jones approach using our own training sets, as illustrated by Figure 6.

The cascade face detector finds the location of a human face in an input image and provides a good starting point for the subsequent AAM search. which then precisely marks the major facial features such as mouth, eyes, nose, and so forth.

Pose Estimation

Here we follow the method of Cootes, et al. (2000). They assume that the model vector c is related to the viewing angle, θ, approximately by a correlation model:

$$c = c_0 + c_c \cos(\theta) + c_s \sin(\theta)$$

where c_0, c_c and c_s are vectors that are learned from the training data. (Here we consider only head turning. Head nodding can be dealt with in a similar way).

For each of the images labeled with pose θ in the training set, we perform active appearance models search to find the best fitting model parameters c_i;

then c_0, c_c, and c_s can be learned via regression from the vectors $\{c_i\}$ and vectors $\{(1, \cos\theta_i, \sin\theta_i)'\}$. Given a new face image with parameters c, we can estimate orientation as follows. We first transform $c = c_0 + c_c \cos(\theta) + c_s \sin(\theta)$ to:

$$c - c_0 = (c_c c_s)\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}$$

Using R_c^{-1}, which is the left pseudo-inverse of the matrix $(c_c \mid c_s)$, then this becomes:

$$R_c^{-1}(c - c_0) = \begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}$$

Letting $(x_\alpha, y_\alpha)' = R_c^{-1}(c - c_0)$, then the best estimate of the orientation is:

$$\theta = \tan^{-1}(y_\alpha / x_\alpha)$$

The estimation of θ is not entirely accurate due to landmark annotation errors or possibly regression learning errors. But this model appears to be quite sufficient to be used to synthesize the face images for recognition.

Frontal View Synthesis

After we estimate the angle θ, we can use the model to synthesize new views. Here we will synthesize a frontal view face image, which will be used for face recognition.

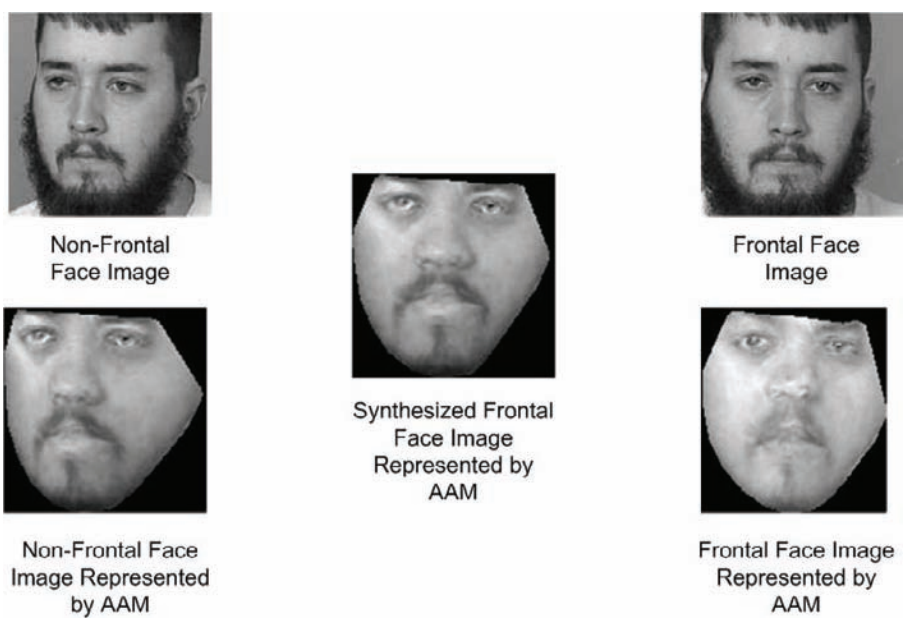
Let c_{res} be the residual vector, which is not explained by the correlation model:

$$c_{res} = c - (c_0 + c_c \cos(\theta) + c_s \sin(\theta))$$

To reconstruct at a new angle, α, we simply use the parameters:

$$c(\alpha) = c_0 + c_c \cos(\alpha) + c_s \sin(\alpha) + c_{res}$$

Figure 7. (L to R) Frontal view, turned view, synthesized frontal view from turned view using frontal rotation model on a face image from the Feret Database (NIST, 2001)



As we want to synthesize frontal view face image, α is 0, so this becomes:

$$c(0) = c_0 + c_c + c_{res}$$

The shape and texture at angle 0° can be calculated by:

$$x(0) = \bar{x} + Q_s c(0)$$

and

$$g(0) = \bar{g} + Q_g c(0)$$

The new frontal face image then can be reconstructed. A front view synthesized from the frontal correlation model can be seen in Figure 7.

RESULTS FROM EXPERIMENTS

Training of Frontal Face Models

To train the frontal face model, we collected face image samples from 40 individuals for the training set. For each person, we had three im-

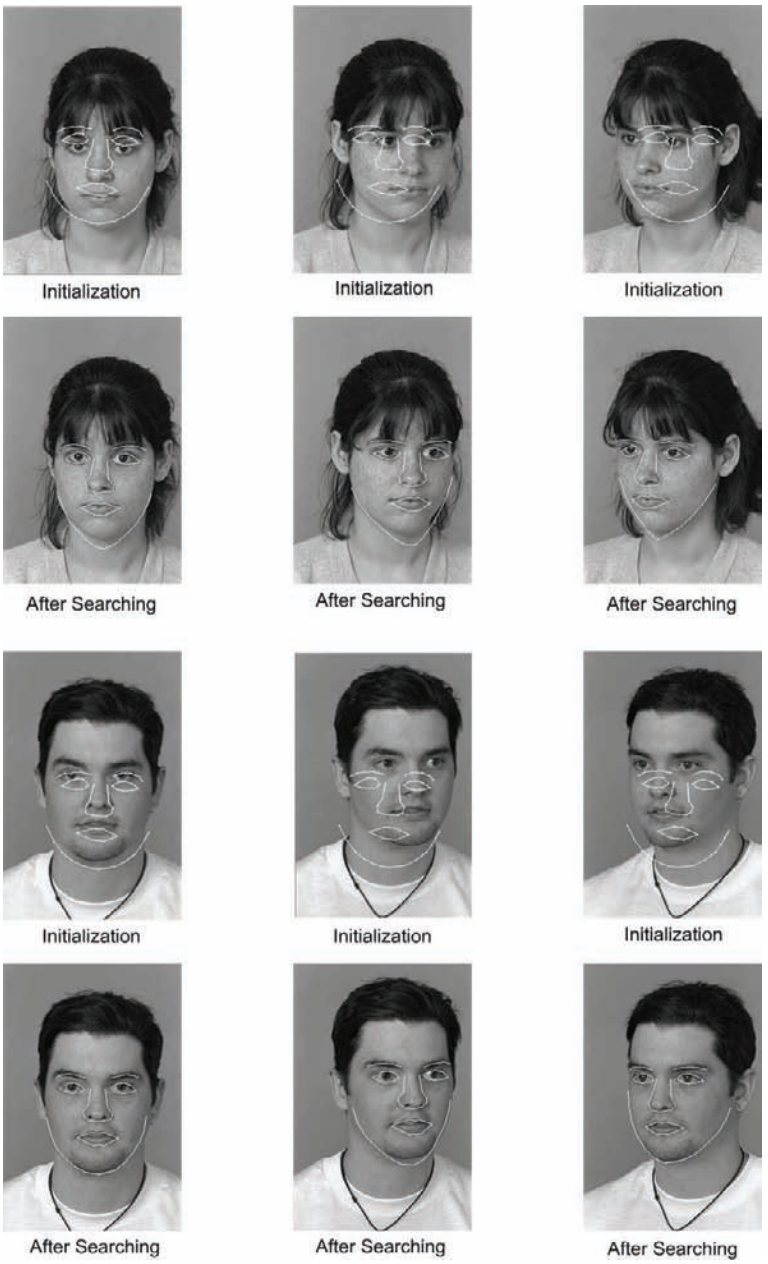
Figure 8. Sample training face images from Feret database (NIST, 2001) with 58 points labeled on the main facial features to determine the face model

ages in three pose views (left 15°, frontal, right 15°); all these images are extracted from the Feret database (NIST, 2001). We labeled each of these 120 face images with 58 points around the main features, including eyes, mouth, nose, eyebrows, and chin. Some labeled training face images can be seen in Figure 8.

Searching Results of Active Shape Models (ASMs) and Active Appearance Models (AAMs)

Some search results of active shape models and active appearance models on the Feret Database (NIST, 2001) using the statistical appearance

Figure 9. Active shape models (ASMs) search results on Feret database (NIST, 2001)



models we trained can be seen in Figure 9 and Figure 10.

Training of Correlation Model

We then apply our combined AdaBoost-based cascade face detector and AAM search on the rest of the images of the Feret b-series dataset where each person has seven pose angles, ranging from left 25°, 15°, 0°, to right 15°, 25°. Despite Cootes, et al.'s statement that a near frontal face model can deal with pose change from left 45° to right 45°, we found that the AAM search cannot locate the facial features precisely on most face images with high (40°) pose rotation, even given a very good initialization position, especially for face images with a large noise. AAM searching on the remaining face images with smaller pose angle

Figure 10. Some active appearance models (AAMs) search results on Feret database (NIST, 2001)

change can achieve 95% search accuracy rate. Frontal parameters c_0, c_c, and c_s are learned from the successful AAM search samples. By using the same method, we get the rotation parameters for left and right rotation model, respectively.

Synthesis Results from a Frontal Face Image

Given a frontal face image as in Figure 11, Figure 12 shows the synthesized results from left 45° to right 45°.

Figure 11. Original frontal face image from Feret database (NIST, 2001)

Figure 12. Synthesized results from left 45° to right 45°

High Pose Angle Face Recognition Results

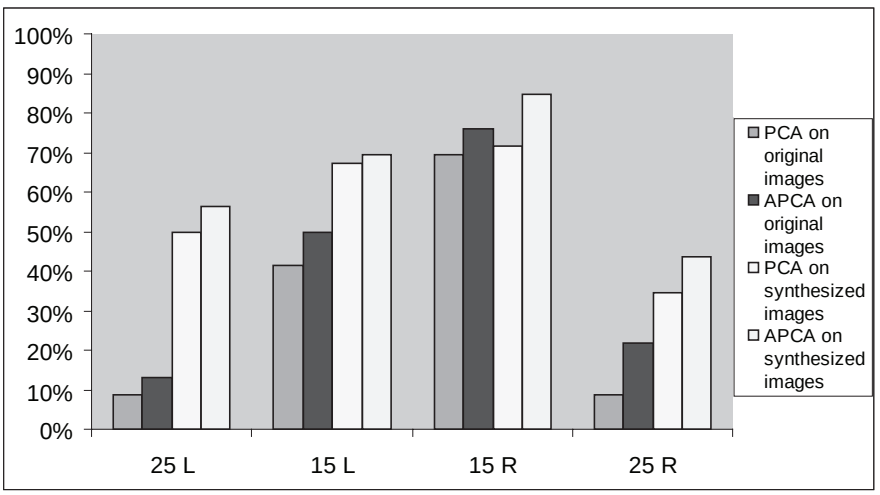
We trained our APCA face recognition model using the Asian Face Database (I.M. Lab). We selected face images from 46 persons with good AAM search results; each person had seven pose angles ranging from left 25°, 15°, 0°, to right 15°, 25°. We then formed two datasets: one is the original image set and the other is the synthesized frontal image set; each of them contains 322 images. We only registered the frontal view images into the gallery and applied both PCA and APCA on the high pose angle images for testing (276 images on each of the datasets). The overall recognition results from the threefold cross-validated trials are shown in Figure 13.

It can be seen from Figure 13 that the recognition rates of PCA and APCA on synthesized images is much higher than that of the original high pose angle images. The recognition rate increases by up to 48 percentage points from 9% to 57% for images with a view angle of 25º. Yet even for smaller rotation angles less than 15º, the accuracy increases by up to 29 percentage points from 50% to 79%. Note that the recognition performance of APCA is always significantly higher than PCA, which is consistent with the results in Chen and Lovell (2004) and Lovell and Chen (2005).

REAL-TIME AUTOMATED FACE RECOGNITION SYSTEMS

To date, the majority of the research work on automated face recognition (AFR) has focused primarily on developing novel algorithms and/or improving the efficiency and accuracy of existing algorithms. As a result, most solutions developed (similar to the examples given in previous sections) are typically high-level software programs targeted for general-purpose processors that are expensive and usually nonreal-time solutions. Since face recognition is typically the first step and frequently a bottleneck in most solutions due to the large search space and computationally intensive operations, it is reasonable to suggest an embedded implementation specifically optimized to detect faces and recognize them. An embedded solution would entail many advantages such as cost and miniaturization, as only a subset of the hardware components are required compared to

Figure 13. Recognition rate for PCA and APCA on original and synthesized images with small rotation angles



the general computer-based solutions. The resulting solution can then be integrated with other technologies such as security cameras to create smart devices.

Now that reliable, accurate, and efficient face recognition algorithms are available, coupled with advances in embedded technologies, low-cost implementations of robust real-time face detectors can be explored. The following subsection discusses the common embedded technologies and known embedded implementations of automatic face recognition (AFR) systems.

Related Work

There is a vast range of embedded technologies and associated design methodologies that can be employed for the design of an embedded AFR system. The most common technologies are pure hardware, embedded microprocessors, and configurable hardware.

Pure hardware systems are typically based on very large-scale integrated circuit (VLSI) semiconductor technology implemented as application-specific integrated circuits (ASIC). Compared to the other technologies, ASICs have a high operating frequency resulting in better performance, low power consumption, high degree of parallelism, and well-established design tools. However, a large amount of development time is required to optimize and implement the designs. Also, due to the fixed nature of this technology, the resulting solutions are not flexible and cannot be easily changed, resulting in high development costs and risk.

On the other hand, software programs implemented on general purpose processors (GPPs) offer a great deal of flexibility, coupled with very well-established design tools that can automatically optimize the designs; little development time and costs are required, thus assuming less risk. GPPs are ideally suited to applications that are primarily made up of a control processing because operations are carried out sequentially (one operation after another). However, they are disadvantaged because minimal or no special instructions are available to assist with data processing (B.D.T. Inc., 2004). Digital signal processors (DSPs) extend GPPs in the direction of increasing parallelism and providing additional support for applications requiring large amounts of data processing. The drawbacks of microprocessors (both GPPs and DSPs) are high-power consumption and inferior performance compared to an ASIC. The performance of the final solution is limited to the selected processor.

Finally, configurable platforms such as field programmable gate arrays (FPGAs) combine some of the advantages from both pure hardware and pure software solutions (B.D.T. Inc., 2002); more specifically, the high parallelism and computational speed of hardware and the flexibility and short design cycle of software. By inheriting characteristics from both hardware and software solutions, naturally the design space for FPGAs is extended for better trade-offs between performance and cost. These design trade-offs are far superior to that of pure hardware or software solutions alone (B.D.T. Inc., 2005). From an efficiency point of view, the performance measures for FPGAs (i.e., operating frequency, power consumption, etc.) are generally halfway between the corresponding hardware and software measures.

VLSI-Based Face Detector

Theocharides, Link, Vijaykrishnan, Irwin, and Wolf (2004) investigated the implementation of a neural network-based face detection algorithm in 160nm VLSI technology. The authors' motivation in selecting this technology is due largely to its ability to support a compact and high-speed implementation. The face detection algorithm used is that proposed by Rowley, Baluja, and Kanade (1998a), which is selected for the following properties: high detection rate, faces can be in different orientations, and the algorithm has a high degree of parallelism. The last property is

the most important one in pure hardware-based implementations.

The implemented system operates on grey-scale images with a dimension of 300x300 pixels exhibiting the following operating characteristics: clock frequency of 409.5 KHz, power consumption of 7.35 Watts, and area of 30.4 mm2. Though the operating frequency is relatively low, a throughput of 424 images per second was achieved. The system is also able to maintain a reasonably high detection accuracy of 75%, which is comparable to its software counterpart.

As illustrated by Theocharides, et al. (2004), a VLSI-based implementations can yield an accurate solution with high throughput and low power consumption. However, as mentioned previously, pure hardware-based systems are extremely inflexible. If any system parameters require changing, for example, the size of the input image, the overall system including the environment in which the circuit is integrated, may need to be redesigned.

Face Detection Implemented on Configurable Platforms

Several configurable hardware-based implementations exist, including that by McCready (2000), Sadri, et al. (2004), and Paschalakis and Bober (2003). McCready (2000) specifically designed a novel face detection algorithm for the Transmogrifier-2 (TM-2) configurable platform. The Transmogrifer-2 is a multiboard FPGA-based architecture proposed by Lewis, Galloway, Van Ierssel, Rose, and Chow (1998) that is made up of between one and 16 boards. Each prototyping board consists of two Altera Flex 10K100 FPGA chips, four ICube field programmable interconnect device (FPID) chips, and 8MB of SRAM, and operates at a frequency of 12.5 MHz.

The algorithm was intentionally designed with minimal mathematical operations that could execute in parallel—engineering effort has been put in to reduce the number of multiplications required. The implemented system required nine boards of the TM-2 system, requiring 31,500 logic cells (LCs). The system can process 30 images per second with a detection accuracy of 87%. The hardware implementation is said to be 1,000 times faster than the equivalent software implementation.

On the other hand, Sadri, et al. (2004) implemented the neural network-based algorithm proposed by Rowley, et al. (1998) on the Xilinx Virtex-II Pro XC2VP20 FPGA. Skin color filtering and edge detection is incorporated to reduce the search speed. The solution is partitioned such that all regular operations are implemented in hardware, while all irregular control-based operations are implemented on Xilinx's embedded hardcore PowerPC processor. This partitioning allows the advantages of both hardware and software to be simultaneously exploited. The system operates at 200MHz and can process up to nine images per second. Inadequate information is given regarding the resource usage, but a minimum of 18,000 look-up tables (LUTs) and 2Mbytes of memory are required.

Finally, Paschalakis and Bober (2003) designed a system with the intention of integrating it within a power- and resource-constrained environment such as 3G mobile phones, where the face can be tracked for mobile conferencing applications. The algorithm is based on skin color modeling, which makes use of the LogRG 2D color space proposed by Forsyth, et al. (2005). Essentially, detection is implemented by identifying the largest skin region and tracking the change in the centroids of these regions over subsequent frames. Strictly speaking, this is not face detection, but it is adequate for the application in which it is intended. The system is implemented on an Altera Apex20K EP20K1000EBC652-1 device requiring 8.3% of the total logic elements (LE), 1.4% of embedded system blocks (ESB), and 700 bytes of memory. The operating frequency is 33MHz and can process up to 434 frames per second.

The examples presented illustrate the obvious compromises between accuracy and algorithm robustness vs. the amount of resources required; that is, to improve the performance of the face detection algorithms, either increase the embedded design complexity, which generally results in higher power consumption and hardware costs, or settle for the less superior solution.

System Level-Based Design Methodology

Kianzad, et al. (2005) present a framework for designing and exploring different possible face detection implementations on embedded reconfigurable systems. The face detection algorithm employed is that presented by Moon, Chellappa, and Rosenfeld (2002), which is based on using edge information to detect objects in 2D. In order to exploit parallelism, multiple processors are instantiated and synchronous dataflow diagrams based on synchronization graph models are used to analyze and schedule task execution on each processor. The proposed design was implemented on the Xilinx ML310 development board and is able to process 12 frames per second.

The problem with this proposed design approach is that the model is not easily extendable to other face detection algorithms and hardware platforms. The models created are tightly coupled with the face detection algorithm and hardware architecture used. As a result, in order to consider different design alternatives, several models will need to be created.

DSP-Based Implementations

There are two DSP-based AFR system implementations by Batur, Flinchbaugh, and Hayes (2003) and Wei and Bigdeli (2004) that are reported in the literature.

The implementation by Batur, et al. (2003) consists of four stages: face detection, face feature localization, face normalization, and face recognition. The probabilistic visual learning algorithm proposed by Moghaddam and Pentland (1997) is used for face detection. The system was implemented on the Texas Instruments TMS320C6416 fixed point DSP operating at 500 MHz.

On the other hand, the system implemented by Wei and Bigdeli (2004) is made up of three stages: image normalization, face detection, and face recognition. Where face detection is performed using the algorithm proposed by Rowley, et al. (1998). The system was implemented on the Analog Devices ADSP-BF535 EZ-KIT Lite development board containing a 16-bit fixed point DSP operating at 300MHz.

Similar optimization techniques were employed by both systems, including converting floating point operations to fixed point, write time-consuming functions in assembly, use available parallelism in the DSPs, and use look-up tables in place of complex arithmetic operations. Unfortunately, both implementations still require over a second to process an image where face detection consumes the majority of the processing time.

Future Directions

Field programmable gate array technology has gained considerable attention over the last decade as an established platform for embedded systems. With the improvement of design tools, it is a platform that supports easy integration of both hardware and software solutions, allowing their advantages to be easily exploited. Software support takes the form of embedded processors, which provides flexibility, while hardware support is in the form of custom instructions, custom peripherals, or coprocessors allowing parallel and fast execution. The advantage of this design paradigm is that existing high-level software solutions can be easily ported onto an embedded system, and unlike GPPs or DSPs, hardware support is readily available. Also, this form of hardware-software partitioning provides an easy means of continual design refinement directed toward

evolving system specifications and performance improvements.

As most existing AFR algorithms inherently make use of complex operations and are generally not parallel, pure hardware- or software-based implementations are not the most ideal solutions. Our proposed method will use an embedded soft-core processor integrated with custom processor instructions to aid with complex operations relating to the face detection algorithm.

Optimization Using Custom Instructions

Configurable custom processors are becoming an ever more popular implementation technology of choice for addressing the demands of complex embedded applications. Unlike traditional hard-wired processors that consist of a fixed instruction set from which application code is mapped, configurable processors can be augmented with application-specific instructions, implemented as hardware logic to optimize bottlenecks. This lends toward a method for hardware-software partitioning whereby the efficiency of hardware and the flexibility of software are integrated.

There is a number of benefits in extending a configurable processor with custom instructions. First, transparency: the added custom instructions will improve the performance of the tasks for which they are designed with minor changes to the original code. Second, rapid development and short time-to-market: there is a wide variety of off-the-shelf configurable cores that could be used as a base for development. Additional instructions could be integrated into the processor core as the need to extend its computational capabilities arises. Finally, low-cost access to domain specific processors: generally, the fundamental characteristics of an application area are similar. These characteristics can be summarized as a set of instructions and applied to a variety of similar applications; for example, multimedia applications (Bigdeli, Biglari-Abhari, Leung & Wang, 2004).

Unfortunately, there are two minor drawbacks to using custom instructions. For one, additional hardware is required, as such a bank of resources needs to be set aside specifically for custom instructions. This is becoming less of an issue as embedded technologies become more economical. Second, as the custom instructions are directly integrated into the processor's pipeline, the maximum operating frequency may be degraded if the instruction is poorly designed.

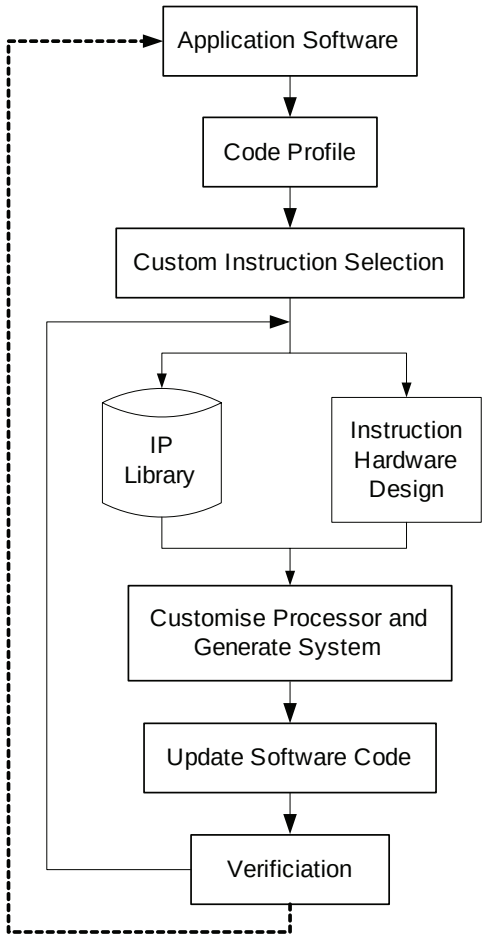
Augmenting configurable processor cores with custom instructions is a proven optimization technique that has been applied to a wide and varied range of embedded applications. Some published examples include encryption (Groszschaedl, Kumar & Paar, 2004; Juliato, Araujo, Lopez & Dahab, 2005; Tai-Chi, Zeien, Roach & Robinson, 2006), audio encoding (Bower, 2004), embedded real-time operating systems (RTOS) (Oliver, Mohammed, Krishna & Maskell, 2004), biometrics (Aarajt, Ravit, Raghunathant & Jhat, 2006; Gupta, Ravit, Raghunathan & Jha, 2005), and multimedia (Tsutsui, Masuzaki, Izumi, Onoye & Nakamura, 2002).

Custom Instruction Design Flow

The design flow for identifying and integrating custom instructions into configurable processors is summarized in Figure 14. This is a generic framework that could be applied to any application. First, the software code is profiled to reveal bottlenecks that could be alleviated with the introduction of custom instructions. Obviously, the operation, which has the most significant impact on the overall performance, is optimized first.

There are two approaches to designing the new custom instructions. One method is to reuse IP cores already available in the development suite or third-party libraries, which will further shorten development time. Alternatively, the instructions are designed from scratch. This generally requires greater engineering effort, but the designer has better control over design decisions.

Figure 14. Custom instructions design flow



Once the hardware module for the instruction is implemented and tested, it is added to the processor, and the whole system is regenerated. Then the software code is updated to make use of the new instructions. Finally, the functionality of the system is verified to ensure bugs are not introduced with the new instruction. This process is repeated until either the performance requirements or resource limits are met.

Design Considerations

As discussed in previous sections, in practice, most cameras such as those used in CCTV are positioned so that capturing frontal image is not possible. Furthermore, in real-world applications, lighting condition is not usually desirable. In order to achieve real-time performance, a combination of optimization techniques that are low resolution images, fixed-point arithmetic, conversion of key functions to low-level codes, and most importantly through custom instruction, are applied to improve the overall system speed and performance.

CONCLUSION AND FUTURE WORK

In this chapter, we described the face recognition algorithm adaptive principal component analysis (APCA) and rotated adaptive principal component analysis (RAPCA), which are insensitive to illumination and expression variations. We then extend our previous work to multiview face recognition by interpreting facial features and synthesizing realistic frontal face images when given a single novel face image. The experimental results show that after frontal pose synthesis, the recognition rate increases significantly, especially for larger rotation angles.

Furthermore, we examined how an automated face recognition system can be implemented on embedded systems. We also explored various design approaches. We currently have two prototype systems for the real-time automated face recognition. The first prototype was entirely implemented on an Analog Devices Blackfin DSP processor capable of verifying a face from a database of 16 faces under a second. This was done as a replacement for PIN identification on a NOKIA mobile phone. The second prototype was developed using a hardware-software approach on a NIOS II processor with extended instructions. The NIOS II processor was configured on an Altera FPGA.

In our continuing work, we will extend the work described in this chapter to pose angle change larger than 25º. We also will use the face models and correlation models developed in this chapter to synthesize virtual views under different lighting

conditions, facial expressions, and poses when given a frontal face image. These synthesized virtual images can be used as training samples for face recognition algorithm, like support vector machine (SVM) or neural network. Thus, we can form a face recognition system using dual algorithms: one is adaptive principal component analysis, and the other is SVM- or neural network-based algorithm, which would enhance the system performance.

ACKNOWLEDGMENT

This project is supported by a grant from the Australian Government Department of the Prime Minister and Cabinet. NICTA is funded by the Australian government's Backing Australia's Ability initiative, in part through the Australian Research Council.

REFERENCES

Aarajt, N., Ravit, S., Raghunathant, A., & Jhat, N.K. (2006). Architectures for efficient face authentication in embedded systems. *Proceedings of the Design, Automation and Test in Europe Conference*. 2, 1–6.

Adini, Y., Moses, Y., & Ullman, S. (1997). Face recognition: The problem of compensating for changes in illumination direction. *Pattern Analysis and Machine Intelligence, IEEE Transactions, 19*(7), 721–732.

Batur, A.U., Flinchbaugh, B.E., & Hayes III, M.H. (2003). A DSP-based approach for the implementation of face recognition algorithms. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2, 253–256.

B.D.T. Inc. (2002). Comparing FPGAs and DSPs for embedded signal processing.

B.D.T. Inc. (2004). Using general-purpose processors for signal processing. *Proceedings of the ARM Developers' Conference*.

B.D.T. Inc. (2005). Processors for consumer audio/video applications. *GSPx*.

Belhumeur, P.N., Hespanha, J.P., & Kriegman, D.J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 19*(7), 711–720.

Beymer, D., & Poggio, T. (1995). Face recognition from one example view. *Proceedings of Fifth International Conference on Computer Vision*, 500–507.

Bigdeli, A., Biglari-Abhari, M., Leung, S.H.S., & Wang, K.I.K. (2004). Multimedia extensions for a reconfigurable processor. *Proceedings of the 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing*, 426–429.

Black, M.J., Fleet, D.J., & Yacoob, Y. (2000). Robustly estimating changes in image appearance. *Computer Vision and Image Understanding, 78*(1), 8–31.

Bower, J. (2004). A system-on-a-chip for audio encoding. *Proceedings of the International Symposium on System-on-Chip,* 149–155.

Chellappa, R., Wilson, C.L., & Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE,* 705–741.

Chen, S., & Lovell, B.C. (2004). Illumination and expression invariant face recognition with one sample image. *Proceedings of the 17th International Conference on Pattern Recognition*. 1, 300–303.

Cootes, T.F., Edwards, G.J., & Taylor, C.J. (2001). Active appearance models. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 23*(6), 681–685.

Cootes, T.F., & Taylor, C.J. (1992) Active Shape Models - 'Smart Snakes'. *Proc. British Machine Vision Conference* (pp. 266-275). Springer-Verlag.

Cootes, T.F., Taylor, C.J., Cooper, D.H., & Graham, J. (1995). Active shape models—Their training and application. *Computer Vision and Image Understanding, 61*(1), 38–59.

Cootes, T.F., Walker, K., & Taylor, C.J. (2000). View-based active appearance models. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 227–232.

Darrell, T., & Pentland, A. (1993). Space-time gestures. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 335–340.

Edwards, G. J., Taylor, C.J., & Cootes, T.F. (1998). Interpreting face images using active appearance models. *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, 300–305.

Fleuret, F., & Geman, D. (2001). Coarse-to-fine face detection. *International Journal of Computer Vision, 41*(1-2), 85–107.

Forsyth, D., Fleck, M., & Bregler, C. (1996). Finding naked people. *Proceedings of the 1996 European Conference on Computer Vision*, Vol. 2, 592–602.

Groszschaedl, J., Kumar, S.S., & Paar, C. (2004). Architectural support for arithmetic in optimal extension fields. *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 111–124.

Gupta, P., Ravit, S., Raghunathan, A., & Jha, N.K. (2005). Efficient fingerprint-based user authentication for embedded systems. *Proceedings of the Design Automation Conference*, 244–247.

I.M. Lab. *Asian face image database*. Retrieved from http://nova.postech.ac.kr/

Juliato, M., Araujo, G., Lopez, J., & Dahab, R. (2005). A custom instruction approach for hardware and software implementations of finite field arithmetic over F/sub 2163/ using Gaussian normal bases. *Proceedings of the IEEE International Conference on Field-Programmable Technology*, 5–12.

Kianzad, V., et al. (2005). An architectural level design methodology for embedded face detection. *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 136–141.

Lades, M., et al. (1993). Distortion invariant object recognition in the dynamic link architecture. *Computers, IEEE Transactions, 42*(3), 300–311.

Lewis, D.M., Galloway, D.R., Van Ierssel, M., Rose, J., & Chow, P. (1998). The transmogrifier-2: A 1 million gate rapid-prototyping system. *Proceedings of the IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, 188–198.

Lovell, B.C., & Chen, S. (2005). Robust face recognition for data mining. In J. Wang (ed.), *Encyclopedia of Data Warehousing and Mining*, (965–972). Hershey, PA: Idea Group.

McCready, R. (2000). Real-time face detection on a configurable hardware system. *Proceedings of the Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications*, 157–162.

Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object representation. *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 696–710.

Moon, H., Chellappa, R., & Rosenfeld, A. (2002). Optimal edge-based shape detection. *Proceedings of the IEEE Transactions on Image Processing, 11*, 1209–1227.

Murase, H., & Nayar, S.K. (1993). Learning and recognition of 3D objects from appearance. *Proceedings of the IEEE Workshop on Qualitative Vision*, 39–50.

NIST. (2001). *Feret database*. Retrieved from http://www.itl.nist.gov/iad/humanid/feret/

Oliver, T.F., Mohammed, S., Krishna, N.M., & Maskell, D.L. (2004). Accelerating an embedded RTOS in a SoPC platform. *Proceedings of the TENCON Conference*, 4, 415–418.

Paschalakis, S., & Bober, M. (2003). A low cost FPGA system for high speed face detection and tracking. *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT)*, 214–221.

Pentland, A., Moghaddam, B., & Starner, T. (1994). View-based and modular eigenspaces for face recognition. *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 84–91.

Phillips, P.J., Grother, P., Micheals, R., Blackburn, D.M., Tabassi, E., & Bone, M. (2003). Face recognition vendor test 2002. *Proceedings of the Analysis and Modeling of Faces and Gestures*, 44.

Rowley, H.A., Baluja, S., & Kanade, T. (1998a). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 23–38.

Rowley, H.A., Baluja, S., & Kanade, T. (1998b). Rotation invariant neural network-based face detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 38–44.

Sadri, M.S., et al. (2004). An FPGA based fast face detector. *Proceedings of the Global Signal Processing Expo and Conference*.

Sanderson, C., Bengio, S., & Gao, Y. (2006). On transforming statistical models for non-frontal face verification. *Pattern Recognition*, *39*(2), 288–302.

Sanderson, C., & Paliwal, K.K. (2002). Fast feature extraction method for robust face verification. *Electronics Letters*, *38*(25), 1648–1650.

Tai-Chi, L., Zeien, R., Roach, A., & Robinson, P. (2006). DES decoding using FPGA and custom instructions. *Proceedings of the International Conference on Information Technology: New Generations*, 575–577.

Theocharides, T., Link, G., Vijaykrishnan, N., Irwin, M.J., & Wolf, W. (2004). Embedded hardware face detection. *Proceedings of the 17th International Conference on VLSI Design*, 133–138.

Tsutsui, H., Masuzaki, T., Izumi, T., Onoye, T., & Nakamura, Y. (2002). High speed JPEG2000 encoder by configurable processor. *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, 1, 45–50.

Turk, M.A., & Pentland, A.P. (1991a). Face recognition using eigenfaces. *Computer Vision and Pattern Recognition*, 586–591.

Turk, M., & Pentland, A. (1991b). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*(1), 71–86.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *1*, I-511–I-518.

Wei, M., & Bigdeli, A. (2004). Implementation of a real-time automated face recognition system for portable devices. *Proceedings of the IEEE International Symposium on Communications and Information Technology*, *1*, 89–92.

Wiskott, L., Fellous, J.M., Kuiger N., & von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *Pattern Analysis and*

Machine Intelligence, IEEE Transactions, 19(7), 775–779.

Wiskott, L., & von der Malsburg, C. (1996). Face recognition by dynamic link matching. In J. Sirosh, R. Miikkulainen, & Y. Choe (Eds.), *Lateral interactions in the cortex: Structure and function*. Austin, TX: UTCS Neural Networks Research Group.

Xiujuan, C., Shiguang, S., & Wen, G. (2003). Pose normalization for robust face recognition based on statistical affine transformation. *Proceedings of the Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia*, 3, 1413–1417.

Yilmaz, A., & Gokmen, M. (2000). Eigenhill vs. eigenface and eigenedge. *Proceedings of the 15th International Conference on Pattern Recognition*, 2, 827–830.

Yongsheng, G., & Leung, M.K.H. (2002). Face recognition using line edge map. *Pattern Analysis and Machine Intelligence, IEEE Transactions, 24*(6), 764–779.

ENDNOTE

[1] Ten to 15 minutes to recognize a face from 111 models using a computer of the day in 1996 (Wiskott & von der Malsburg, 1996).

Chapter IX
Occlusion Sequence Mining for Activity Discovery from Surveillance Videos

Prithwijit Guha
Indian Institute of Technology - Kanpur, India

Amitabha Mukerjee
Indian Institute of Technology - Kanpur, India

K.S. Venkatesh
Indian Institute of Technology - Kanpur, India

ABSTRACT

Complex multiobject interactions result in occlusion sequences, which are a visual signature for the event. In this work, multiobject interactions are tracked using a set of qualitative occlusion primitives derived on the basis of the persistence hypothesis: objects continue to exist even when hidden from view. Variable length temporal sequences of occlusion primitives are shown to be well correlated with many classes of semantically significant events. In surveillance applications, determining occlusion primitives is based on foreground blob tracking and requires no prior knowledge of the domain or camera calibration. New foreground blobs are identified as putative objects that may undergo occlusions, split into multiple objects, merge back again, and so forth. Significant activities are identified through temporal sequence mining; these bear high correlation with semantic categories (e.g., disembarking from a vehicle involves a series of splits). Thus, semantically significant event categories can be recognized without assuming camera calibration or any environment/object/action model priors.

INTRODUCTION

Unsupervised activity discovery remains one of the more challenging areas in computer vision. Key problems involve the identification of object and action features, and temporal data mining for abstracting the activity from the data. This chapter presents a set of occlusion features that are tracked over time to yield a surprisingly rich inventory of actions involving interactions among multiple objects.

Object interactions in 3D space often leave their imprint in image space in terms of occlusions. Instead of treating occlusions as a problem, we show that temporal sequences of occlusion phenomena constitute a qualitative signature for large classes of events. In particular, events involving actual contact (e.g., push, embark, hit) necessarily involve overlap in image space for part of the event history. However, many classes of noncontact situations also result in occlusions, and the sequence of such occlusions can (depending on viewpoint) lead to characterization of specific events (e.g., overtaking, crossing). We claim that occlusions among objects constitute an inexpensive and cognitively important cue to reasoning about interactions in space.

Here we explore the limits of what can be learned based on occlusion phenomena. The primary advantage of such an approach is that unlike quantitative approaches using supervised priors for object behavior recognition (Gavrilla, 2005; Haritaoglu, Harwood & Davis, 2000; Zhao & Nevatia, 2004), occlusion signatures do not require any priors for either objects or events. From a cognitive perspective, categorizing events by combining occlusion with other low-level features such as trajectory and segmentation may constitute a key part of the process leading to formation of image schema (Mandler, 1992). Working together with preattentive cues such as image flow and motion, temporal learning in sequences of occlusion phenomena may constitute a prelinguistic model for concept formation for both activities and objects. These links to cognitive processes also reflect computational efficiencies to be gained by focusing attention and avoiding more expensive 3D computations as called for in Granlund (2003).

In order to discover events from sequences of occlusion states, we mine variable-length temporal sequences of occlusion primitives and learn signatures for a wide class of actions. For example, a group of people hugging each other; a person coming on a bicycle, getting off, and going into a building; a crowd of people embarking a tempo; a person boarding and riding a cycle (Figure 1) are all events that have stable signatures in terms of occlusion primitives or *O-primitives*. The feature

Figure 1. Person boarding and riding a cycle: (a) Cycle static and person moving—both are isolated from each other; (b) person and cycle are in contact—both are static; (c) person and cycle are in contact—both are moving

(a) (b) (c)

set for sequence mining constitutes the O-primitives together with quantified motion features.

Occlusion mining is especially relevant for activities where object trajectories result in occlusion at least for part of its scene presence. Such activities may include:

- Objects interacting without contact (e.g., two cars crossing each other)
- Limited contact situations (e.g., two persons shaking hands)
- Objects completely enclosed within other objects (e.g., people entering a bus)
- Objects emerging from other objects (e.g., disembarking a vehicle)

Occlusion signatures constitute an extremely general feature set; clearly finer discrimination calls for more object specific features (e.g., in distinguishing actions such as hugging from handshaking).

Once events are discovered from sequence mining, an important discovery constitutes the number of objects involved in the activity. These are relevant in conceptual and linguistic models for the action. Single object actions correspond to intransitive verbs in language, while multiple object interactions corresponding to transitive verbs. These are distinguished by the number of objects whose features participate in the event discovery process. Single object actions like "turning left onto the road" are based on a monadic set of features (based on an object's motion vectors), while multiple object interactions mostly involve some degree of occlusion in actions such as overtake, embark, and so forth.

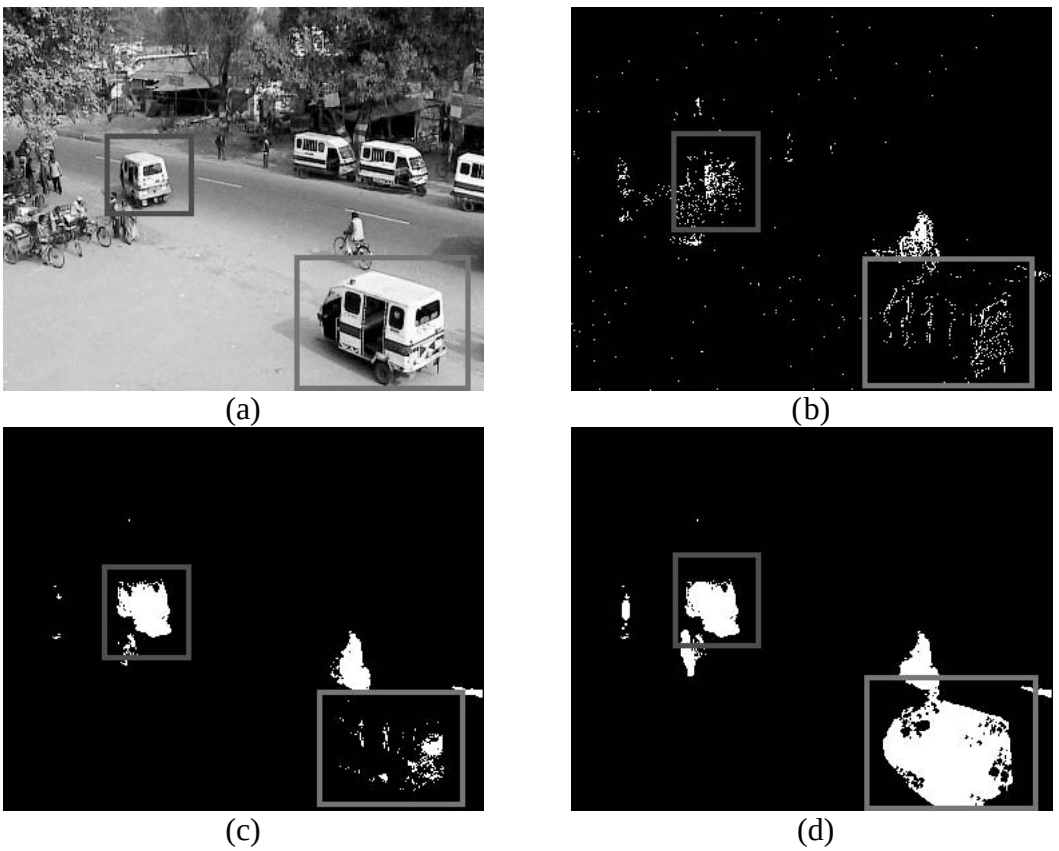
FOREGROUND EXTRACTION

Objects are identified as foreground regions based on one of two kinds of evidence: first, as regions of change with respect to a learned background model; and second, as regions exhibiting motion. Learning the background model in the presence of objects is a challenging problem in itself. Several approaches have been proposed to incrementally learn the background scene model in the presence of objects. The most commonly adopted algorithms include the computation of median (Haritaoglu et al., 2000) or fitting (temporally evolving) Gaussian mixture models (Stauffer & Grimson, 1999; Zivkovic, 2004) on the temporal pixel color histogram of the image sequence. These approaches continuously learn the multimodal mixture models with the assumption that the moving objects appear at a certain pixel only temporarily and the *true* background remains accessible to the system more frequently leading to higher weight of the corresponding mode. However, such an approach is prone to transient errors persisting over a number of frames (depending on the learning rate), resulting in two types of errors. First, if objects learned as part of the background suddenly start moving, ghosts (uncovered background regions classified as foreground) appear in the foreground segmentation. Second, when a moving object comes to stasis, it is eventually learned as a part of the background, which may not be desirable in itself, and also in the transition period, objects interacting with it would not be identified. Both these problems are averted in the present approach by combining background model and motion evidence, and updating based on tracking/previous motion-history feedback.

Generally, the background model $\mathbf{B}_t$ at the $\mathbf{t}^{th}$ instant is selectively updated based on the classification results of the $\mathbf{t}^{th}$ frame $\mathbf{\Omega}_t$. Classification based on $\mathbf{B}_{t-1}$ first results in a set of foreground pixels $\mathbf{F}_b(t) \subseteq \mathbf{\Omega}_t$. Next, an interframe motion estimation (Proesmans, Gool, Pauwels & Osterlinck, 1994) is performed between $\mathbf{\Omega}_t$ and $\mathbf{\Omega}_{t+1}$ to delineate the set of moving foreground pixels $\mathbf{F}_m(t) \subseteq \mathbf{\Omega}_t$. This results in single-frame latency that helps us in identifying the regions that suddenly start moving or come to a stop.

Pixels identified as both foreground and moving are clearly identified as object pixels. Among

Figure 2. Results of foreground detection. (a) The tempo *in the leftmost bounding box starts moving, and the* tempo *to the right has already come to rest; Foreground extraction results (b) using only per pixel Gaussian mixture model with traditional exponential forgetting; (c) with motion detection only ($\boldsymbol{F}_m$) and (d) combining motion evidence with tracking feedback ($\boldsymbol{F}_t$).*



the mismatched pixels, moving pixels not identified as foreground are denoted as $\mathbf{F_{hole}(t) = F_m(t) - F_b(t)}$. On the other hand, the set of nonmoving pixels in $\mathbf{F_b(t)}$, is given by $\mathbf{F_{ghost}(t) = F_b(t) - F_m(t)}$ and is identified as a possible background candidate. However, these nonmoving ghost pixels may contain actual object regions that have not shown up in the optical flow, or where an object has actually come to a stasis. Using information from the motion history and tracking (discussed in the next section), we delineate the set of object pixels that are known to have come to rest, $\mathbf{F_0(t) \subseteq F_{ghost}(t)}$. The set of object pixels that emerge from this analysis is defined as $\mathbf{F(t) = (F_b(t) - F_{ghost}(t)) \cup F_{hole}(t) \cup F_0(t)}$. Now, the complement of **F(t)** is used to update the background model to **B(t)**. The set of detected foreground pixels **F(t)** is further subjected to shadow removal (based on the criteria of equality among subunity intensity modulations in the three color channels), neighborhood voting, followed by connected component analysis to obtain the set of disjoint foreground blobs $\mathbf{F_t = \{ F_i(t) : i = 1,\ldots n_t \}}$. These blobs constitute the basic units (putative objects) that are tracked over the entire sequence, and it is their participation in occlusion that results in O-primitive identification and eventually in the activity discovery.

TRACKING MULTIPLE OBJECTS

Here we adopt the multiobject tracking algorithm proposed in Guha, Mukerjee, and Venkatesh (2005), which works by comparing the foreground blobs at time **t**, $\mathbf{F_i(t)}$ with the predictions based on their previous positions and shape. The same foreground (object) pixel being claimed by more than one object (foreground blob) is one of the primary indicators of occlusion.

We define several elementary occlusion behaviors according to the **persistence hypothesis**: *Objects continue to exist even when hidden from view.* The object-blob association is performed over an *active* set $\mathbf{S_A(t)}$ containing objects tracked until the $\mathbf{t^{th}}$ instant and also a set $\mathbf{S_{lost}(t\text{-}1)}$ of objects that have disappeared within the viewing window. The system initializes itself with empty sets, and the objects are added (removed) as they (dis)appear in the field of view. The proposed approach is a two-stage process. Initially, the objects in $\mathbf{S_A(t\text{-}1)}$ are localized in the current frame $\mathbf{\Omega_t}$. This is followed by the identification of O-primitives by the process of object-blob association with selective updates of object features. This process is detailed next.

Object Representation and Localization

All moving objects are considered as objects, are detected based on extracted foreground blobs, and are initialized with features computed from the blobs. Objects maintain their identity as they are successfully tracked across frames, and even when they are reidentified upon reappearance. The $\mathbf{j^{th}}$ object $\mathbf{A_j(t)}$ is characterized by its supporting region (the set of pixels it occupies, $\mathbf{a_j(t)}$), color (weighted color distribution $\mathbf{h_j(t)}$), and motion (position history of minimum bounding rectangle of $\mathbf{a_j(t)}$, as its previous $\boldsymbol{\tau}$ centers $\mathbf{C_j(t)}$).

The pixel set $\mathbf{a_j(t)}$ and weighted color distribution $\mathbf{h_j(t)}$ are initially learned from the foreground blob extracted at the first appearance of the object and are then updated throughout the sequence whenever it is in isolation. The color distribution $\mathbf{h_j(t)}$ is computed from the **b**-bin color histogram of the region $\mathbf{a_j(t)}$ (in $\mathbf{\Omega_t}$) weighted by the Epanechnikov kernel (Comaniciu, Ramesh & Meyer, 2003) supported over the minimum bounding ellipse of $\mathbf{a_j(t)}$ (centered at $\mathbf{C_j(t)}$) and is given by:

$$h_j[l](t) = \frac{1}{C_E} \sum_{X \in a_j(t)} K_E\left(\left\|X - c_j(t)\right\|^2\right) \delta\left(l - B_f(X)\right) \quad (1)$$

$$C_E = \sum_{X \in a_j(t)} K_E\left(\left\|X - c_j(t)\right\|^2\right) \quad (2)$$

where $\mathbf{C_E}$ is the normalizing constant computed from the Epanechnikov kernel $\mathbf{K_E}$ and the function $\mathbf{B_f}$ maps the pixel location $\mathbf{X} \equiv (x,y)$ to its corresponding color bin derived from the pixel $\mathbf{\Omega_t}(x,y)$.

The objects in the $\mathbf{t^{th}}$ frame are localized by their trajectory information and color distribution obtained until the $\mathbf{(t\text{-}1)^{th}}$ instant. An estimate $\mathbf{C^{(0)}_j(t)}$ is obtained by extrapolating from the trajectory $\mathbf{\{C_j(t\text{-}1),...C_j(t\text{-}\tau)\}}$. The mean-shift iterations (Comaniciu et al., 2003), initialized at an elliptic region centered at $\mathbf{C^{(0)}_j(t)}$ further localize the object region at $\mathbf{a_j(t) \in \Omega_t}$.

Identifying Occlusion Primitives

The extent of association between a predicted object region $\mathbf{a_j(t)}$ for an object in $\mathbf{S_A(t\text{-}1) = \{ A_j(t\text{-}1) : j = 1,\ldots m_{t\text{-}1} \}}$ and the foreground blob $\mathbf{F_i(t) \in F(t)}$ is estimated by constructing a thresholded *localization confidence matrix* $\mathbf{\Theta_{AF}(t)}$ and the *attribution confidence matrix* $\mathbf{\Psi_{FA}(t)}$. These confidences are computed by a fractional overlap measure:

$$\gamma(\varpi_1, \varpi_2) = \frac{|\varpi_1 \cap \varpi_2|}{|\varpi_1|}$$

signifying the fraction of the region ϖ_1 overlapped with ϖ_2.

$$\Theta_{AF}[j,i](t)=\begin{cases}1; & \gamma\left(a_j(t),F_i(t)\right)\geq\eta_A\\ 0; & \textit{Otherwise}\end{cases} \quad (3)$$

$$\psi_{FA}[i,j](t)=\begin{cases}1; & \gamma\left(F_i(t),a_j(t)\right)\geq\eta_F\\ 0; & \textit{Otherwise}\end{cases} \quad (4)$$

where the thresholds η_A and η_F signify the extent of allowable localization and attribution confidences. The number of foreground regions attributed to the **j**$^{\textbf{th}}$ object can be computed as either,

$$\Theta_A[j](t)=\sum_{i=1}^{n_t}\Theta_{AF}[j,i](t)$$

or,

$$\psi_A[j](t)=\sum_{i=1}^{n_t}\psi_{FA}[i,j](t).$$

Similarly, objects localized in $\mathbf{F_i(t)}$ can also be obtained as,

$$\Theta_F[i](t)=\sum_{j=1}^{m_{t-1}}\Theta_{AF}[j,i](t)$$

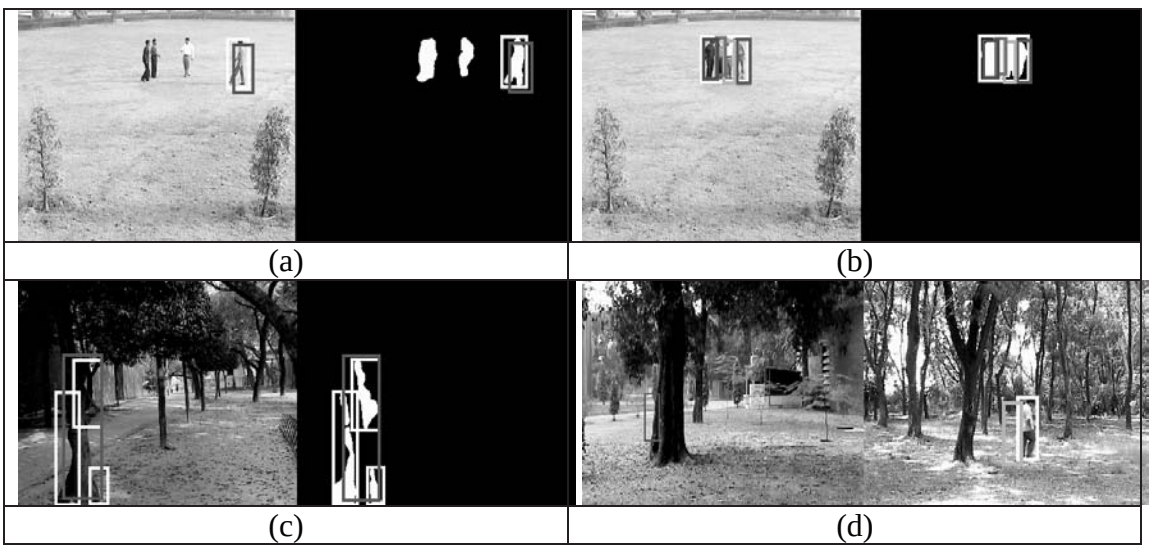
or,

$$\psi_F[i](t)=\sum_{j=1}^{m_{t-1}}\psi_{FA}[i,j](t).$$

A discussion on using these measures for identifying the occlusion cases can be found in Guha, et al. (2005).

The **j**$^{\textbf{th}}$ object in $\mathbf{S_A(t\text{-}1)}$ is **isolated** or unoccluded ***O*(I)[j,t]**, if the localization confidence is significantly high and the associated foreground blob is not overlapped with other objects. However, when the object **disappears** (***O*(D)[j,t]**) both localization and attribution confidences fall below η_A and η_F signifying very poor or no association of the object to any foreground blob. In case of **partial occlusions** (***O*(P)[j,t]**), the attribution

Figure 3. Cases of occlusions: (a) isolation: object unoccluded by other objects or parts of background; in this state, both visual characteristics and motion history of the object is updated; (b) crowding: multiple objects merge into a single blob; (c) partial occlusion: object occluded by tree is visible as three fragmented blobs; in the states (b) and (c), the object is recognized, and only motion history is updated; (d) disappear: object is not associated to a foreground blob due to either complete occlusion by a static object learned as the part of the background (here a tree) or tracking failure



confidence of one or more foreground blobs to the **jth** object remains high, although the localization confidence falls significantly. On the other hand, while in a **crowd** (***O*(C)[j,t]**), the localization confidence of the **jth** object in the crowded blob (overlapped with more than one object) remains high, although the attribution confidence of that blob to the object remains low. Thus, the four Boolean predicates for these occlusion primitives can be constructed as follows in Box 1.

To construct the current active set $\mathbf{S_A(t)}$, updates are applied to all color, shape, and trajectory of individual objects under ***O*(I)**, but only to the trajectory of objects under ***O*(P)** and ***O*(C)**. Objects under ***O*(D)** are moved from $\mathbf{S_A(t)}$ to $\mathbf{S_{lost}(t)}$. This enables the system to remain updated with object features while keeping track of them.

The **entry/reappearance** of an object is attributed to the existence of a foreground blob $\mathbf{F_i(t)}$ in the scene having no association with any object from $\mathbf{S_A(t\text{-}1)}$ and is thus detected as $O(N)_i(t) = [\Theta_F[i](t) = 0] \wedge [\psi_F[i](t) = 0]$ The features of the new blob $\mathbf{F_i(t)}$ are matched against those in $\mathbf{S_{lost}(t\text{-}1)}$ to search for the reappearance of objects. If a match is found, the object is moved from $\mathbf{S_{lost}(t\text{-}1)}$ to $\mathbf{S_A(t)}$ and a *reappearance* is noted. Otherwise, a new object is added to $\mathbf{S_A(t)}$, and the system detects an *entrance*. Similarly, an object is declared to **exit** the scene if its motion-predicted region lies outside the image region and is thus removed from the active set.

ACTIVITY DISCOVERY

Activities can be broadly classified into two categories:

- **Single-object actions** or events with a single participant, the object. Such actions have no object on which the action is being performed and correspond in natural language syntax to the intransitive verb category (e.g., "John runs").
- **Object-object interactions** or events with two or more participants, the object, as well as an object on which the action is taking place (e.g., "John rides a bike"), which corresponds to the transitive verb category in syntax.

Models of single-object behaviors are characterized by the object and some characterization of the temporal character of the action (e.g., the class of trajectories the object may take, the path taken by a vehicle in a traffic scenario, or the pose sequence exhibited by a dancer). Activities in this class include "Cars drive toward the left," "the motorcycle joins the road," "the man hides behind the tree," and so forth.

Object-object interactions exhibit several different modes. The actions may involve actual contact (e.g., riding a bike, boarding or disembarking a vehicle, grouping, etc.), or they may involve interactions at a distance (e.g., following, chasing, overtaking, etc.). In terms of image space, actual contacts are necessarily reflected in

Box 1.

$$O(I)[j,t] = \exists i[\Theta_{AF}[j,i](t) = 1] \wedge [\Theta_F[i](t) = 1] \quad (5)$$

$$O(D)[j,t] = [\Theta_A[j](t) = 0] \wedge [\psi_A[j](t) = 0] \quad (6)$$

$$O(P)[j,t] = \forall i[\psi_{FA}[i,j](t) = 1] \wedge [\Theta_F[i](t) = 1] \wedge [\psi_A[j](t) \geq 1] \quad (7)$$

$$O(C)[j,t] = \exists i[\Theta_{AF}[j,i] = 1] \wedge [\Theta_F[i](t) > 1] \quad (8)$$

O-primitive structures, but noncontact situations in the 3D world are not necessarily characterized by nonoverlap in the image space. More so, the interacting objects may be either homogeneous (e.g., "car 1 overtaking car 2") or heterogeneous (e.g., "man entering the tempo").

Both single-object actions and object-object interactions can be expressed as temporal sequences of object states (actions) or co-occurring states of interacting objects. Thus, the domain of activity analysis demands efficient statistical sequence modeling techniques for recognizing significant temporal patterns from the time-series data of action/interaction features. A number of methodologies employing hidden Markov models, time-delay neural networks, recurrent networks, and so forth, have been proposed for modeling and recognition of action/interaction sequences in a supervised learning framework. On the other hand, unsupervised learning of activity patterns have also been proposed by trajectory clustering (Johnson & Hogg, 1995) or variable length Markov model learning (Galata, Johnson & Hogg, 2001). A good overview of such techniques can be found in Buxton (2003).

Supervised activity modeling techniques are mostly task oriented and hence fail to capture the corpus of events from the time-series data provided to the system. Unsupervised data mining algorithms, on the other hand, discover the modes of spatio-temporal patterns, thereby leading to the identification of a larger class of events. The use of VLMM in the domain of activity analysis was introduced for automatic modeling of the actions in exercise sequences (Johnson, Galata & Hogg, 1998) and interactions like handshaking (Galata et al., 2001) or overtaking of vehicles (Galata, Cohn, Magee & Hogg, 2002) in a traffic scenario. These approaches propose to perform a vector quantization over the object feature and trajectory space to generate temporally indexed object-state sequences from video data. These sequences are parsed further to learn VLMMs leading to the discovery of behavioral models of varying temporal durations.

Motion (pose) primitives derived from object (state) trajectories are a necessary set of activity descriptors but are not sufficient, as they lack the power to describe the interactions involving object-region contacts in the image space. We thus augment the activity feature space with the set of occlusion primitives that form a more fundamental notion of interaction signatures. More so, we recognize that the occlusion state transition sequence forms a more significant interaction description than the occlusion state sequences. In this work, we aim to discover the interactions arising out of objects moving in complex environments and undergoing both static and dynamic occlusions with parts of background and other moving objects, respectively. In the following subsections, we discuss the methodologies adopted for sequence modeling and event primitive representations for interaction modeling.

Incremental Transition Sequence Learning

Activities of a single object and interactions among multiple objects are captured as sequences of occlusion primitives, together with object motion data. The combined atomic event primitive $\boldsymbol{\varepsilon}$ constitutes the set $\boldsymbol{E}$. Our approach to mining in this space involves the construction of an *activity tree* $\boldsymbol{T_\alpha}$ whose branches represent variable length event sequences.

An empty (first in, first out) buffer $\boldsymbol{\beta_j}$ (of length L, the maximum sequence length) and the null activity tree $\boldsymbol{T_\alpha(j)}$ (containing only the root node $\boldsymbol{\rho_j}$) are initialized at the very first appearance of every $\mathbf{j^{th}}$ object $\mathbf{A_j}$. Each node of $\boldsymbol{T_\alpha(j)}$ is a two tuple $\mathbf{T_n \equiv (\varepsilon , \pi)}$ containing the primitive $\boldsymbol{\varepsilon \in E}$ and a real number $\boldsymbol{\pi} \in (0,1]$ signifying the probability of occurrence of the path $\mathbf{\{ \rho_j, \ldots, T_n \}}$ among the set of all possible paths of the same length.

Let $\boldsymbol{\varepsilon(j,t)}$ be the event primitive observed for $\mathbf{A_j}$ at time $\mathbf{t}$. If there is a transition in this event primi-

tive (i.e., if $\boldsymbol{\varepsilon(j,t) \neq \varepsilon(j,t-1)}$), then $\boldsymbol{\varepsilon(j,t)}$ is pushed to $\boldsymbol{\beta_j}$. Let the set of **l**-length paths (originating from $\boldsymbol{\rho_j}$) of $\boldsymbol{T_\alpha(j,t)}$ be $\boldsymbol{B^{(l)}(j,t) = \{ \alpha_u^{(l)}(j,t): u = 1,\ldots b_l \}}$, where $\boldsymbol{b_l}$ is the number of **l**-length branches in the tree. More so, if the sequence $\boldsymbol{\beta_j[l\text{-}k](t)}$ ($\boldsymbol{k}$ = 1,…**l**) signifies the $\boldsymbol{b^{th}}$ path of $\boldsymbol{B^{(l)}(j,t)}$, then the probabilities $\boldsymbol{\pi^{(l)}(j,t)}$ ($\boldsymbol{u}$ = 1,… $\boldsymbol{b_l}$) of the nodes of $\boldsymbol{T_\alpha(j,t)}$ at the $\boldsymbol{l^{th}}$ depth are updated as:

$$\pi_u^{(l)}(j,t) = (1-\eta_l(t))\pi_u^{(l)}(j,t-1) + \eta_l(t)\delta(u-b) \quad (9)$$

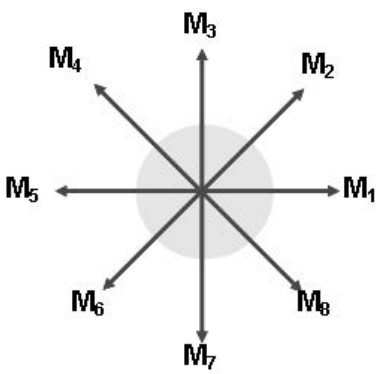
where, $\eta_l(t)$ is the rate of learning **l**-length sequences at the $\boldsymbol{t^{th}}$ instant, and $\delta(\cdot)$ is the Kronecker delta function. In the current implementation, a fixed learning rate η is employed such that $\eta_l(t) = \max(t^{-1}, \eta), \forall l$.

Occurrence of a new event primitive results in the formation of newer variable length sequences in the buffer. Thus, new nodes signifying this event primitive are added at various levels of the tree, thereby growing newer branches. Each new node is initialized with an initial probability of $\eta_l(t)$, whereas the older node probabilities in the same levels are penalized by multiplying with a factor of (1.0 - $\eta_l(t)$). This ensures the self-normalizing nature of node probability updates (as in equation 9 such that they add up to 1.0 at each depth.

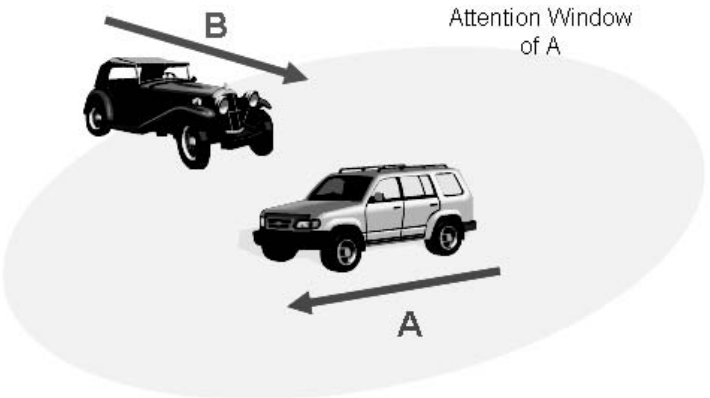
Unsupervised Interaction Learning

We construct event primitives for objects by combining their occlusion states and motion primitives. The occlusion states of *isolation* (***O*(I)**), *partial occlusion* (***O*(P)**), *crowding* (***O*(C)**), *disappearance* (***O*(D)**), *exit* (***O*(X)**), *entry* (***O*(E)**), and *entrance of new object in neighborhood* (***O*(N)**) unite to form a seven-bit occlusion-driven interaction descriptor. The direction of motion of the object is quantized to assign one of the eight motion primitives $\mathbf{M_1}$ to $\mathbf{M_8}$ signifying the compass directions of *east*, *northeast*, to *southeast* (going counterclockwise), respectively. Besides, a motion

Figure 4. Event primitive descriptors: monadic action model: One of the seven occlusion primitives and the motion primitive are combined to obtain the single-object atomic event descriptor; dyadic interaction model: Attentive focus (query object) is A; object B, if within the attention window of A, is considered to interact with A. The two co-occurring occlusion and motion primitives of A and B constitute the atomic interaction descriptor; temporally ordered sequences of these descriptors are parsed to discover meaningful activity.

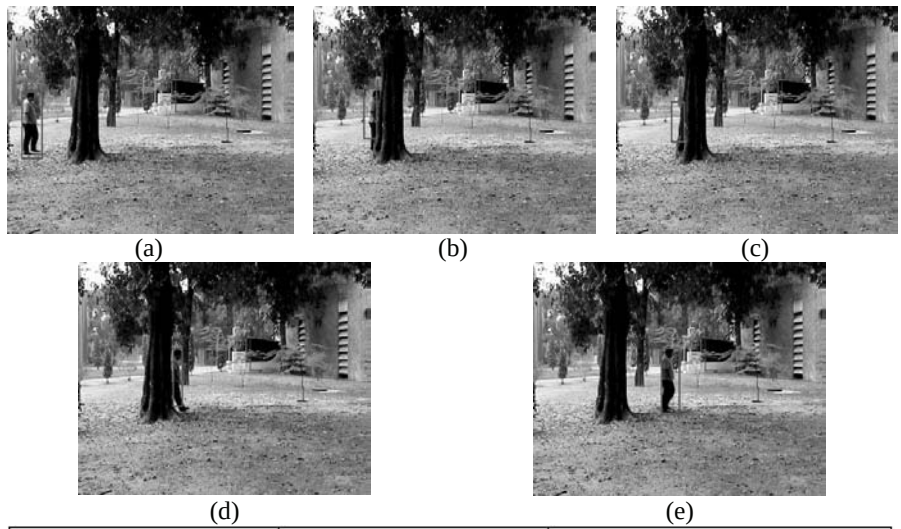


Motion Primitives for Stasis (M_0) and (M_1 – M_8) Compass Directions



Interaction Descriptors as Co-occurring Occlusion and Motion Primitives (O_A,M_A,O_B,M_B)

Figure 5. Example video sequence: man walks from left to right behind a tree; frames and object states: (a) 1-5: isolated; (b) 6: partially occluded; (c) 7-8: disappeared; (d) 9: partially occluded; (e) 10-18: isolated; (f) learning the Activity Tree; the left-most nodes are just below the root of the growing tree; results of incremental transition sequence learning are shown after frames 1, 6, 7, 9, and 18. Branches encode different variable length event sequences along with relative frequencies; thus, in column 2 (after Frame 9), the sequence { (I→P→D) , 0.89 } corresponds to the event primitive sequence (O(I),M_1)→(O(P),M_1)→(O(D),M_0) (i.e., the event sequence "coming from the left and getting hidden" occurs with relative frequency, 89% among observed three-length sequences.



(f)

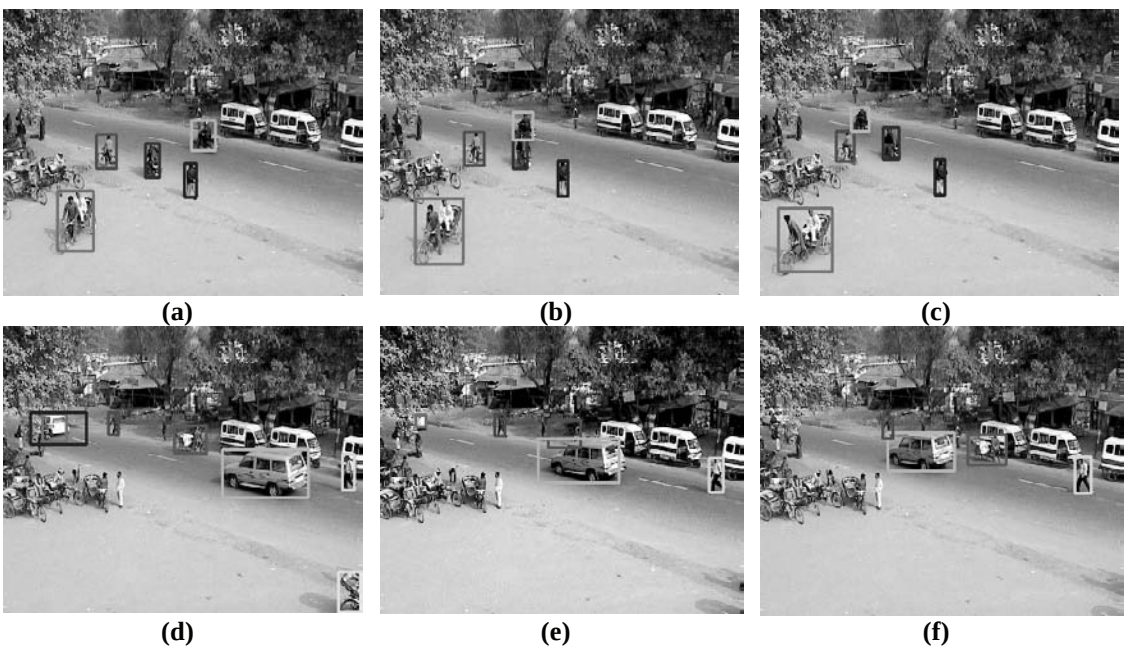
primitive $\mathbf{M_0}$ is used to signify the state of stasis of the object. The final event descriptor for a single object is formed by concatenating the occlusion and motion primitives, as shown in Figure 4.

Consider a short video sequence where a person walks behind a tree from left to right in the image space from which we sample 18 frames to illustrate the process of object-background interaction discovery. Incremental transition sequence learning is performed with a maximum depth of L = 10 and a learning rate inversely proportional to the frame number. Key frames from this sequence along with the growth of the activity tree are shown in Figure 5.

Semantic labels can be assigned to the sequences in the occlusion-primitive space to denote different activities, and subsequences may constitute subactivities. For example, consider the longest path $\mathbf{(O(I),M_1) \rightarrow (O(P),M_1) \rightarrow (O(D),M_0) \rightarrow (O(P),M_1) \rightarrow (O(I),M_1)}$ learned in the activity tree from the video that corresponds to the activity of *walking across a tree from left to right.* Subsequences of this path (i.e., $\mathbf{(O(I),M_1) \rightarrow (O(P),M_1) \rightarrow (O(D),M_0)}$ and $\mathbf{(O(D),M_0) \rightarrow (O(P),M_1) \rightarrow (O(I),M_1)}$) also correspond to the visually significant events of *going to hide from left to right* and *reappearing and moving to the right.*

We consider the object **B** to be interacting with **A** if the center of the minimum bounding box of the former lies within an attention window of the latter (Galata et al., 2002). The interaction primitives are formed by combining the co-occurring occlusion and motion states of the two interacting objects (Figure 5). Figure 6 shows the results of discovering the interaction sequences of *overtaking* and *crossing* from a traffic video.

Figure 6. Overtaking sequence: (a)-(c) A man on bike (Object A) overtaking another man on bike (Object B) generating a sequence $(O_A(I), M_4, O_B(I), M_4) \rightarrow (O_A(C), M_4, O_B(C), M_4) \rightarrow (O_A(I), M_4, O_B(I), M_4)$. *Crossing sequence: (d)-(f) an SUV (Object A) crossing a rickshaw (Object B) generating a sequence* $(O_A(I), M_4, O_B(I), M_8) \rightarrow (O_A(C), M_4, O_B(C), M_8) \rightarrow (O_A(I), M_4, O_B(I), M_8)$



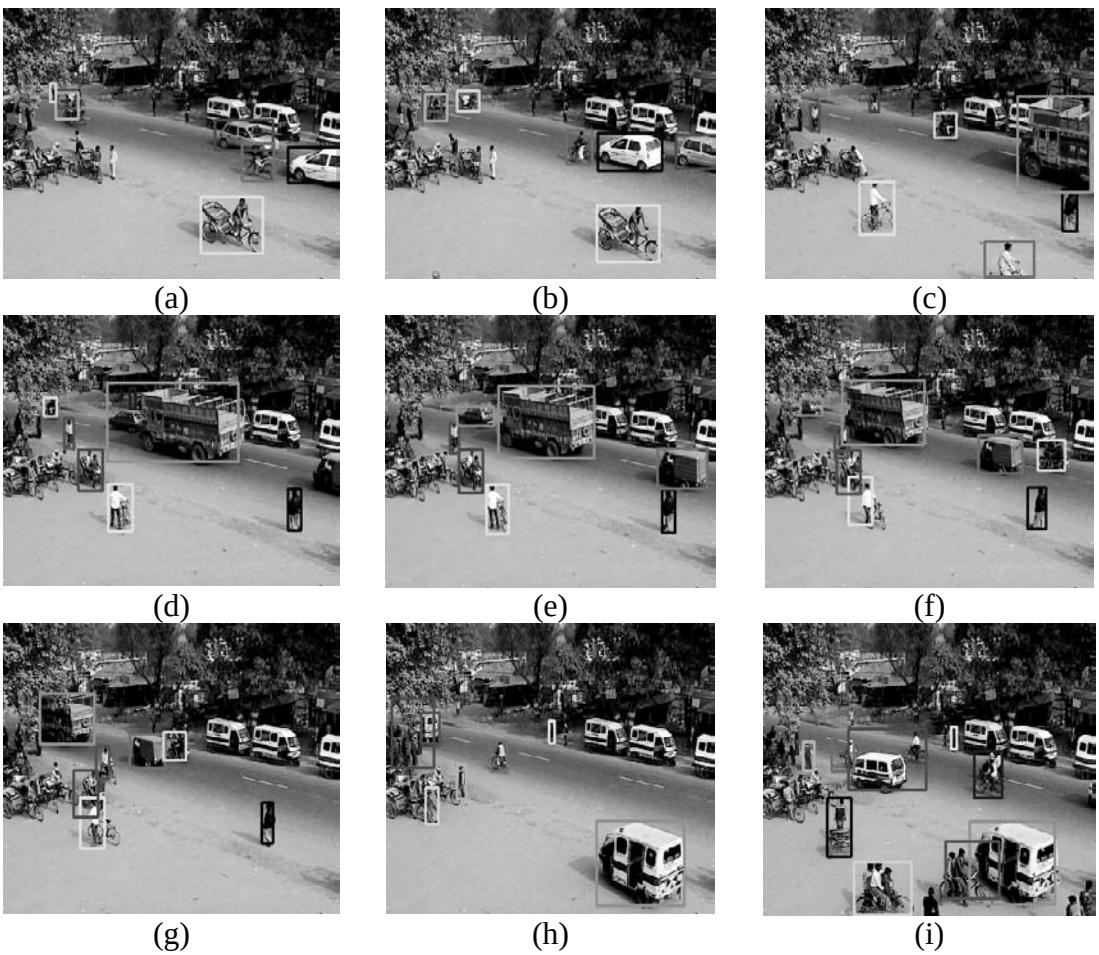
RESULTS

Experiments are performed on a traffic surveillance video of 5,000 frames consisting of a wide variety of vehicles like bikes, rickshaw, cars, heavy vehicles, and so forth, along with men and animals. The background modeling is performed by learning a pixelwise mixture of Gaussians over the RGB color space with a learning rate of $\boldsymbol{\alpha}$ = 0.01 and a diagonal covariance matrix $\boldsymbol{\Sigma_{init}}$ = {4.0}. The foreground extraction is performed with interframe motion information and selective model update with higher layer object position feedback. Comparative results of foreground extraction are shown in Figure 2.

Multiple objects in the traffic video are tracked with O-primitive identification. The tracking performance of the $\mathbf{j^{th}}$ object at the $\mathbf{t^{th}}$ instant is evaluated by the fraction of the ground-truth region of the same ($\mathbf{G_j(t)}$) overlapped with the region $\mathbf{a_j(t)}$, localized by the proposed algorithm, and is thus given by the quantity $\boldsymbol{\gamma}$($\mathbf{G_j(t)}$, $\mathbf{a_j(t)}$). Hence, if there are $\mathbf{m_g(t)}$ (number of) objects present in the ground-truth marked images at the $\mathbf{t^{th}}$ instant, then the overall performance $\boldsymbol{P}$ for a video of T frames is given by

$$p = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{m_g(t)}\sum_{j=1}^{m_g(t)}\gamma\left(G_i(t), a_j(t)\right) \qquad (10)$$

Figure 7. (a)-(i) Results of tracking multiple objects in the traffic surveillance video.

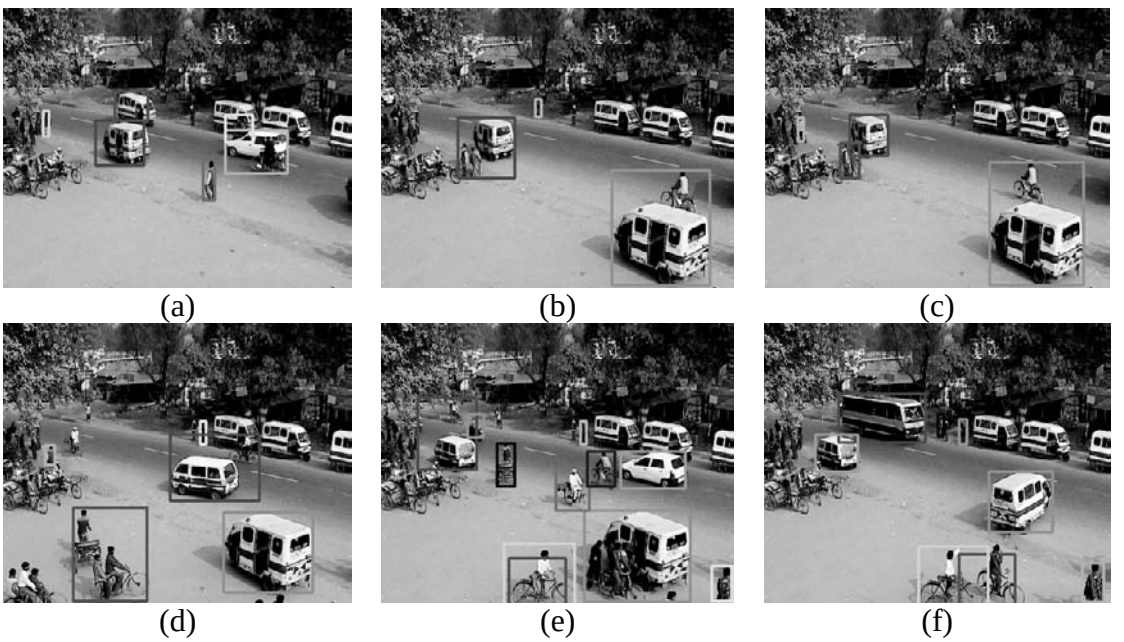


This measure of overall performance ***P*** signifies the average fraction of the actual object regions (or ground-truth regions) localized by the tracking algorithm in a certain video sequence. The overall performance varies as the thresholds η_A and η_F are changed. It is evident from equations 3 and 4 that as the thresholds η_A and η_F are increased, the detection rates of correspondences between predicted object regions and foreground blobs decrease, and thus, the rate of track loss increases. On the other hand, too low values of these thresholds would increase the number of false detections of the O-primitives. Thus, to achieve optimal performances, we have chosen $\eta_A = \eta_F = 0.6$, and an overall tracking performance of approximately 68% is observed. The results of tracking in the traffic surveillance video are shown in Figure 7.

The results of multiobject tracking are logged into a database, where each object is stored with its various appearances (learned only when isolated), image space trajectory, and occlusion state sequence for its scene presence in the surveillance video. These constitute the surveillance logs from which object information can be retrieved with simple SQL queries. We assume the availability of object recognition modules that can categorize the objects based on their appearance features.

The activities are learned with a maximum depth of L = 10 and a learning rate of $\eta(t) = \max(t^{-1}, 0.01)$ at the $\mathbf{t^{th}}$ instant. Activities are discovered for a particular query object by mining its monadic and dyadic occlusion and motion primitive sequences. We have empirically chosen an attention window of size 1.5 times the minimum bounding box of the object for all

Figure 8. Results of activity discovery: (a-c) Disembarking from vehicle (blue bounding box): (a) tempo comes to stop (frame 1440); (b) fragmentation due to people disembarking (frame 1610); (c) new objects (people) formed in neighborhood of tempo (frame 1622); (d-f) boarding on vehicle (green bounding box); (d) people approaching tempo (static), entering its attention horizon (frame 1900); (e) people crowded with tempo (frame 2070); (f) people disappear, tempo moving and is still tracked (frame 2330)



our experiments. In addition to overtaking and crossing, we have discovered the activities of ***disembarking*** from and ***boarding*** vehicles in the traffic video. The results of these interactions are shown in Figure 8.

CONCLUSION

This chapter demonstrates the extent to which unsupervised activity discovery is possible by merely constructing sequences of occlusion events along with the image plane motion. Temporal sequences of O-primitives are posited as a powerful tool for identifying multiobject interactions. The computation of occlusion is made possible by robust foreground extraction (even in the presence of gross occlusion) that enable us to track an object across lengthy image sequences, the occlusion patterns during which are a surprisingly rich indicator of the activity involved.

The generality of occlusion as a phenomenon that pervades all types of object interactions clearly makes it an important area of study. To our knowledge, this is the first work to focus on this domain. Perhaps owing to the same reason, the child learner also quickly becomes sensitive to the presence of objects that are occluded from sight, and occlusion is perhaps the key perceptual indicator for fundamental spatial notions such as containment and contact. In addition, image-plane motions are indicative of other perceptually salient features such as path, source, goal, and so forth.

In future work, we plan to explore other low-level tools available for activity recognition. With qualitative information on camera calibration, one can add detailed spatial characterizations for the motions *translate left / right / toward / away, rotate, speed-up, halt,* and so forth, which can by themselves be informative for many actions.

Event predicates are characterized by the type of activity (modeled as a fine-grained image schema), the ordered set of objects participating in it, as well as optional characteristics such as time, place, manner, and so forth. These arguments also emerge from the work as the dimensions in the feature space where the events are discovered.

In this work, we have classified objects only by their shape and motion characteristics, but possibly a more important characterization is in terms of actions in which an object participates (e.g., what objects participate in embark/disembark events?). This leads to a chicken-and-egg problem: one needs objects to recognize events, and actions to characterize objects. This will remain an important area for object discovery for many years to come.

Based on these low-level categories, one can build up to higher-level constructs based on several sources of additional information:

- **Multimodal Learning.** Given cotemporaneous linguistic descriptions and given the event and object characterization already at hand, it would be a simple enough matter to build grounded models of the head verb and its noun subcategories.
- **Camera Calibration/Ground-Plane Assumption.** By using camera calibration data and making ground-plane assumptions for the objects in a given domain, considerable detail can be added to the event characterizations.
- **Shape and Scene Priors.** While supervised object and event characterization may be extremely useful, we would like to avoid this for some time since it limits the scalability of the approach.

In addition to these aspects, it would be important to extend the work to more general situations (e.g., cameras that can move) (initially with pan-tilt motions) and for dynamic backgrounds (e.g., trees, fountains).

REFERENCES

Buxton, H. (2003). Learning and understanding dynamic scene activity: A review. *Image and Vision Computing, 21*(1), 125–136.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transaction on Pattern Analysis Machine Intelligence, 25*(5), 564–575.

Galata, A., Cohn, A.G., Magee, D., & Hogg, D. (2002). *Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models.* Proceedings of the European Conference on Artificial Intelligence, Lyon, France, 741–745.

Galata, A., Johnson, N., & Hogg, D. (2001). Learning variable-length Markov models of behavior. *Computer Vision and Image Understanding, 81*(3), 398–413.

Gavrilla, D. (1999). Visual analysis of human movement: A survey. *Computer Vision and Image understanding, 73*(1), 82–98.

Granlund, G. (2003). *Organization of architectures for cognitive vision systems.* Proceedings of the Workshop on Cognitive Vision, Schloss Dagstuhl, Germany.

Guha, P., Mukerjee, A., & Venkatesh, K.S. (2005). *Efficient occlusion handling for multiple agent tracking with surveillance event primitives.* Proceedings of the Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, 49–56.

Haritaoglu, I., Harwood, D., & Davis, L. (2000). W4: Real time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 809–830.

Johnson, N., Galata, A., & Hogg, D. (1998). *The acquisition and use of interaction behavior models.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, 866–871.

Johnson, N., & Hogg, D. (1995). *Learning the distribution of object trajectories for event recognition.* Proceedings of the 6th British Conference on Machine vision, Birmingham, 2, 583–592.

Mandler, J. (1992). How to build a baby: II. Conceptual primitives. *Psychological Review, 99*(4), 587–604.

Proesmans, M., Gool, L.V., Pauwels, E., & Osterlinck, A. (1994). *Determination of optical flow and its discontinuities using non-linear diffusion*. Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden, 2, 295–304.

Stauffer, C., & Grimson, W. (1999). *Adaptive background mixture models for real-time tracking.* Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ft. Collins, Colorado, 2, 246–252.

Zhao, T., & Nevatia, R. (2004). *Tracking multiple humans in crowded environments.* Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, 2, 406–413.

Zivkovic, Z. (2004). *Improved adaptive Gaussian mixture model for background subtraction*. Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 2, 28–31.

Chapter X
Human Detection in Static Images

Hui-Xing Jia
Tsinghua University - Beijing, China

Yu-Jin Zhang
Tsinghua University - Beijing, China

ABSTRACT

Human detection is the first step for a number of applications such as smart video surveillance, driving assistance systems, and intelligent digital content management. It's a challenging problem due to the variance of illumination, color, scale, pose, and so forth. This chapter reviews various aspects of human detection in static images and focuses on learning-based methods that build classifiers using training samples. There are usually three modules for these methods: feature extraction, classifier design, and merge of overlapping detections. The chapter reviews most existing methods for each module and analyzes their respective pros and cons. The contribution includes two aspects: first, the performance of existing feature sets on human detection are compared; second, a fast human detection system based on histogram of oriented gradients features and cascaded AdaBoost classifier is proposed. This chapter should be useful for both algorithm researchers and system designers in the computer vision and pattern recognition community.

INTRODUCTION

Human detection is the first step for a number of applications such as smart video surveillance, driving assistance system, and intelligent digital content management. Surveillance cameras are already prevalent in a lot of areas such as banks, department stores, airports, and parking lots. The aim of smart video surveillance is to analyze the video data real-time and alert security officers when predefined events such as burglary happen. Human detection, tracking, and activity recognition are key techniques for the application. Pedestrian safety has become a

worldwide problem with the popularity of vehicles. Accurately detecting pedestrians using a camera and warning the drivers before a crash happens will greatly increase the safety of pedestrians. With the popularity of digital cameras, personal photos have increased exponentially. Manually searching and locating these photos are very tedious. Intelligent digital content management software that automatically adds tags to images to facilitate search is thus an important research area. Most of the images taken are of human, so human detection will form an integral part of such tools.

Human detection is a challenging problem due to a lot of factors (Dalal, 2006). First, the within-class variations are very large. A robust human detector must deal with the change of viewpoint, pose, clothing, illumination, and so forth. Second, background clutter is common and varies from image to image. The detector must be capable of distinguishing the object from complex background regions. Third, partial occlusions create further difficulties because only part of the object is visible for processing. The first two difficulties present conflicting challenges, which must be tackled simultaneously. A detector that is very specific to one type of human instance will give less false detections on background regions, while an overly general detector can handle large intraclass variations but will generate a lot of false detections on background regions.

Given a single image, an ideal human detector should be able to identify and locate all the present humans regardless of their position, scale, or pose. However, because of the articulations of the human body, it will be a very difficult problem to detect humans of all poses and viewpoints; most existing systems only deal with stand-up humans and use learning-based methods. Within learning-based methods, the processing is done as follows: an input image is scanned at all possible locations and scales by a subwindow. Human detection is posed as classifying the pattern in the subwindow as either human or nonhuman. The human/nonhuman classifier is learned from training examples using a machine learning technique. A human detector usually includes three modules: feature extraction, classification, and fusion of multiple detections. The feature extraction module needs to extract the most relevant features for classification; the classifier classifies the feature vector into human or nonhuman. When scanning the image using a human classifier, there will be many positive responses around the object. How to merge these responses to achieve the final position is also an important topic.

The rest of this chapter is organized as follows: section 2 reviews previous work on human detection; sections 3 through 5 introduce various methods for feature extraction, classifier design, and merge of overlapping detections; section 6 proposes the human detection system based on histogram of oriented gradients (HOG) features and cascaded AdaBoost classifier; section 7 compares the performance of various features and classifiers. Finally, section 8 presents some issues for further research.

BACKGROUND

Previous methods on human detection differ in three perspectives: first, they may use different features such as edge features, Haar-like features, and gradient orientation features; second, they may use different classifiers such as neutral network (NN), support vector machine (SVM), and cascaded AdaBoost; third, they may treat the image region as a whole or detect each part first and then combine them by these parts' geometric configurations. In this chapter, these methods are classified into three categories based on the features they use.

Edge features have been used in earlier works. Gavrila and Philomin (1999) use edge template as the feature and compare edge images to a template dataset using the chamfer distance. This method has been experimented on with a DaimlerChrysler

vehicle (Gavrila, Giebel & Munder, 2004). Broggi, Bertozzi, Fascioli, and Sechi (2000) built a template of head and shoulder for pedestrian detection. Edge feature is affected by background clutter greatly and is not very robust.

Haar-like features have been used successfully in face detection and also adopted by a lot of researchers for human detection. Oren, Papageorgiou, Sinha, Osuna, and Poggio (1997) combine over-complete Haar wavelets and SVM classifiers to detect pedestrians. Mohan, Papageorgiou, and Poggio (2001) extend Oren et al.'s work using a cascade of SVM to detect human components first and then vote for a human. Viola, Jones, and Snow (2003) extend the Haar features to capture spatial-temporal information for moving-human detection in video streams under a surveillance system but adopt the cascaded AdaBoost classifier. Haar-like features can be evaluated very quickly using the integral images together with the cascade classifier structure, making a real-time detector possible.

Recently, gradient orientation features such as scale invariance feature transform (SIFT) (Lowe, 2004) descriptor and HOG descriptor (Dalal & Triggs, 2005) have attracted more attention. In order to detect pedestrians on a moving vehicle, Shashua, Gdalyahu, and Hayon (2004) manually divide the human into 13 regions and compute SIFT-like features of each region, then combine these features and train a classifier using AdaBoost. Dalal and Triggs (2005) propose HOG features as human representation, which achieves amazingly good results on human detection combined with SVM classifiers. Later, they extend their approach to detect humans in video streams using oriented histograms of flow and appearance (Dalal, Triggs & Schimid, 2006). Zhu, Avidan, Yeh, and Cheng (2006) integrate the cascaded AdaBoost approach with HOG features to speed up Dalal's method greatly, using linear SVM as a weak classifier. There are also other systems that use gradient orientation features but adopt a parts-based approach that aims at dealing with the great variability in appearance due to body articulation or occlusion. For example, Mikolajczyk, Schmid, and Zisserman (2004) represent human parts as co-occurrences of local orientation features. Their system proceeds by detecting features and then parts, and eventually humans are detected based on assemblies of these parts.

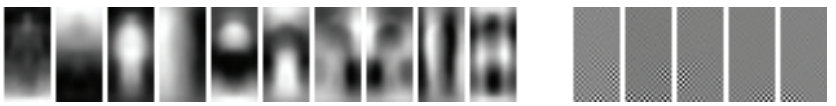
FEATURE EXTRACTION

The ultimate goal of feature extraction for object detection is to find one representation that yields high interclass variability and at the same time achieves low intraclass variability. The feature extraction module needs to extract the most relevant features for classification and provides invariance to changes in illumination, viewpoints, color, and so forth. This section focuses on the Haar-like features and HOG features that have been widely used for human detection, while using the principal component analysis (PCA) coefficients as the baseline features for comparison.

PCA Coefficients

PCA is the most famous feature extraction method that has been used widely in statistical pattern recognition area. In PCA, a linear subspace that best explains the variations of origin data points is constructed. The eigenvectors and eigenvalues of the covariance matrix are computed. The eigenvectors with the largest eigenvalues are identified as the most expressive features, while those with small eigenvalues are assumed as noise and are cut off accordingly, as in Figure 1. The kept eigenvectors form the linear subspace, and the PCA coefficients for each input feature vector are computed by projecting the input feature vector to this subspace. PCA can decrease the dimension greatly while keeping most of the variance. However, PCA coefficients are not guaranteed to be good for classification (Forsyth & Ponce, 2003).

Figure 1. An illustrating example of principle components obtained on the DaimlerChrysler pedestrian dataset introduced in the experiment section, sorted in descending order of corresponding eigenvalues (first 10 and last 5)



Haar-Like Features

The Haar-like features are first proposed by Oren, et al. (1997) as overcomplete Haar wavelets for pedestrian detection. Later, Papageorgiou & Poggio (2000) make a thorough study of the overcomplete Haar wavelets for the detection of face, car, and pedestrian. Following the idea of overcomplete Haar wavelets, Viola and Jones (2001) propose the Haar-like rectangle features and the fast evaluation method for face detection. Later, Lienhart and Maydt (2002) added rotated rectangle features to the feature set, which is called extended Haar-like features. The Haar-like features can be computed very quickly using auxiliary integral images, together with the cascaded AdaBoost algorithm, forming a very powerful object detection framework. The rest of this section introduces the definition and evaluation of overcomplete Haar wavelets and the Haar-like features.

Haar wavelets can encode the difference in average intensity between local regions along different orientations in a multiscale framework, enabling an effective representation for human class. For a given pattern, the Haar wavelet transform computes the responses of the wavelet filters over the image. Each of the three oriented wavelets (vertical, horizontal, and diagonal, as in Figure 2) is computed at several different scales, allowing the system to represent coarse scale features all the way down to fine scale features. In the traditional wavelet transform, the wavelets do not overlap; they are shifted by the size of the support of the wavelet in *x* and *y*. To achieve better spatial resolution and a richer set of features, the overcomplete transform shifts by 1/4 of the size of the support of each wavelet, yielding an overcomplete dictionary of wavelet features. The resulting high dimensional feature vectors are used as training data for classification. Certain prior knowledge is embedded in the choice of the wavelets. First, the absolute values of the magnitudes of the wavelets are used. Second, for color images, the wavelet transform in each of the three color channels is computed, and then the largest one in magnitude is chosen. Third, only the two medium scales of wavelets are selected. Wavelets of the finest scale are assumed to represent noise, and the very coarse scale wavelets that have support as large as the object itself are assumed to encode no information. For example, for image size 18x36 in the experimental section, we selected wavelets of 4x4 and 8x8, from which we obtained 15x33 and 6x15 features, respectively, for each orientation; hence, a total of 1,755 features. Experiments demonstrate that the overcomplete Haar wavelet is an effective representation for human classification (see Figure 3).

Inspired by the overcomplete Haar wavelets, Viola and Jones (2001) propose the Haar-like features for face detection. The basic four feature types are shown in Figure 4. Such a block feature is located in a subregion of a subwindow and varies in shape (aspect ratio), size, and location inside the subwindow. For a small subwindow, there can be thousands of such features for varying shapes, sizes, and locations. These Haar-like features are very useful for object detection for two reasons: first, they form a feature pool that is effective for

Figure 2. Haar wavelets of three different orientations: vertical, horizontal, and diagonal

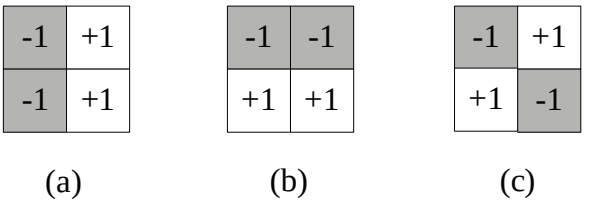


Figure 3. Ensemble average values of the wavelet coefficients on the DaimlerChrysler pedestrian dataset coded using grey level. Coefficients whose values are above the template average are lighter; those below the average are darker. (a-c) vertical, horizontal, and diagonal coefficients of scale 4x4; (d-f) vertical, horizontal, and diagonal coefficients of scale 8x8.

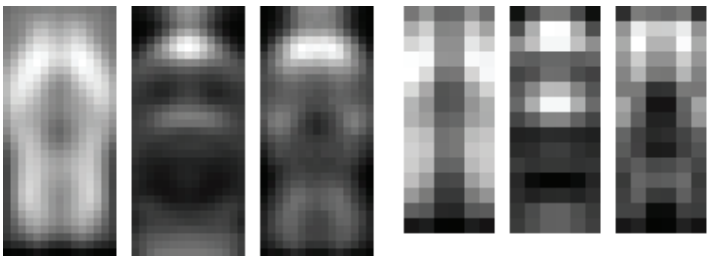
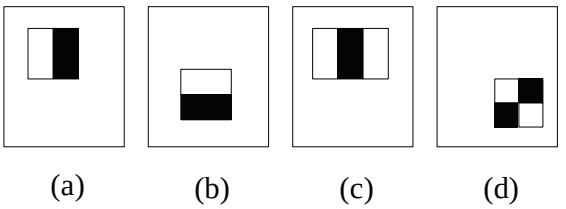


Figure 4. Haar-like features shown relative to the enclosing detection subwindow; the sum of the pixels that lie within the white rectangles are subtracted from the sum of pixels in the black rectangles (Viola & Jones, 2001)



classification and can be selected by AdaBoost; second, they can be evaluated efficiently using integral images (Viola & Jones, 2001). The integral image $ii(x, y)$ at location (x, y) contains the sum of the previous pixels and to the left of (x, y) inclusive, defined as:

$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x, y) \quad (1)$$

The image can be computed in one pass over the original image using the following pair of recurrences:

$$s(x, y) = s(x, y-1) + i(x, y)$$
$$ii(x, y) = ii(x-1, y) + s(x, y) \quad (2)$$

where $s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$ and $ii(-1, y) = 0$. Using the integral image, any rectangular sum can be computed in four array references (see Box 1).

Histogram of Oriented Gradients

Image edges and gradient orientation features have also been used for object detection for a long time, as introduced in the background

Box 1.

$$\sum_{x\le x'<x+w,\, y\le y'<y+h} i(x', y') = ii(x+w-1, y+h-1) + ii(x-1, y-1) - ii(x+w-1, y-1) - ii(x-1, y+h-1) \quad (3)$$

section. Dalal and Triggs (2005) and Dalal, et al. (2006) do a thorough study of representations using gradient orientation features and propose histogram of oriented gradients features, which have demonstrated excellent results for human detection both on static images (Dalal & Triggs, 2005) and video streams (Dalal et al., 2006).

The method is based on evaluating a dense grid of well-normalized local histograms of image gradient orientations over the image windows. The hypothesis is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradient directions, even without precise knowledge of the corresponding gradient or edge positions. The descriptors can be computed as follows (Dalal, 2006):

1. **Normalize gamma and color.** The aim of gamma normalization is to reduce the influence of illumination effects. In practice, either computing the square root or the log of each color channel can be used.
2. **Compute gradients.** This stage computes the first order image gradients, which can capture contour, silhouette, and some texture information, while providing resistance to illumination variations. Several derivative masks (uncentered [-1 1], centered [-1 0 1], and cubic-corrected [1 -8 0 8 -1]) can be used, and experiments show that the simplest 1-D mask [-1 0 1] performs best.
3. **Accumulate weighted votes for gradient orientation over spatial cells.** The third stage aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in poses or appearance. The image window is divided into small spatial regions called "cells." For each cell, a local 1-D histogram of gradient or edge orientations over all the pixels in the cell is accumulated. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote the orientation histogram. To reduce aliasing, votes are interpolated bilinearly between neighboring bin centers in both orientation and position. Experiments show that the "unsigned" gradient performs better than "signed" gradient for human detection.
4. **Normalize contrast within overlapping blocks of cells.** Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram "energy" over local groups of cells called "blocks." The result is used to normalize each cell in the block. Typically, each individual cell is shared among several blocks, but its normalizations are block dependent and thus different. The cell thus appears several times in the final output vector with different normalizations. This may seem redundant, but experiments show that this strategy improves the performance. The normalized block descriptors are referred to as Histogram of Oriented Gradient (HOG) descriptors.
5. **Collect HOGs for all blocks over detection window.** The final step collects the HOG descriptors from all blocks into a combined feature vector for use in the window clas-

sifier. For example, if the template size is 64x128, each cell size being 8x8 with nine bins, four cells in one block, the stride step of adjacent blocks being one cell in both horizontal and vertical directions, there are 7x15=105 blocks in total, and the total feature number is 105x4x9 = 3780.

The HOG representation has several advantages. The use of orientation histograms over image gradients allows HOG to capture local contour information (i.e., the edge or gradient structure), which is very characteristic of local shape. In conjunction with the spatial quantization into cells, it allows them to capture the most relevant information with controllable precision and invariance (e.g., by changing the number of bins and the cell size). Translations and rotations make little difference so long as they are much smaller than the local spatial or orientation bin size. Gamma normalization and local contrast normalization contribute another key component: illumination invariance. The use of overlapping of blocks provides alternative normalizations so the classifier can choose the most relevant one. These steps ensure that as little information as possible is lost during the encoding process.

CLASSIFICATION METHODS

This section focuses on discriminative methods that build a decision boundary directly from the input samples instead of density estimation methods. Machine learning techniques such as support vector machine (Vapnik, 2000) and AdaBoost (Freund & Schapire, 1997) have become the most popular discriminative methods for object recognition owing to their superior performance and relative ease of use. This section first introduces SVM briefly and then introduces a detection framework based on AdaBoost and cascaded classifier structure.

Support Vector Machine

SVM is a technique to train classifiers that is well-founded in statistical learning theory (Vapnik, 2000). SVM minimizes a bound on the empirical error and the complexity of the classifier at the same time. The concept of SVM is formalized in the theory of uniform convergence in probability:

$$R(\alpha) \le R_{emp}(\alpha) + \Phi\left(\frac{h}{l}, \frac{-\log(\eta)}{l}\right) \qquad (4)$$

with probability 1-η. Here, $R(\alpha)$ is the expected risk; $R_{emp}(\alpha)$ is the empirical risk; l is the number of training examples; h is the Vapnik-Chernovenkis (VC) dimension of the classifier that is being used; and $\Phi(\cdot)$ is the VC confidence of the classifier. Intuitively, what this means is that the uniform deviation between the expected risk and empirical risk decreases with larger amounts of training data l and increases with the VC dimension h.

The separating boundary is in general of the form:

$$f(x) = \theta\left(\sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\right) \qquad (5)$$

where l is the number of training data points $(\mathbf{x}_i, y_i)$, (y_i being the label ±1 of training point $\mathbf{x}_i$); α_i are non-negative parameters learned from the data; and $K(\cdot,\cdot)$ is a kernel that defines a dot product between projections of the two arguments in some feature space where a separating hyperplane is then found. For example, when $K(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$ is the kernel, the separating surface is a hyperplane in the space $\mathbf{x}$ (input space). Other kernels include the Gaussian radial basis function $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2\right)$ and the polynomial function $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^n$. In general, any positive definite function can be used as the kernel. The main feature of SVM is that it finds among all possible separating surfaces of the form Equation (5), the one that maximizes the distance

between the two classes of points (as measured in the feature space defined by K). The support vectors are the nearest points to the separating boundary and are the only ones (typically a small fraction of the training data) for which α_i in Equation (5) is positive.

SVM has been widely used for object recognition for the past decade. Papageorgiou and Poggio (2000) build a generic object detection system based on Haar wavelets and SVM classifier, which demonstrate excellent results in pedestrian, car, and face detections. Mohan, et al. (2001) create a two-stage cascade of SVM classifiers. The first stage creates part (head, left arm, etc.) detectors from Haar wavelets. The second stage combines the part detections to obtain the final object detector. Dalal and Triggs (2005) and Dalal, et al. (2006) use HOG features and SVM, build state-of-the-art human detection systems both in static images and videos. The advantage of SVM is that it can build a classifier that generalizes well. The disadvantage of SVM is that for classification problems in object detection community, there are always too many support vectors, making the detection speed very slow.

Cascaded AdaBoost

AdaBoost is a machine learning technique to get a strong classifier by boosting an ensemble of weak classifiers. For AdaBoost learning, a complex nonlinear strong classifier $H_T(x)$ is constructed as a linear combination of T simpler, easily constructible weak classifiers in the following form:

$$H_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad (6)$$

where x is a pattern to be classified; $h_t(x) \in \{0,1\}$ are the T weak classifiers; $\alpha_t \geq 0$ are the combining coefficients. The aim of AdaBoost is to learn a sequence of best weak classifiers $h_t(x)$ and the best combining weights α_t. The simplest type of weak classifiers is a "stump," which is a single-node decision tree. A "stump" weak classifier $h_j(x)$ consists of a feature f_j, a threshold θ_j and a polarity p_j indicating the direction of the inequitably sign:

$$h_j(x) = \begin{cases} 1 & if \ \ p_j f_j(x) < p_j \theta_j \\ 0 & otherwise \end{cases} \qquad (7)$$

The weak learning algorithm is designed to select the feature that best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified. So AdaBoost can select the feature and train the strong classifier at the same time. The flow of the algorithm is as follows (Viola & Jones, 2001):

- Given example images $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ where y_i = 0,1 for negative and positive examples, respectively.
- Initialize weights $\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for y_i = 0,1 respectively, where m and l are the number of negatives and positives respectively.
- For t = 1,...,T :
  1. Normalize the weights $\omega_{t,i} \leftarrow \omega_{t,i} \Big/ \sum_{j=1}^{N} \omega_{t,j}$ so that ω_t is a probability distribution.
  2. For each feature j train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to ω_t, $\varepsilon_j = \sum_i \omega_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier h_t with the lowest error ε_t.
  4. Update the weights: $\omega_{t+1,i} \leftarrow \omega_{t,i} \beta_t^{1-e_i}$, where e_i = 0 if example x_i is classified correctly, e_i = 1 otherwise, and $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

- The final strong classifier is:

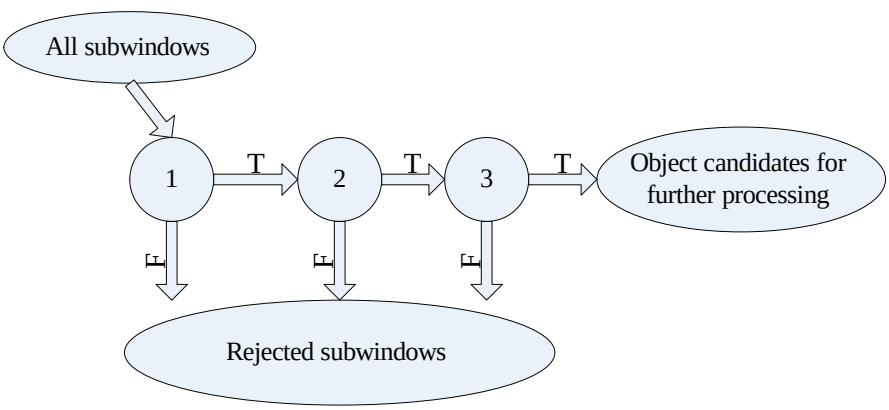
$$h(x)=\begin{cases}1 & \sum_{t=1}^{T}\alpha_t h_t(x)\geq\frac{1}{2}\sum_{t=1}^{T}\alpha_t \\ 0 & otherwise\end{cases}$$

where $\alpha_t=\log\frac{1}{\beta_t}$

AdaBoost can select features and train a strong classifier at the same time. However, a single strong classifier trained by AdaBoost is not efficient for object detection task. For the detection problem, most of the subwindows are negative, which requires the classifier to have an extremely low false positive rate; for example, 10^{-6}. A single strong classifier will consist of too many features, which makes the detection speed very slow. Since most of the negative subwindows can be determined with a very simple classifier, Viola and Jones (2001) propose the cascade classifier structure to speed up the detection process. The cascade classifier structure, together with the AdaBoost algorithm, forms a general object detection framework referred to as cascaded AdaBoost.

The cascaded AdaBoost classifier consists of a cascade of strong classifiers, as in Figure 5. Simpler classifiers are used to reject the majority of subwindows before more complex classifiers are called upon to achieve low false positive rates. A positive result from the first classifier triggers the evaluation of a second classifier. A positive result from the second classifier triggers a third classifier, and so forth. A negative outcome at any point leads to the immediate rejection of the subwindow. Since most of the subwindows are simple negative examples, they can be rejected at very early stages of the cascade. So this strategy can increase the detection speed greatly. During training, the number of weak classifiers of each stage and output threshold can be adjusted to meet a predefined hit rate *h* and false alarm rate *f*. If there are *m* stages in the cascade classifier, the final detection rate is h^m, and false alarm rate is f^m. For example, if $h = 0.995$, $f = 0.5$, $m = 20$, the final detection rate is about 0.99, and false alarm rate is about 10^{-6}.

Figure 5. Schematic depiction of the detection cascade; a series of classifiers are applied to every subwindow; the initial classifier eliminates a large number of negative examples with very little processing; subsequent layers eliminate additional negatives but require additional computation; after several stages of processing, the number of subwindows has been reduced radically; further processing can take any form such as additional stages of the cascade or an alternative detection system such as SVM (Viola & Jones, 2001)



FUSION OF MULTIPLE DETECTIONS

There are usually multiple overlapping detections for each object in the image, and these must be merged. Rowley (1999) proposes a heuristic method for fusing overlapping detections. The number of detections within a specified neighborhood is computed, and if it is greater than a threshold, the centroid of these detections is taken as the location of the detection result. Centroids are computed in 3-D position and scale space. The number of detections gives the detection score. Viola and Jones (2001) propose a simpler method. The set of detections is partitioned into disjoint subsets, and each partition corresponds to a single final detection. Two detections are taken to be in the same subset if their bounding regions overlap a threshold. The final detection and location region is the average of all the detections in the set. Dalal (2006) proposes a generalized solution to the fusion of multiple overlapping detections. The problem is posed as a kernel-density estimation problem. The solution is given by locating modes of the density estimate using a mean shift-based mode detection procedure (Comaniciu & Meer, 2002). Experiments in Dalal (2006) show that this method is superior to previous heuristic methods.

FAST HUMAN DETECTION BY BOOSTING HOG FEATURES

In this section, we present a novel real-time human detection system based on Viola's cascaded AdaBoost framework and HOG features. Each bin of the histogram is treated as a feature and used as the basic building element of the cascade classifier. The system keeps both the discriminative power of HOG features for human detection and the real-time property of the cascaded AdaBoost framework.

The Framework

The architecture of the system is the same as the structure shown in Figure 5. The stage classifier consists of an ensemble of classification and regression tree (CART) as weak classifiers combined by AdaBoost. During training, the threshold of each stage classifier is adjusted, and the number of weak classifiers is increased until the hit rate and false alarm rate meet predefined values. The CART classifier is trained aggressively (i.e., the leaf of the tree that decreases the error most will be split, and corresponding feature and threshold will be saved as the parameters of the node). The node number of each CART can be used as a meta-variable to control the complex of the whole system.

The HOG Feature Pool

From the results in Dalal and Triggs (2005) and Dalal, et al. (2006), we know that the HOG feature is more suitable for human detection than PCA coefficients and Haar-like features. By treating each bin of origin HOG features as a feature, we create a new feature pool for human detection used by AdaBoost algorithm and cascade training. With the help of an array of integral images, each feature in our feature pool can be computed at any position and any scale in constant time; only eight look-ups are needed, which enables the real time properties of our human detection system.

Each feature is defined by its cell position $C\left(x_c, y_c, w_c, h_c\right)$, the parent block position $B\left(x_b, y_b, w_b, h_b\right)$, and the orientation bin number k, so each feature f is denoted by $f(C, B, k)$. Let $G(x,y)$ and $\theta\left(x, y\right)$ be the strength and orientation of the gradient at point (x,y). We divide the orientation range $[0, \pi]$ into K bins and denote the value of k_{th} bin to be:

$$\psi_k\left(x, y\right) = \begin{cases} G\left(x, y\right) & if\ \theta\left(x, y\right) \in bin_k \\ 0 & otherwise \end{cases} \quad (8)$$

Then the feature value is defined as:

$$f(C,B,k)=\frac{\sum_{(x,y)\in C}\psi_k(x,y)+\varepsilon}{\sum_{(x,y)\in B}G(x,y)+\varepsilon} \quad (9)$$

All the features can be computed very quickly and in constant time by eight look-ups with the help of $K + 1$ auxiliary integral images:

$$IG_k(x,y)=\sum_{x'\le x,y'\le y}\psi_k(x,y),\quad k=1,\ldots,K$$
$$IG(x,y)=\sum_{x'\le x,y'\le y}G(x,y) \quad (10)$$

Then what follows is displayed in Box 2. Each feature $f(C, B, k)$ can be evaluated in eight look-ups.

If we do not put any constraint on the relative position and size of the cell and the block, the feature number will be too large. Inspired by the extended Haar-like feature definition of Linehart and Maydt (2002), we only consider the relative position of the cell and the block shown in Figure 6. The black rectangle represents the position of the cell, while the whole large rectangle denotes the position of the block. The ratio between the width and the height of the block is 1:2, 1:1, and 2:1. For a predefined training sample size, the feature template can be moved at a predefined stride step and enlarged at a scale step, forming a feature pool learned by AdaBoost. We should note here that this feature pool is much richer than Dalal's HOG feature, which only contains the templates 2(a-d) in Figure 6 at a fixed scale. Our feature pool contains many more templates and can change the scale and position freely.

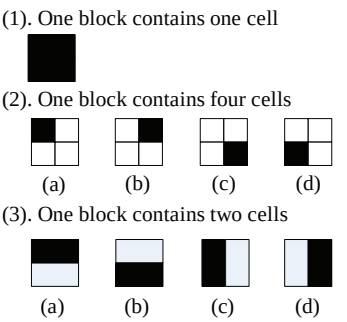
EXPERIMENTS

There are striking differences in the performance reported in the literature. The variations come

Box 2.

$$\begin{aligned}\sum_{(x,y)\in C}\psi_k(x,y) &= IG_k(x_c-1,y_c-1)+IG_k(x_c+w_c-1,y_c+h_c-1)\\ &\quad -IG_k(x_c-1,y_c+h_c-1)-IG_k(x_c+w_c-1,y_c-1)\\ \sum_{(x,y)\in B}\theta(x,y) &= IG(x_b-1,y_b-1)+IG(x_b+w_b-1,y_b+h_b-1)\\ &\quad -IG(x_b-1,y_b+h_b-1)-IG(x_b+w_b-1,y_b-1)\end{aligned} \quad (11)$$

Figure 6. HOG feature templates. The whole rectangle denotes the block, and the black rectangle denotes the relative position of the cell; the orientation histogram in the cell is normalized by the gradient strength in the block.



from various datasets and test criteria they use. In this section, we make performance evaluation on human classification by various combinations of different features and classifiers using the DaimlerChrysler pedestrian database and test criteria reported in Munder and Gavrila (2006). Besides, we also compare our human detection system with Viola and Jones' (2001) and Dalal and Trigg's (2005) system.

Figure 7 and Table 1 show the content of the DaimlerChrysler pedestrian dataset. Pedestrian examples are extracted manually from video images recorded at various daytimes and locations with no particular constraints on pedestrian pose or clothing, except that they are standing in an upright position and are fully visible. Pedestrian images are mirrored, and the bounding boxes are shifted randomly by a few pixels in horizontal and vertical directions; six pedestrian examples are thus obtained from each label. Nonpedestrian examples are generated by extracting representative patterns from video images known not to contain any pedestrian. The databases are split into five fully disjointed sets, three for training and two for testing, which allows for a variation of training and test sets during the experiments. Examples recorded at the same time and location are kept within the same set so that, for example, a pedestrian captured in a sequence of images does not show up in multiple datasets. This ensures truly independent training and test sets, but also implies that examples within a single dataset are not independent.

Classification performance is evaluated by ROC curves, which quantify the trade-off between detection rate (the percentage of positive examples correctly classified) and the false positive rate (the percentage of negative examples incorrectly classified). In order to compare the performance of two classifiers, a confidence interval is needed to decide whether performance differences are significant or represent noise. For each combination of the feature and classification method, three classifiers are trained, each by selecting one out

Figure 7. Pedestrian and nonpedestrian samples from the DaimlerChrysler benchmark dataset (Munder & Gavrila, 2006)

Table 1. DaimlerChrysler pedestrian dataset (Munder & Gavrila, 2006)

	#Datasets	Pedestrian Labels Per Set	Pedestrian Examples Per Set	Nonpedestrian Examples Per Set	Additional Nonpedestrian Images
Training Sets	3	800	4800	5000	1200
Test Sets	2	800	4800	5000	0

"Pedestrian Labels" denotes the number of pedestrians manually labeled, whereas "Pedestrian Examples" denotes the number of pedestrian examples in each dataset derived from the pedestrian labels by mirroring and shifting.

of the three training sets. Testing the three classifiers on the two test sets yields six different ROC curves (i.e., six different detection rates for each possible number of false positives). When taken as six independent tests that follow a normal distribution, a confidence interval of the true mean detection rate is given by the distribution (Munder & Gavrila, 2006) as:

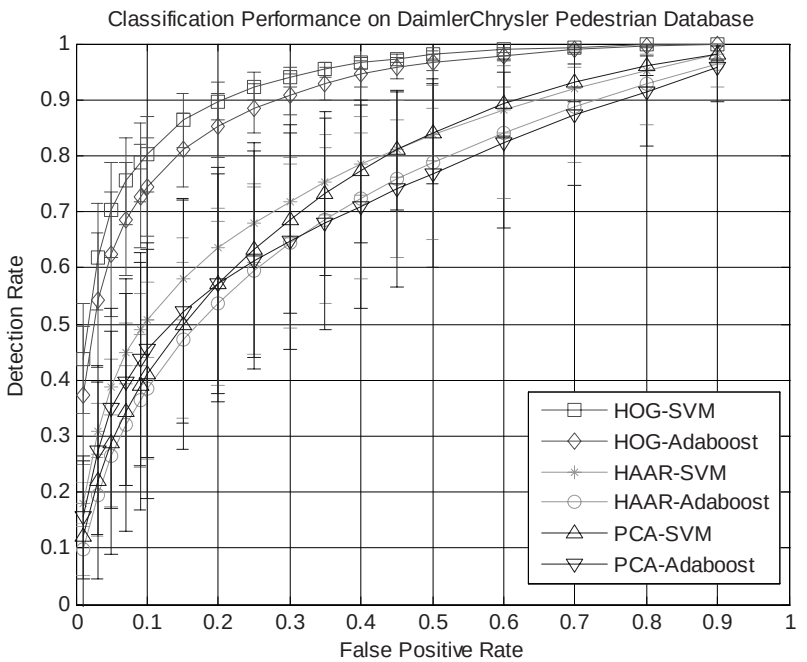
$$\bar{y} \pm t_{(\alpha/2, N-1)} \frac{s}{\sqrt{N}} \approx \bar{y} \pm 1.05s \tag{12}$$

where $\bar{y}$ and s denote the estimated mean and standard deviation, respectively; 1 - α = 0.95 is the desired confidence interval; N = 6 is the number of tests. Hence, the estimated standard deviation of the detection rate approximately represents 95% confidence interval. Although this analysis is somewhat optimistic, as it assumes independency of the individual ROC curves, it still provides a reasonable indicator for performance comparison.

Comparisons of Different Feature and Classifier

We apply each classification method to each type of feature to get a separate investigation into the effectiveness of features and classifiers. For the PCA coefficients, we consider values that capture 95% of the variance. For the overcomplete Haar wavelets, given the input images of size 18x36, we selected wavelets of 4x4 and 8x8, from which we obtained 15x33 and 6x15 features, respectively, for each orientation; hence, a total of 1,755 features. For the HOG features, the cell size is 4x4, the bin number being 9, from which we obtained 864 features. For each feature set, the linear SVM and discrete AdaBoost with "stump" are used; the associated parameters are selected by cross-validation using only the training dataset.

Figure 8. Comparison of various feature extraction and classification methods on DaimlerChrysler pedestrian dataset



The results are shown in Figure 8. Two observations can be made: First, HOG features outperform PCA coefficients and overcomplete Haar wavelets; the reason for this may lie in the fact that HOG can describe the local shape of human more efficiently. PCA coefficients are global features, but the global shape of humans varies greatly. The color also varies greatly, so overcomplete Haar wavelets are not as effective on human detection as on face detection. Second, SVM generally outperform AdaBoost with "stump" as a weak classifier. The reason for this lies in that the features are not independent, and the "stump" cannot capture the relationship very well.

Comparisons of Three Systems

The three systems we compare include Dalal and Trigg's (2005) HOG-SVM-Bootstrapping system, Viola and Jones' (2001) Haar-AdaBoost-Cascade system, and our HOG-AdaBoost-Cascade system. Dalal's system uses HOG features and SVM classifier to train an initial classifier, and then collects more negative images by bootstrapping techniques on the associated additional nonpedestrian images for each training set. The final classifier is trained using initial positive samples and all the negative samples. The ROCs are obtained by varying the threshold

Figure 9. Classification performance of three systems

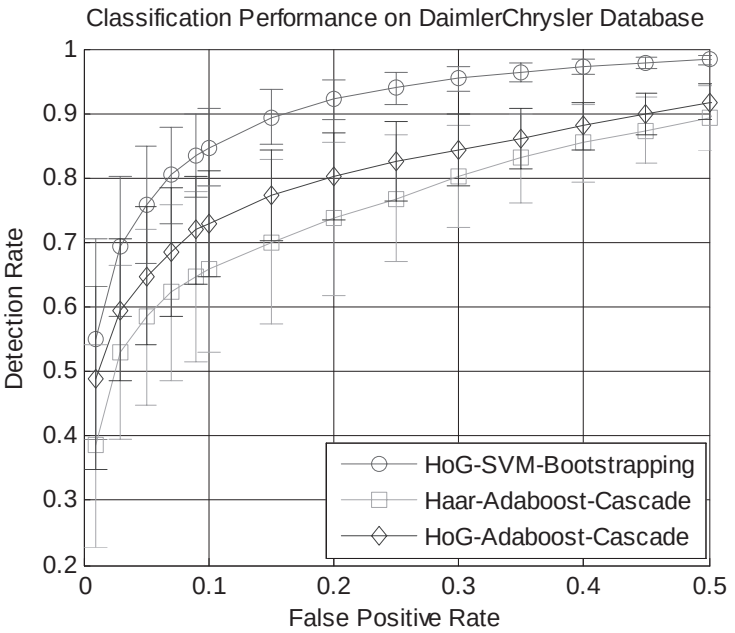


Table 2. Speed comparisons of three systems (CPU P4 3GHz, RAM 1G)

	Sparse Scan (800 windows per image)	**Dense Scan (12800 windows per image)**
HOG-SVM-Bootstrapping	300ms	5sec
Haar-AdaBoost-Cascade	5ms	32ms
HOG-AdaBoost-Cascade	29ms	51ms

of the SVM classifiers. Viola's system is designed for face detection and can also be used on human detection directly. It uses Haar-like features, discrete AdaBoost with "stump" as weak classifier, and the cascade structure. The ROCs are obtained by varying the stage number of the classifiers. Our system uses the same framework as Viola's system, except that we use HOG features instead of Haar-like features.

The classification results on DaimlerChrysler pedestrian database are shown in Figure 9. We also test the speed on the INRIA human database (Dalal & Triggs, 2005), and the results are shown in Table 2. From the results, we can see that HOG features are more powerful than Haar-like features on classification at a little cost of speed; the SVM-Bootstrapping classifier structure outperforms the AdaBoost-Cascade structure in classification performance, but the AdaBoost-Cascade structure can increase the speed greatly. We should note that the speed of our HOG-AdaBoost-Cascade system does not increase greatly from a sparse scan to a dense scan. This is because most of the time is spent on the evaluation of the auxiliary integral images instead of the evaluation of each subwindow. Some of the detection results of our HOG-AdaBoost-Cascade system are shown in Figure 10. From the results, we can see that our system can detect human accurately, irrespective of the illumination.

FUTURE TRENDS

Although a lot of work has been done in the past decade, human detection is still a problem that is far from solved. There are a lot of problems that need to be solved in the future. First, this chapter focuses on the detection of human in static images; how to detect human in video streams is a problem. There are usually two strategies for human detection in video sequence. The first strategy is to detect humans in each static frame and then combine these results through tracking, as in Gavrila, et al. (2004) and Shashua, et al. (2004). The second strategy is to extract features using more than one frame and then build a classifier for human detection, as in Viola, et al. (2003) and Dalal, et al. (2006). Which strategy is more effective needs further research. Second, this chapter only deals with PCA, Haar, and HOG features. It is worth investigating texture and color invariant descriptors. The overall system could use AdaBoost to learn the most relevant features to increase the detection performance. Third, it would be interesting to use the top-down context information. Several researchers have begun to use context by modeling the relationships between different objects or object classes, surrounding image regions, or scene categories (Hoiem, Efros & Hebert, 2006; Kumar & Hebert, 2005; Murphy, Torralba, & Freeman, 2003). In the future, it would be interesting to add these higher-level contexts information to detect human.

Figure 10. Some of the detection results of INRIA human database

CONCLUSION

This chapter proposes an in-depth study of several aspects of human detection in static images. Most of existing systems for human detection use learning-based methods. There are three major modules: feature extraction, classification, and merge of overlapping detections. Haar-like features and HOG features are the most popular feature sets; SVM and cascaded AdaBoost are the most popular classifier design methods. Heuristic clustering and mean shift mode seeking are two major methods for fusion of overlapping blocks. We evaluate the first two modules for different methods using the DaimlerChrysler pedestrian classification benchmark dataset. Experiments show that HOG feature performs the best for pure classification among all the feature sets. The cascaded AdaBoost classifier performs better in speed at a little cost of accuracy than SVM. We also design a novel fast human detection system based on HOG and cascaded AdaBoost. Experiments demonstrate its efficiency for human detection. Future researches should include the detection of human in video streams, the inclusion of color and texture features, and the use of top-down context information.

ACKNOWLEDGMENT

This work has been supported by the Grants NNSF-60573148 and SRFDP-20060003102.

REFERENCE

Broggi, A., Bertozzi, M., Fascioli, A., & Sechi, M. (2000). Shape-based pedestrian detection. *Proceedings of the IEEE Intelligent Vehicles Symposium*, 215–220.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603–619.

Dalal, N. (2006). *Finding people in images and videos* [unpublished doctoral dissertation]. Institut National Polytechnique Grenoble.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2, 886–893.

Dalal, N., Triggs, B., & Schimid C. (2006). Human detection using oriented histograms of flow and appearance. *Proceedings of the European Conference on Computer Vision*, 2, 428–441.

Forsyth, D.A., & Ponce, J. (2003). *Computer vision: A modern approach*. Pearson Education Press.

Freund Y., & Schapire R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119-139.

Gavrila, D.M., Giebel, J., & Munder, S. (2004). Vision-based pedestrian detection: The protector system. *Proceedings of the IEEE Intelligent Vehicles Symposium*, 13–18.

Gavrila, D.M., & Philomin, V. (1999). Real-time object detection for "smart" vehicles. *Proceedings of the International Conference on Computer Vision*, 87–93.

Hoiem, D., Efros, A.A., & Hebert, M. (2006). Putting objects in perspective. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, 2137–2144.

Kumar, S., & Hebert, M. (2005). A hierarchical field framework for unified context-based classification. *Proceedings of the International Conference on Computer Vision*, Beijing, China, 1284–1291.

Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection.

Proceedings of the International Conference on Image Processing, 1, 900–903.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.

Mikolajczyk, K., Schmid, C., & Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust part detectors. *Proceedings of the European Conference on Computer Vision, 1,* 69–82.

Mohan, A., Papageorgiou, C., & Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(4), 349–361.

Munder, S., & Gavrila, D.M. (2006). An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(11), 1863–1868.

Murphy, K.P., Torralba, A.B., & Freeman, W.T. (2003). Graphical model for recognizing scenes and objects. *Proceedings of the Neural Information and Processing Systems,* Vancouver, Canada.

Oren, M., Papageorgiou, C., Sinha, P., Osuna, E., & Poggio T. (1997). Pedestrian detection using wavelet templates. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 193–199.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision, 38*(1), 15–33.

Rowley, H.A. (1999). *Neural network-based face detection* [unpublished doctoral dissertation]. Carnegie Mellon University.

Shashua, A., Gdalyahu, Y., & Hayon, G. (2004). Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. *Proceedings of the IEEE Intelligent Vehicles Symposium,* 1–6.

Vapnik, V. (2000). *The nature of statistical learning theory.* Berlin, Germany: Springer-Verlag Press.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 1, 511–518.

Viola, P., Jones, M., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *Proceedings of the International Conference on Computer Vision, 2,* 734–741.

Zhu, Q., Avidan, S., Yeh, M.C., & Cheng, K.T. (2006). Fast human detection using a cascade of histograms of oriented gradients. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2,* 1491–1498.

Chapter XI
A Brain-Inspired Visual Pattern Recognition Architecture and Its Applications

Fok Hing Chi Tivive
Member, IEEE, and University of Wollongong, Australia

Abdesselam Bouzerdoum
Senior Member, IEEE, and University of Wollongong, Australia

ABSTRACT

With the ever-increasing utilization of imagery in scientific, industrial, civilian, and military applications, visual pattern recognition has been thriving as a research field and has become an essential enabling technology for many applications. In this chapter, we present a brain-inspired pattern recognition architecture that can easily be adapted to solve various real-world visual pattern recognition tasks. The architecture has the ability to extract visual features from images and classify them within the same network structure; in other words, it integrates the feature extraction stage with the classification stage, and both stages are optimized with respect to one another. The main processing unit for feature extraction is governed by a nonlinear biophysical mechanism known as shunting inhibition, which plays a significant role in visual information processing in the brain. Here, the proposed architecture is applied to four real-world visual pattern recognition problems; namely, handwritten digit recognition, texture segmentation, automatic face detection, and gender recognition. Experimental results demonstrate that the proposed architecture is very competitive with and sometimes outperforms existing state-of-the-art techniques for each application.

INTRODUCTION

Humans and other advanced species rely heavily on their visual pattern recognition system for survival. They possess an innate ability to process visual information and make decisions on the fly. For example, humans can recognize familiar faces in a crowd, characters and words in a newspaper, and a signature on a bank check, whereas animals can easily distinguish prey from predator in a cluttered natural environment. Despite several decades of intensive research effort, the design of human-competitive visual pattern recognition machines remains a difficult and illusive task. The traditional pattern recognition approach is to divide the system into two stages: the feature extraction stage and the pattern classification stage. The success of such an approach relies heavily on the successful extraction of salient features that are robust to intrinsic and extrinsic variations in the input image. However, feature extraction can be a computationally intensive process and depends heavily on the expert knowledge of the designer—it is more of an art than an exact science. More recently, many researchers have shifted their attention to natural intelligence in an attempt to design intelligent machines and pattern recognition systems. Among the new computational paradigms inspired by natural intelligence are *artificial neural networks* (ANNs) and *evolutionary computation*. Artificial neural networks achieve their computational power by learning complex models from examples, similar to a human brain. For example, the conventional feedforward artificial neural networks, also known as *multilayer perceptrons* (MLPs), have been shown to be universal approximators under very general conditions (Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe & White, 1989). Due to their capability to learn directly from input data and produce accurate results, ANNs have been applied to a broad spectrum of applications, covering areas as diverse as finance, medicine, engineering, geology, physics, genomics, and so forth.

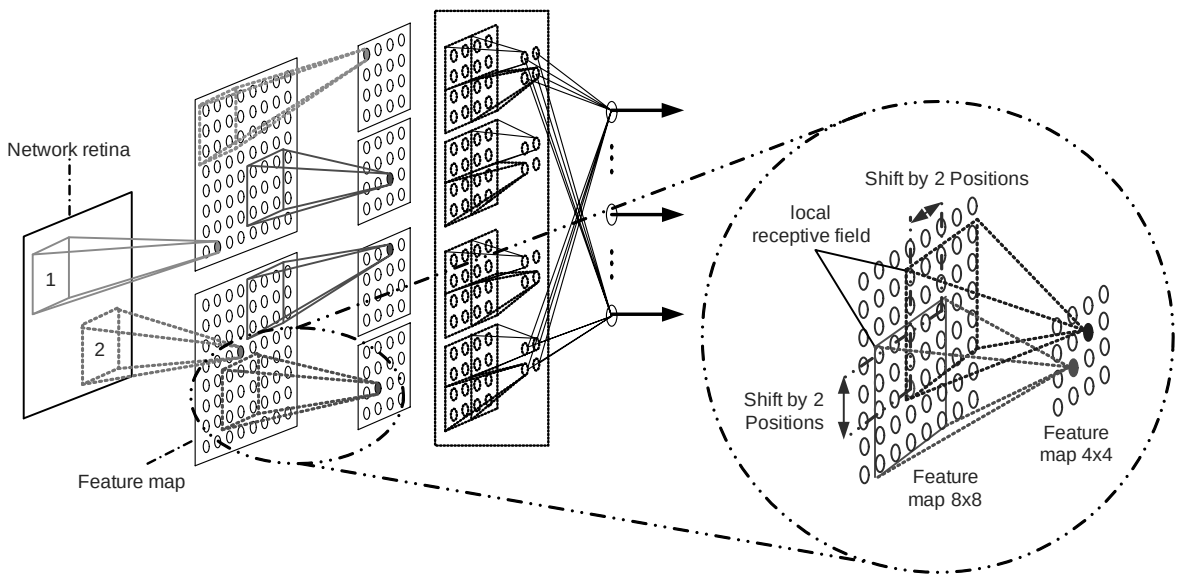
However, one of the drawbacks of conventional feedforward ANNs is that the size of the network (i.e., the number of its free parameters) grows with the input dimension, making learning a much harder task. Moreover, the generalization ability of the network suffers due to the problem of overfitting and loss of crucial information as correlations between neighboring inputs are totally ignored. Therefore, in the past two decades, researchers have focused their attention not only on the development of fast learning algorithms, but also on the identification of significant network structures and weight constraints that can improve the generalization ability and simplify the network architecture. Inspired by the hierarchical and retinotopic organization of the early visual system, a number of researchers have developed hierarchical neural networks that can operate on two-dimensional (2-D) input images, extract visual features, and perform classification within the same network architecture. For example, Fukushima, Miyake, and Ito (1983) proposed the well-known *neocognitron* for visual pattern recognition based on the structure of the visual cortex suggested by Nobel laureates Hubel and Wiesel (1965). Years later, LeCun and his colleagues (1989) proposed a series of convolutional neural network (CoNN) architectures based upon the three structural ideas of local receptive fields, weight sharing, and subsampling. These hierarchical neural networks can easily deal with variability in object shape and possess a certain degree of local invariance to distortions and translations; they exhibit better generalization ability compared to the fully-connected feedforward networks (Kroner & Moratz, 1996; LeCun, 1989). However, the resulting network architectures are often massive with a very large number of free parameters to be determined through learning. Furthermore, most convolutional neural network architectures employ the simple sigmoid neuron as the main processing element of the architecture.

In this chapter, we present a brain-inspired neural network architecture that can easily be adapted to solve various visual pattern recognition tasks. The architecture is derived from its predecessor convolutional neural network models (Fukushima et al., 1983; LeCun, Bottou, Bengio & Haffner, 1998). However, the proposed architecture employs shunting inhibition as the elementary visual processing mechanism for feature extraction; the biophysical mechanism of shunting inhibition plays a major role in visual information processing in the brain. Furthermore, three systematic connection schemes have been developed to link the processing layers in a hierarchical structure. The next section describes the network architecture, the shunting neural model, the connection schemes, and a hybrid learning method developed for training the proposed visual pattern recognition architecture. In Section 3, the proposed brain-inspired neural network architecture is employed for solving four visual pattern recognition problems based on real-world data. Experimental results are presented, which show that the proposed architecture achieves competitive performance as the state-of-the-art solutions developed specifically for these problems. Finally, concluding remarks are presented in Section 4.

THE BRAIN-INSPIRED NEURAL NETWORK ARCHITECTURE

The proposed visual pattern recognition architecture is based upon the three concepts of weight sharing, subsampling, and local receptive fields (Fukushima et al., 1983; LeCun et al., 1998). It has a 2-D multilayered network topology with feedforward connections. The input layer, also called the network retina, is a 2-D array of input nodes that receive signals from images of arbitrary size. The input layer is succeeded by several feature extraction layers. The feature extraction stage is then followed by a pattern classification stage, which may comprise one or more process-

Figure 1. A three-layer architecture of the proposed brain-inspired neural network: the first hidden layer contains two feature maps, followed by four feature maps in the second hidden layer and one output layer. Local averaging and down-sampling operations are applied to the second hidden layer to reduce the number of input signals fed to the classification layer.



ing layers. Figure 1 illustrates a three-layered architecture comprising two feature extraction layers and one classification layer.

Visual Feature Extraction

Each feature extraction layer comprises several planes of shunting neurons, known as *feature maps*. In a conventional fully connected MLP architecture, each neuron receives inputs from all neurons in the previous layer, and the number of its free parameters, or weights, is equal to the number of connections (plus the bias term). By contrast, each neuron in a feature map receives inputs only from a local region in the input plane, the *local receptive field*. Therefore, the number of connection weights is related to the receptive field size and not the size of the input plane. Furthermore, all the neurons in a feature map share the same set of connection weights, or *weight sharing*. The mechanism of weight sharing forces the neurons in a feature map to perform the same operation on different parts of the input plane. Hence, the role of a feature map is to extract the same elementary feature from several locations in the input; other feature maps in the same layer extract different features using different sets of weights.

Another operation that is implemented at each hidden layer is the down-sampling of the feature maps, which has the effect of reducing the number of connections and introducing into the architecture some degree of tolerance to spatial shifts and distortions. By down-sampling, some information about the exact location of the detected visual feature is discarded, but once a feature is detected, its absolute location is no longer important—only its position relative to other features is relevant to the classification (LeCun et al., 1998). Down-sampling by a factor of two is achieved by simply shifting the centers of receptive fields of adjacent neurons in the feature map by two positions: horizontally and vertically. However, if the subsampling factor is greater than two, the output of the feature map is first convolved with a low-pass filter before down-sampling in order to reduce the aliasing effect.

Network Connection Schemes

Even though the link between the neuron and its inputs is established by the use of a local receptive field, a connection scheme between layers is required to construct the network topology. A common connection scheme is to connect each feature map to all the feature maps in the succeeding layer (i.e., a full-connection scheme). In a fully connected network, each hidden layer has an arbitrary number of feature maps, depending on the number of visual features to be extracted. In addition, there are two partial-connection schemes that have been developed for the proposed architecture: *binary* and *toeplitz*. In these partial-connection schemes, each hidden layer contains twice as many feature maps as the previous layer, barring the first layer, which can have an arbitrary number of feature maps. A schematic diagram of these two partial connection schemes is shown in Figure 2. In a binary-connected network, each feature map branches out to two feature maps, forming a binary tree. In the toeplitz connection scheme, all feature maps in the first hidden layer connect to the same number of feature maps in the second layer, but the number of connections is determined by the number of feature maps in the second layer. A feature map in the second layer may have one-to-one or one-to-many connections with the feature maps in the preceding layer, but the connection matrix has a toeplitz form. As an example, suppose the first hidden layer has four feature maps, labeled A, B, C, and D, and the second hidden layer has eight feature maps, labeled F1 to F8. The toeplitz connections between these two layers are shown in Table 1. In this case, the first and last feature maps, F1 and F8, form one-to-one connections with feature maps A and D, respectively; F2 forms connections with feature maps A and B; and F3 is connected to feature maps A, B, and C. The remaining feature

Figure 2. Partial-connection schemes: (a) Binary connection scheme, each feature map branches out to two feature maps of the succeeding layer forming a binary tree; (b) toeplitz connection scheme, each feature map in the first hidden layer connects to five feature maps in the second hidden layer forming a toeplitz connection matrix

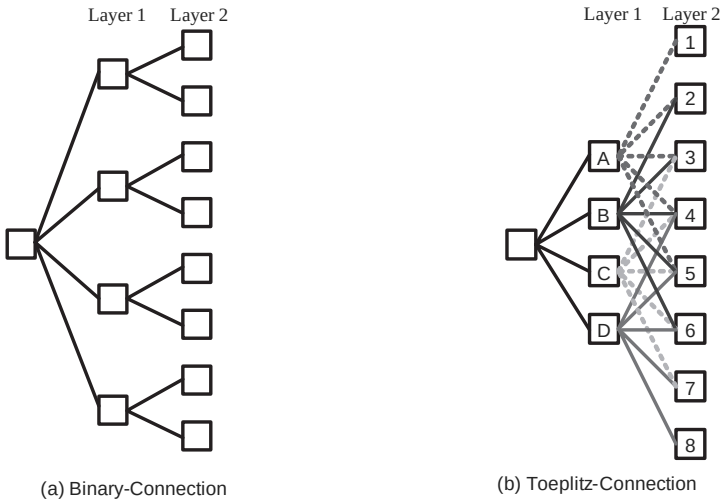


Table 1. Toeplitz connections scheme: An X signifies the presence of a connection between the feature maps of the first and second hidden layers

Layer 2 / Layer 1	F1	F2	F3	F4	F5	F6	F7	F8
A	X	X	X	X	X			
B		X	X	X	X	X		
C			X	X	X	X	X	
D				X	X	X	X	X

maps (F4–F7) complete the toeplitz connection matrix. Note that each feature map of the first hidden layer connects to five feature maps of the second hidden layer.

Pattern Classification Stage

The pattern classification stage may comprise one or more processing layers. It receives inputs from the last feature maps and outputs a class label for the input pattern. There are two possibilities for generating input signals to the pattern classification stage. In the first case, the outputs of the feature extraction neurons are used directly as input signals, without preprocessing. This, however, may result in a large number of noisy inputs. To reduce the number of inputs and noise, filtering and subsampling operations are applied to all the feature maps. Then the filtered signals are used as inputs to the classification stage. The classification stage may be any classifier that can be trained together with the feature extraction stage.

Although, any type of classifier may be used in the classification stage, herein we restrict the classification stage to being a single layer of sigmoid neurons. Therefore, the response of an output neuron is a weighted sum of its input

signals, added to a bias term and passed through an activation function:

$$y = h\left(\vec{V} \cdot \vec{Z} + b\right) \quad (1)$$

where *y* is the output of the neuron, *h* is the output activation function, $\vec{V}$ is the vector of connection weights, and $\vec{Z}$ is the vector of feature inputs from the last hidden layer of the network.

Shunting Neuron Model

It is well known that biological neurons receive inputs from excitatory and inhibitory synapses. In particular, the biophysical mechanism of shunting inhibition has been shown to play a very important role in neuronal information processing in the brain. Mitchell and Silver (2003) investigated the ability of shunting inhibition to modulate neuronazl gain during synaptic excitation and suggested that it can act as a cellular mechanism for gain control, which is vital for normal sensory and cognitive activity in the brain. Prescott and Koninck (2003) observed that shunting inhibition can indeed mediate firing rate gain control under background synaptic noise and dendritic saturation, and established that the nonlinearity caused by dendritic saturation enhances the divisive effect of shunting inhibition on the output firing rate. Other researchers have also shown that the mechanism of shunting inhibition plays a major role in the functional organization and visual information processing in cortical cells (Anderson, Carandini, & Ferster, 2000; Borg-Graham, Monier, & Fregnac, 1998; Fregnac, Monier, Chavane, Baudot, & Graham, 2003). More recently, experiments conducted by Vida, Bartos, and Jonas (2006) demonstrated that shunting inhibition introduces several functional advantages to the interneuron network function, such as increasing the robustness of gamma oscillations and boosting of coherent oscillatory activity in interneuron networks.

Bouzerdoum and Pinter (1993), inspired by this biophysical mechanism, developed an artificial shunting neuron model that was used to solve various image processing tasks (Beare & Bouzerdoum, 1999; Bouzerdoum, 1993, 1994; Bouzerdoum & Pinter, 1992; Cheung, Bouzerdoum, & Newland, 1999; Pontecorvo & Bouzerdoum, 1997). This type of neuron was subsequently used in feedforward networks for supervised pattern classification and regression (Arulampalam & Bouzerdoum, 2003; Bouzerdoum, 1999, 2000). It has been established that shunting inhibitory neurons are more computationally powerful than their sigmoid counterparts; a single shunting neuron can solve linearly nonseparable classification problems (e.g., the XOR and parity problems).

To mimic the process of visual feature extraction in the brain, shunting inhibition is employed here for information processing by the feature maps. The response of a static feedforward shunting neuron can be modeled as:

$$z = \frac{g\left(\vec{W} \cdot \vec{I} + b\right)}{a + f\left(\vec{C} \cdot \vec{I} + d\right)} \quad (2)$$

where *z* is the neuron output, $\vec{I}$ is the vector of external inputs, *a* is the passive decay rate, $\vec{W}$ and $\vec{C}$ are, respectively, sets of excitatory and inhibitory weights, *b* and *d* are constants known as the *bias terms*, and *f* and *g* are the activation functions of the neuron. As stated earlier, all the neurons in a feature map share the same set of weights (i.e., $\vec{W}$ and $\vec{C}$). However, there are two strategies for adapting the bias terms and the passive decay rate. All the neurons in a feature map can either share the same bias parameters as well as the passive decay rate or have their own biases and passive decay rates. Herein, all the neurons in a feature map share the same set of weights, the bias terms, and the passive decay rate.

One of the virtues of the shunting inhibitory neuron is that it possesses an adaptive input-output transfer characteristic. By contrast, sigmoid and radial basis function neurons have their input-

output transfer characteristic fixed by the choice of the activation function and can only be translated, rotated, or stretched. On the other hand, the shape of the input-output transfer characteristic of a shunting neuron is determined by the network parameters as well as the activation functions. The advantage is that by just varying the network parameters (e.g., through learning), different types of decision surfaces can be generated. Figure 3 illustrates some examples of input-output transfer characteristics of a shunting neuron with fixed activation functions *f* and *g*.

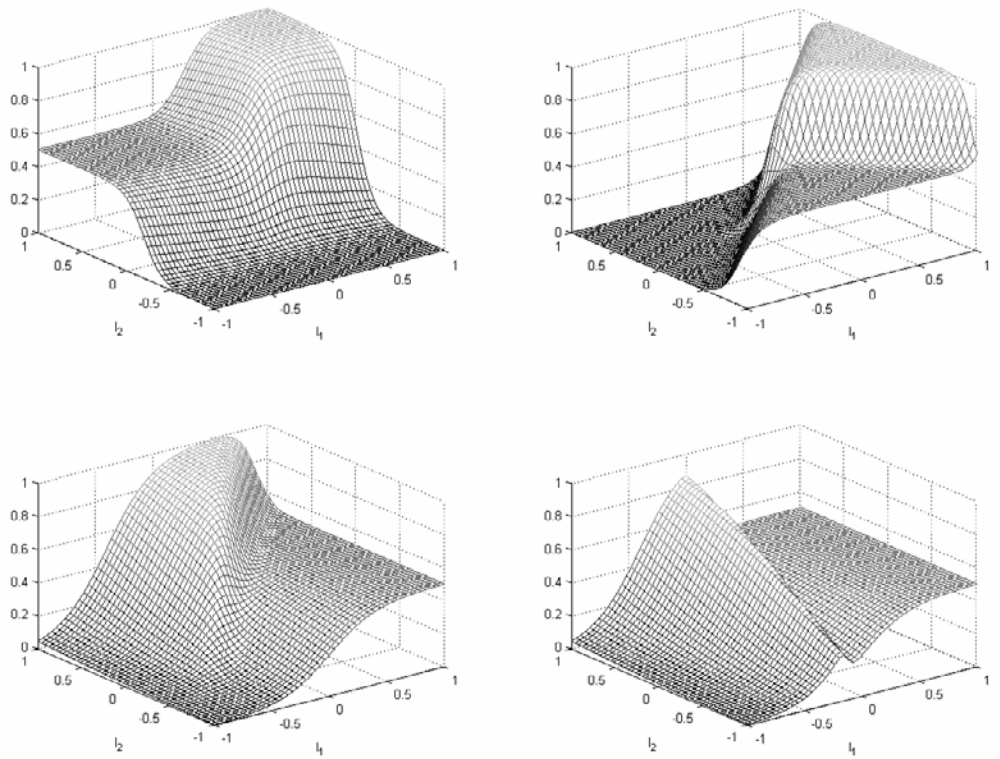
Network Training Process

Since the purpose of the proposed architecture is to learn various visual pattern recognition tasks, a series of supervised learning algorithms have been developed based on error backpropagation. The algorithms are batch training techniques, ranging from the simple gradient steepest descent to quasi-Newton optimization and Levenberg-Marquardt algorithms. In particular, a fast hybrid first-order method, based on the combination of Rprop (Riedmiller, 1994), Quickprop (Fahlman, 1988), and SuperSAB (Tollenaere, 1990), has been developed to train the proposed architecture (Tivive & Bouzerdoum, 2006). The weight update rule of this hybrid method is given by:

$$w(k+1) = w(k) + \Delta w(k) + \mu(k)\Delta w(k-1) \quad (3)$$

where $\Delta w(k)$ is the local update weight and is computed based on the behavior of the local gradient $g(k)$ during two successive iterations, similar to the Rprop technique. The mathematical

Figure 3. Different shapes of input-output transfer characteristics can be generated by adapting the weights and bias terms of the shunting neuron



expressions for the step size and weight update of the local weight are given by:

$$\gamma(k) = \begin{cases} \min(1.2\gamma(k-1), 10) & \text{if } g(k)g(k-1) > 0 \\ \max(0.5\gamma(k-1), 10^{-10}) & \text{if } g(k)g(k-1) < 0 \\ \gamma(k-1) & \text{if } g(k)g(k-1) = 0 \end{cases} \quad (4)$$

and

$$\Delta w(k) = -sign(g(k))\gamma(k) \quad (5)$$

In the traditional gradient descent with momentum algorithm, the momentum term $\mu(k)$ is often fixed to a predefined value in the interval (0, 1). In the hybrid training method, however, it is adjusted with respect to the magnitude of the Quickprop step; that is,

$$\tilde{\mu}(k) = \left| \frac{g(k)}{g(k-1) - g(k)} \right| \quad (6)$$

To prevent the momentum term from dominating the weight update, it is restricted to the interval [0.5, 1.5]. Moreover, when either the previous or current local gradient is zero, the momentum rate is set to zero in order to prevent a weight update in the succeeding iteration. In other words, at the k^{th} iteration, the momentum term μ (k) is given by:

$$\mu(k) = \begin{cases} \min(\tilde{\mu}(k), 1.5), & \text{if } g(k)g(k-1) > 0 \\ \max(\min(\tilde{\mu}(k), 1.5), 0.5), & \text{if } g(k)g(k-1) < 0 \\ 0, & \text{if } g(k)g(k-1) = 0 \end{cases} \quad (7)$$

To further increase the convergence speed of the hybrid method, a small portion of the current gradient is added to the weight whenever there is a decrease in error:

$$w(k+1) = w(k+1) - \alpha(k)g(k) \quad (8)$$

The method used to compute the learning rate $\alpha(k)$ is based on the SuperSaB training method:

$$\alpha(k) = \min(\tilde{\alpha}(k), 0.9) \quad (9)$$

where

$$\tilde{\alpha}(k) = \begin{cases} 1.2\alpha(k-1) & \text{if } g(k)g(k-1) > 0 \\ 0.5\alpha(k-1) & \text{if } g(k)g(k-1) < 0 \\ \alpha(k-1) & \text{if } g(k)g(k-1) = 0 \end{cases} \quad (10)$$

To summarize the hybrid training method, a pseudo-code of the algorithm is given in Table 2.

Before training commences, the weights of the neurons are initialized with small random values taken from a uniform distribution on the interval [-1/*N*, 1/*N*], where *N* is the receptive field size; the biases are initialized similarly with *N* = 1. To prevent the denominator term in (2) from becoming zero, and hence avoid division by zero, it is bounded from below by a small positive constant ε:

$$a + f\left(\vec{C} \cdot \vec{I} + d\right) \geq \varepsilon > 0 \quad (11)$$

This was achieved by placing a lower bound on the passive decay rate:

$$\tilde{a} \geq \varepsilon - \min(f) \quad (12)$$

where *f* is the activation function of the denominator. Accordingly, the passive decay rate is initialized in the range (0, 1].

APPLICATIONS

This section presents four visual pattern recognition applications based on real-world data. The first two, visual document analysis and texture segmentation, are multiclass problems where the networks are trained to classify input patterns into one of several classes. The other two, face detection and gender recognition, are tackled because of their importance and large potential

Figure 4. A pseudo-code of the hybrid training method

Input: Initialize $\gamma_i \leftarrow 0.001$, $\mu_i \leftarrow 0.01$, $\alpha_i \leftarrow 0.1$. Calculate the local gradient $g_i(k)$, where $i = 1, \cdots, N$ iterations.

while stopping criterion is not met **do**
  Calculate the adaptive momentum rate $\widetilde{\mu}_i(k)$, according to (6).
  if $g_i(k)g_i(k-1) > 0$ **then**
    $\gamma_i(k) \leftarrow \min(\eta_{inc}\gamma_i(k-1), \gamma_{max})$,
    $\widetilde{\alpha}_i(k) \leftarrow \eta_{inc}\alpha_i(k-1)$.
    $\mu_i(k) \leftarrow \min(\widetilde{\mu}_i(k), 1.5)$,
  else if $g_i(k)g_i(k-1) < 0$ **then**
    $\gamma_i(k) \leftarrow \max(\eta_{dec}\gamma_i(k-1), \gamma_{min})$,
    $\widetilde{\alpha}_i(k) \leftarrow \eta_{dec}\alpha_i(k-1)$,
    $\mu_i(k) \leftarrow \max(\min(\widetilde{\mu}_i(k), 1.5), 0.5)$,
    $g_i(k) \leftarrow 0$.
  else if $g_i(k)g_i(k-1) = 0$ **then**
    $\mu_i(k) \leftarrow 0$,
    $\gamma_i(k) \leftarrow \gamma_i(k-1)$,
    $\widetilde{\alpha}_i(k) \leftarrow \alpha_i(k-1)$.
  end if
  $\alpha_i(k) \leftarrow \min(\widetilde{\alpha}_i(k), 0.9)$.
  $\Delta w_i(k) \leftarrow -sgn(g_i(k))\gamma_i(k)$.
  if $g_i(k)g_i(k-1) < 0$ and $E(k) > E(k-1)$ **then**
    $\Delta w_i(k) \leftarrow -\mu_i(k)\Delta w_i(k-1)$,
    $w_i(k+1) \leftarrow w_i(k)$.
  else
    $w_i(k+1) \leftarrow w_i(k) + \Delta w_i(k) + \mu_i(k)\Delta w_i(k-1)$.
  end if
  if $E(k) < E(k-1)$ **then**
    $w_i(k+1) \leftarrow w_i(k+1) - \alpha_i(k)g_i(k)$.
  end if
end while

in biometric applications. Since we are stepping into a new era of intelligent man-machine interactions, the latter two applications have recently attracted considerable interest.

Visual Documents Analysis

One of the very first visual pattern recognition tasks that researchers attempted to solve is optical character recognition. Handwritten character and digit recognition is a challenging problem because of the difficulty of designing algorithms that can tolerate distortions in the input data and variations in the writing styles. Many techniques have been reported, and those that have achieved state-of-the-art performance are mostly neural-based methods (Calderon, Roa & Victorino, 2003; LeCun et al., 1998; Poisson, Gaudin & Lallican, 2002; Simard, Steinkraus & Platt, 2003). However, the neural networks used in these handwritten digit recognition systems often have massive and complex architectures with a very large number of weights; consequently, a very large training dataset is often required to properly train the neural network.

The proposed pattern recognition architecture was applied to handwritten digit recognition. First, three networks—binary-, toeplitz-, and fully connected networks—were trained on a small data set containing 10,000 patterns (1,000 patterns per digit class) taken from the MNIST database.[1] All three networks contain two feature extraction layers comprising 12 feature maps and one classification layer comprising 10 output units; the input retina has the size of 24x24 pixels, and all neuron receptive fields are 5x5. After training, the networks were tested on the entire test set of the MNIST database; their classification rates are listed in Table 2. With a small training set, the proposed networks already achieve classification

Table 2. Classification performances of the binary-connected, toeplitz-connected, and fully connected networks on the test set (10,000 handwritten digit patterns) of the MNIST database

Network Architecture	Classification Rate for Each Digit Class (%)										Accuracy (%)
	0	1	2	3	4	5	6	7	8	9	
Binary	98.3	97.9	95.1	92.2	93.0	91.3	95.9	93.4	93.0	91.8	94.1
Toeplitz	97.1	96.7	96.6	95.0	92.8	92.2	95.5	91.0	89.3	90.1	93.6
Full	95.9	96.5	92.4	86.6	89.1	85.2	93.7	89.0	84.7	88.2	90.2

Table 3. A confusion matrix showing the performance of the binary-connected network, trained on the entire training set of the MNIST database. An additional column is added to the confusion matrix to list the recognition rate for each digit class.

Actual Class	Network Predicted Class										Recognition Rate (%)
	0	1	2	3	4	5	6	7	8	9	
0	970	0	1	0	0	0	6	2	1	0	99.0
1	0	1120	2	2	0	0	2	1	8	0	98.7
2	7	1	1001	6	1	0	0	3	5	0	97.8
3	0	0	5	982	0	7	0	6	10	0	97.2
4	1	0	3	0	954	0	4	1	0	19	97.1
5	3	0	2	11	1	862	5	1	4	3	96.6
6	8	3	4	0	2	2	935	2	2	0	97.6
7	1	2	8	6	2	0	0	999	0	6	97.6
8	4	2	1	6	4	5	1	3	945	3	97.6
9	4	4	0	10	11	7	1	5	8	959	95.0
Overall accuracy											97.3

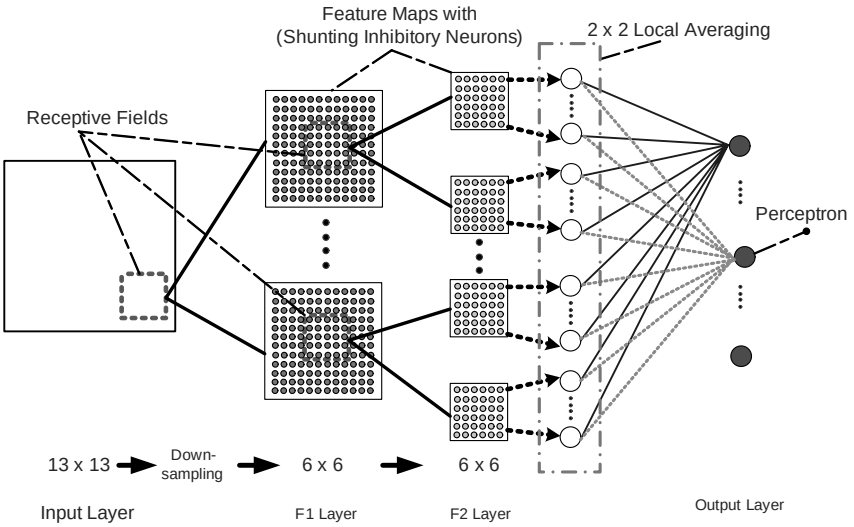
rates higher than 90%. Moreover, these results show that the partially connected networks perform better than the fully connected one.

In the second stage, a binary-connected network comprising 24 feature maps (eight in the first layer and 16 in the second layer) and 2,722 weights was trained using the 60,000 training patterns in the MNIST database. The results on the test set are shown in Table 3. The network achieves an overall recognition rate of 97.3%. Most of the false recognitions are due to the writing style; some patterns are written with heavy strokes that even humans have trouble recognizing correctly.

Texture Segmentation

Texture segmentation is one of the most widely studied pattern recognition problems because the textural features of an image are vital cues for many machine vision applications such as classification of ground cover types in satellite imagery, industrial and biomedical surface inspection, and content-based image retrieval. A number of texture analysis techniques have been reported in the literature, and two comprehensive reviews on common texture analysis approaches are given in Materka and Strzelecki (1998) and Tuceryan and Jain (1998). Even though numerous studies have been conducted, texture segmentation remains a

Figure 5. The architecture of a binary-connected network used for texture classification. In the first hidden layer, the feature map is down-sampled to 6x6, and between the second and output layers, a local averaging operation is employed.



subtle pattern recognition problem because the extraction of informative features from texture images is a very difficult task. In recent years, Gabor and wavelet frame decompositions have become popular analysis tools for texture analysis, and significant improvement in classification results have been achieved. Nevertheless, there is still the problem of selecting the appropriate filter banks for the given set of texture images. Some researchers have attempted to develop artificial neural network techniques to extract the discriminative features from the texture images by either manual tuning or using a supervised learning algorithm to adapt the network weights as texture filters (Jain & Karu, 1996; Lin & Shou, 2005).

In the proposed texture segmentation system, a three-layer binary network with a 13x13 pixel retina has been used to produce an output that indicates the texture class to which the center pixel belongs. The receptive fields used in the first and second hidden layers are 7x7 and 5x5, respectively. The down-sampling operation is performed only at the first hidden layer in order to reduce the spatial resolution of the feature maps to 6x6. Between the second hidden and the output layers, a local averaging operation is performed on all the feature maps; that is, a nonoverlapping mask of 2x2 is used to average every four outputs into a single signal, which is then fed to the output neurons. Figure 5 shows a schematic diagram of the binary-connected network for texture classification.

The training patterns are texture images from the Brodatz image database (Brodatz, 1966), which contains natural textures with different density, roughness, and regularity. This database has been widely used and has become a benchmark for texture analysis algorithms. Randen and Husøy (1999) created a set of texture mosaics from these images to compare different texture classification approaches.[2] To evaluate the texture segmentation system, five texture mosaics are used: 11(a) (Nat-5c), 11(d) (Nat-5v3), 11(h) (Nat-10), 12(a) (D4D84), and 12(c) (D5D92) (see Figure 6).

Most texture segmentation approaches include a feature conditioning stage to improve the classification performance; it usually consists of a

Figure 6. Texture mosaics used to evaluate the proposed network

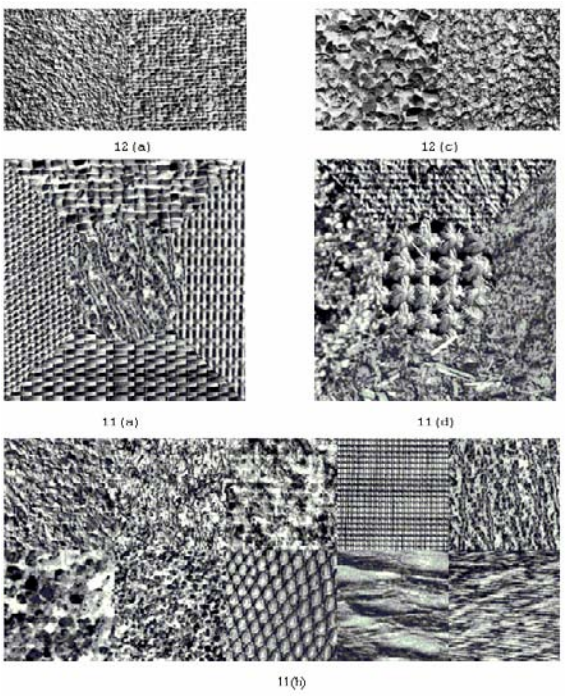
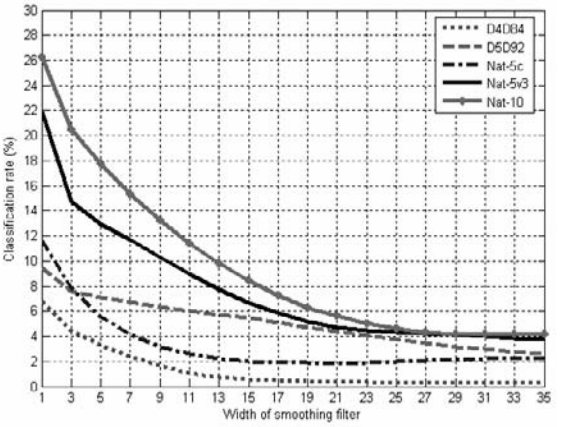


Figure 7. The classification performances of the texture segmentation system as a function of the smoothing mask size, which is applied to the output image before the winner-take-all operation is invoked



smoothing filter followed by a nonlinear squashing transformation. In our segmentation method, a smoothing filter is convolved with the output images obtained from the network responses before applying the winner-take-all scheme to generate the segmented output image. Figure 7 demonstrates that the classification error rate decreases markedly by increasing the width of the smoothing filter.

For each texture mosaic shown in Figure 6, a different network was developed; the different network configurations are shown in Table 4. Each output neuron represents a texture class and produces an image of network responses having the same size as the input texture mosaic. A winner-take-all scheme is used to determine the class of the input pixel (i.e., the maximum output indicates the texture class of the corresponding pixel in the input image). Table 5 presents the classification error rates of the five networks, along with the performance results of other texture classification approaches; namely, the co-occurrence matrix, Gabor, wavelet, and quadrature mirror filters (QMF); the results of other approaches are taken from Randen and Husøy (1999). The experimental results in Table 5 show that the proposed brain-inspired pattern recognition architecture can easily be employed for texture segmentation. On the texture mosaics 11(d) and 11(h), the proposed system performs better than the co-occurrence, wavelet, and Gabor filter methods. Furthermore, when a lowpass filtering is applied to the network outputs, the classification error rates across all five texture images decrease markedly. Figure 8 shows an example of the segmented output images of a texture mosaic with and without filtering.

Automatic Face Detection

With the increasing demand for visual surveillance and security systems, face detection has attracted considerable attention from the computer vision research community. It has become an important technology for many applications such as biometric authentication, surveillance, intelligent man-machine interactions, and so forth. For a human to recognize faces in a crowd is an easy

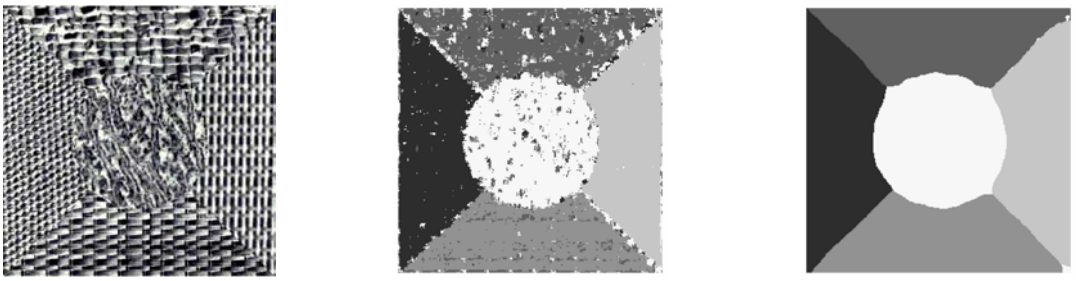
Table 4. Network configuration used for different number of texture classes

Network Index	No. of Weights	No. of Feature Maps		No. of Output Neurons
		Layer 1	Layer 2	
Net-01	974	4	8	2
Net-02	1490	5	10	5
Net-03	3106	8	16	10

Table 5. Error rates of different texture classification approaches

Texture Classification Approach	Texture Mosaics Used for Testing					Mean (%)
	11(a)	11(d)	11(h)	12(a)	12(c)	
Proposed method	11.7	21.9	26.2	6.8	9.4	15.2
Proposed method with post filtering	1.9	3.8	4.2	0.3	2.6	2.6
Co-occurrence	9.9	51.1	35.3	1.9	3.3	20.3
Gabor filter bank	8.2	36.9	39.7	6.5	15.6	21.4
Wavelet – Daubechies 4	8.7	23.4	40.9	5.7	8.2	17.4
QMF filter bank – f16b (d)	8.7	18.4	39.8	8.1	8.2	16.6

Figure 8. Example of a segmented texture mosaic: (a) Input image, (b) network responses, and (c) segmented image of the texture mosaic



(a) (b) (c)

task, but for a machine to identify and locate a human face in an image or a sequence of images is still a challenging problem due to the fact that human faces are highly nonrigid objects with intrinsic and extrinsic variations. Yang, Kriegman, and Ahuja (2002) and Hjelmas and Low (2001) published two comprehensive surveys on face detection techniques. Most of the popular face detectors are appearance-based, which can capture the representative variability of facial appearance from the training images (Garcia & Delakis, 2004; Rowley, Baluja & Kanade, 1998; Viola & Jones, 2001). The general framework of these face detection systems comprises two basic components: a face classifier that can distinguish between face and nonface pattern, and a face localization procedure that uses the output of the face classifier to detect and localize human faces in a given image.

Face Classifier

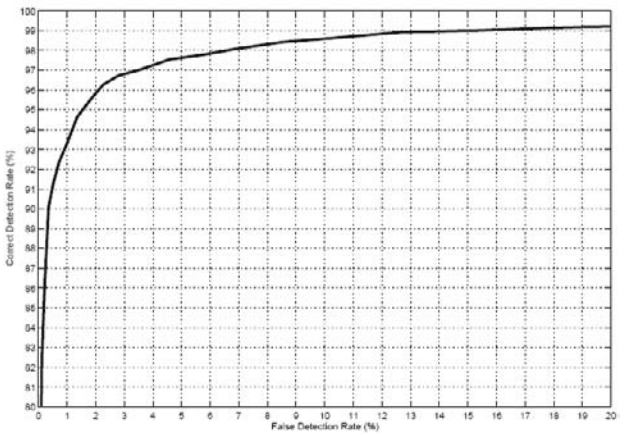
The face classifier is a network with three processing layers and a 36x36 input retina; the first

hidden layer has two feature maps, and the second layer has four feature maps. The feature maps of the first and second hidden layers are subsampled by a factor of two, and those feature maps in the second layer are convolved with a low-pass filter followed by a down-sampling operation. The output layer has one neuron whose output is used to classify the input pattern into a face or a nonface. The training data were taken from the Phung face database (Phung, Bouzerdoum & Chai, 2005). This database contains a large number of face patterns segmented from digital images collected from various sources. These images contain faces of people of different ages, ethnic backgrounds, and genders. The images also vary in terms of background, lighting conditions, facial expression, and pose. Figure 9 shows a sample of the facial images in the database.

The desired outputs corresponding to the face and nonface patterns are set to 1 and -1, respectively. Generating a training set with representative nonface patterns is a difficult task because the nonface class is extremely large; any background window can be regarded as a nonface pattern. To overcome this problem, a bootstrap training procedure (Rowley et al., 1998) was used to avoid collecting nonface patterns manually. A separate test set of segmented face and nonface patterns was used to assess the performance of the face classifier. The ROC (receiver operating characteristic) curve of the classifier is presented in Figure 10. There is a trade-off between the correct classification rate and the false alarm rate; by simply changing a threshold value, the correct classification rate may be increased arbitrarily at the expense of the false alarm rate. At

Figure 9. Examples of face patterns used to train the face classifier

Figure 10. The ROC curve for the face/nonface classifier



5% false alarm rate, the correct face detection rate is 97.6%.

Face Localization

After the network is trained as a face/nonface classifier, it is employed to detect and localize faces in digital image, or image sequences. Since the network retina is a fixed-size array of 36x36 input nodes and the human faces often appear with different sizes, the input image is subsampled at different scales to form an image pyramid. At each level of the pyramid, the image is processed by the network, resulting in a pyramid of network responses. Due to the twofold down-sampling operation performed at each processing layer, the image generated at the output layer is 1/16 the size of the input image. This means that adjacent pixels in the output image represent network responses to input windows whose centers are located four pixels apart. Once an output image is computed, a threshold T_{net} is applied, and those network responses that are greater than T_{net} are considered face candidates; their positions are mapped back to an image having the same size as the original input image. This process is repeated for every level of the image pyramid. The threshold T_{net} is often computed from a set of face and nonface patterns where the minimum error occurs; however, in practice, this threshold does not work well for every image. One solution to overcome this is to set T_{net} initially to zero and process the smallest image (i.e., the top-level image) of the pyramid. Then the mean of all the network responses that are greater than zero is used as the threshold for the next level of the pyramid, and so forth.

During the face detection process, there is always a certain number of background windows that will generate high network responses and hence will be misclassified as face candidates. To reduce the number of false alarms, the following postprocessing steps are taken. Assuming the human face is symmetric, the detected face candidate is folded along the Y-axis (mirror face image) and passed back to the network. The average of both network responses is taken as the final score of the detected face candidate. If the final score is less than T_{net}, the network response at that location is set to zero. Moreover, a number of overlapping detections usually occurs around the true face position. To fuse these overlapping detections into a single face candidate, a simple clustering method is applied. First, at each level of the image pyramid, a search in an 8x8 region is performed around each face candidate, and the number of positive responses within the search grid is taken as the confidence score of the face candidate. Then all the face candidates from the series of output images are stored in a list and sorted in descending order according to their confidence scores. Suppose that S_{max} is the size of the top face candidate in the list (i.e., the face candidate with the highest score). All face candidates whose centers are within a neighborhood of $0.25S_{max}$ from the center of the top face candidate and whose sizes are between $0.6S_{max}$ and $1.4S_{max}$ are grouped into a single face candidate cluster, and the cluster is removed from the list of face candidates. The process is repeated until all face candidates in the list are clustered. For each cluster, the center of the representative face is taken as the centroid of the cluster, and its confidence score is computed as the sum of all face candidate scores in the cluster. The confidence score of the cluster can be used to verify the corresponding representative face candidate by comparing it to a given threshold. This verification scheme has also been applied in other face detection systems to reject false detections (Garcia and Delakis, 2004; Rowley et al., 1998; Sung & Poggio, 1998).

Furthermore, to estimate the size and position of the detected faces, two fine searches are performed. The face candidate is first tested at nine scales, ranging from 0.4 to 1.6 of the detected size. Then, the size of the representative face is computed as the average size of the positive detections; its score is taken as the sum of the responses of networks that give positive detections. The

position of the face is sought in a search grid of eight pixels around the center of the representative face. The location of all detected face candidates within that region are averaged to give the final location. The confidence score of the face is also computed as the sum of all positive network responses, together with its previous confidence score. Although the remaining face candidates have passed all these postprocessing steps, there may still be some multiple detections around the true face. All remaining face candidates that now have high confidence scores are stored in a list and sorted in descending order. Then a search for overlapping face candidates is performed, starting around the center of the face with the highest confidence score. Those overlapping face candidates whose centers are within a search region of size $0.5S_{max}$ are rejected. Furthermore, if the intersecting area of the overlapping face candidate is greater than $0.2(S_{max})^2$, the face candidate is also removed. The remaining representative faces are passed to the network for final verification.

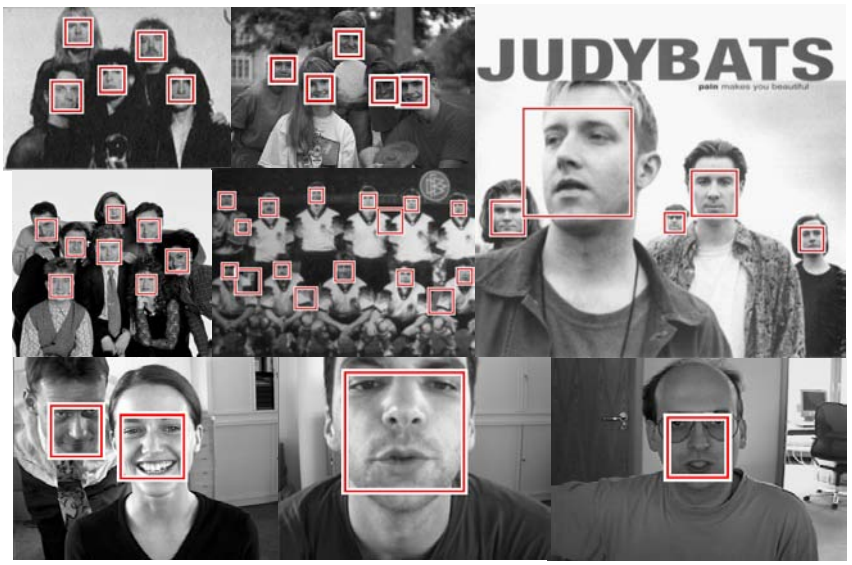
Performance of the Face Detection System

The overall face detection system (face classification and face localization) was evaluated on digital images taken from three databases: MIT-CMU, FERET, and BioID. The detection rates of the face detector are presented in Table 6, together with the number of false detections. Figure 11 shows some detected images from the MIT-CMU and BioID face databases.

Table 6. Detection performance of the face detector on BioID, FERET and MIT/CMU databases

Test set	No. of faces	Correct detection rate	False detections
BioID Database	1522	98.3 %	238
FERET Database	1762	98.4 %	25
MIT-CMU Database	507	87.7 %	138
Web Images	1871	98.0 %	252

Figure 11. Output images generated from the face detection system



Gender Recognition

Human facial attributes such as gender, age, ethnicity, and identity are important visual cues that humans use constantly for social interactions and communication. These demographic features are significant for the development of intelligent machine vision systems. Since the early 1990s, there has been a strong research effort in gender classification due to its potential applications in man-machine interaction, face recognition, passive demographic data collection, surveillance, security, animation, and so forth. Various approaches have been developed for gender recognition, including MLPs (Balci & Atalay, 2002; Golomb, Lawrence, & Sejnowski, 1991; Sun, Yuan, Bebis, & Louis, 2002), radial basis function networks (RBFs) (Brunelli & Poggio, 1992), AdaBoost classifier (Shakhnarovich, Viola, & Moghaddam, 2002; Wu, Ai, & Huang, 2003), and support vector machines (SVMs) (Moghaddam, & Yang, 2002). However, gender recognition remains a difficult perceptual task for machines because of changes in facial appearance and other extrinsic image variations (e.g., lighting conditions, image quality, face size, and pose).

The brain-inspired pattern recognition architecture was used for gender recognition. The network architecture contains three processing layers—two hidden layers and one output layer—and an input layer of size 32x32. The two hidden layers contain a number of feature maps, ranging from two to eight. The output layer consists of a single neuron used to classify the input image as a male or female. The toeplitz- and binary-connection schemes are used to connect the feature maps between layers. Both the toeplitz- and binary-connected networks were trained and evaluated on two face datasets: the first one, DB-1, is the FERET database containing 1,152 male and 610

Figure 12. Examples of face images used for training and testing the gender classifier: (a) Male face images and (b) female face images

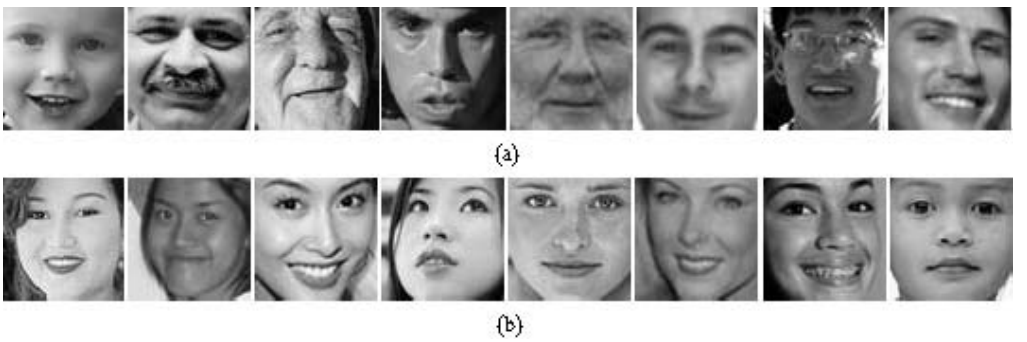


Table 7. Gender classification performance of the binary-connected networks (BCs) on DB-1 and DB-2 databases

Network Index	No. of Weights	No. of Feature Maps		Classification Rate (%)					
				DB-1 (FERET)			DB-2 (Phung)		
		L1	L2	Male	Female	Total	Male	Female	Total
BC-01	575	2	4	97.1	94.3	96.1	86.1	90.2	88.1
BC-02	862	3	6	97.1	93.6	95.9	88.4	87.8	88.1
BC-03	1149	4	8	97.6	96.4	97.2	87.0	90.3	88.7

Table 8. Gender classification performance of the toeplitz-connected networks (TCs) on DB-1 and DB-2 databases

Network Index	No. of Weights	No. of Feature Maps		Classification Rate (%)					
				DB-1 (FERET)			DB-2 (Phung)		
		L1	L2	Male	Female	Total	Male	Female	Total
TC-01	575	2	4	96.9	94.3	96.0	85.9	89.7	87.8
TC-02	862	3	6	97.1	95.4	96.5	88.7	88.9	88.8
TC-03	1149	4	8	97.6	94.3	96.4	87.0	89.9	88.4

Table 9. Comparison of performance of various gender classifiers

Face Database	Classifier	Classification Rate (%)		
		Male	Female	Total
Phung	Toeplitz-connected network	88.7	88.9	88.8
FERET	Binary-connected network	97.6	96.4	97.2
FERET	SVM Moghaddam-Yang	97.95	95.21	96.62

female face images; and the second one, DB-2, is the Phung database (Phung et al., 2005), with 4,000 male and 4,000 female face images. Figure 12 shows some examples of the male and female faces taken from the DB-1 database.

The evaluation of the networks is based on a fivefold cross-validation procedure. The classification rates of the binary- and toeplitz-connected networks are listed in Tables 7 and 8, respectively. On the FERET database, both the toeplitz- and binary-connected networks achieve excellent results with classification accuracy over 95%. Using 12 feature maps, the binary-connected network slightly outperforms the SVM-based system of Moghaddam and Yang (2002); it achieves a classification accuracy of 97.2%, (see Table 9). On DB-2, the classification rates are lower than those achieved with the FERET database, but all the networks achieve a classification rate around 88% (see Tables 7 and 8). One of the reasons for the drop in performance on DB-2 is that the data contain faces of people of different ages (ranging from children to the elderly); some of the faces are difficult to classify even for humans.

CONCLUSION

A multilayered pattern recognition architecture was presented, which is inspired by the functional and structural organization of the brain. It operates directly on two-dimensional inputs and preserves the input topographic mapping in the intermediate layers. The network architecture can easily be trained to solve various visual pattern recognition tasks using supervised learning. The artificial neuron used for the extraction of visual features is governed by the biophysical mechanism of shunting inhibition, which plays a significant role in visual information processing in the brain. The proposed architecture was applied to four real-world visual pattern recognition problems; namely, handwritten digit recognition, texture segmentation, face detection, and gender classification.

ACKNOWLEDGMENT

This work has been supported in part by the Australian Research Council.

REFERENCES

Anderson, J.S., Carandini, M., & Ferster, D. (2000). Orientation tuning of input conductance, excitation, and inhibition in cat primary visual cortex. *Journal of Neurophysiology, 84*, 909–926.

Arulampalam, G., & Bouzerdoum, A. (2003). A generalized feedforward neural network architecture for classification and regression. *Neural Networks, 16*(5-6), 561–568.

Balci, K., & Atalay, V. (2002). PCA for gender estimation: Which eigenvector contribute? *Proceedings of the Sixteenth International Conference on Pattern Recognition*, Vol. 3, 363–366.

Beare, R., & Bouzerdoum, A. (1999). Biologically inspired local motion detector architecture. *Journal of the Optical Society of America A: Optics, Image Science, and Vision, 16*(9), 2059–2068.

Borg-Graham, L.J., Monier, C., & Fregnac, Y. (1998). Visual input evokes transient and strong shunting inhibition in visual cortical neurons. *Nature, 393*(6683), 369–373.

Bouzerdoum, A. (1993). The elementary movement detection mechanism in insect vision. *Philosophical Transactions of the Royal Society of London, B-339*, 375–384.

Bouzerdoum, A. (1994). A hierarchical model for early visual processing. *Proceedings of the Human Vision, Visual Processing, and Digital Display V, Proceedings of SPIE 2179*, San Jose, California, 10–17.

Bouzerdoum, A. (1999). A new class of high-order neural networks with nonlinear decision boundaries. *Proceedings of the Sixth International Conference on Neural Information Processing, 3*, 1004–1009. Perth, Australia.

Bouzerdoum, A. (2000). Classification and function approximation using feedforward shunting inhibitory artificial neural networks. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 613–618.

Bouzerdoum, A., & Pinter, R.B. (1992). Nonlinear lateral inhibition applied to motion detection in the fly visual system. In R.B. Pinter & B. Nabet (Eds.), *Nonlinear vision* (pp. 423–450). Boca Raton, FL: CRC Press.

Bouzerdoum, A., & Pinter, R.B. (1993). Shunting inhibitory cellular neural networks: Derivation and stability analysis. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40*, 215–221.

Brodatz, P. (1966). *Texture: A photographic album for artists and designers.* New York: Dover.

Brunelli, R., & Poggio, T. (1992). Hyberbf networks for gender classification. *Proceedings of the DARPA Image Understanding Workshop*, 311–314.

Calderon, A., Roa, S., & Victorino, J. (2003). Handwritten digit recognition using convolutional neural networks and Gabor filter. *Proceedings of the International Congress on Computational Intelligence*, Medellin, Colombia.

Cheung, H.N., Bouzerdoum, A., & Newland, W. (1999). Properties of shunting inhibitory cellular neural networks for colour image enhancements. *Proceedings of the Sixth International Conference on Neural Information Processing*, Vol. 3, 1219–1223.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*, 303–314.

Fahlman, S.E. (1988). *An empirical study of learning speed in backpropagation networks* (Tech. Rep. No. CMU-CS-88-162). Pittsburgh, PA: Computer Science, Carnegie Mellon University.

Fregnac, Y., Monier, C., Chavane, F., Baudot, P., & Graham, L. (2003). Shunting inhibition, a silent step in visual computation. *Journal of Physiology, Paris, 97*, 441–451.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions Systems, Man, and Cybernetics, SMC-13*(5), 826–834.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks, 2*(3), 183–192.

Garcia, C., & Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1408–1423.

Golomb, B., Lawrence, D., & Sejnowski, T. (1991). Sexnet: A neural network identifies sex from human faces. *Advances in Neural Information Processing Systems*, 572–577.

Hjelmas, E., & Low, B.K. (2001). Face detection: A survey. *Computer Vision and Image Understanding, 83*(3), 236–274.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366.

Hubel, D.H., & Wiesel, T.N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology, 28*, 229–289.

Jain, A.K., & Karu, K. (1996). Learning texture discrimination masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(2), 195–205.

Kroner, S., & Moratz, R. (1996). Capacity of structured multilayer networks with shared weights. *Proceedings of the International Conference on Artificial Neural Networks,* Bochum, Germany, Vol. 1112, 543–550 .

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), *Connectionism in perspective*. Zurich, Switzerland: Elsevier.

LeCun, Y., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE, 86*(11), 2278–2324.

Lin, C.-T., & Shou, Y.-W. (2005). Texture classification and representation by CNN based feature extraction. *Proceedings of the 9th International Workshop on Cellular Neural Networks and Their Applications,* 210–213.

Materka, A., & Strzelecki, M. (1998). *Texture analysis methods—A review* (Tech. Rep. No. COST B11). Brussels: Institute of Electronics, Technical University of Lodz.

Mitchell, S.J., & Silver, R.A. (2003). Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron, 38*(3), 433–445.

Moghaddam, B., & Yang, M.-H. (2002). Learning gender with support faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 707–711.

Phung, S.L., Bouzerdoum, A., & Chai, D. (2005). Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(1), 148–154.

Poisson, E., Gaudin, C.V., & Lallican, P.-M. (2002). Multi-modular architecture based on convolutional neural networks for online handwritten character recognition. *Proceedings of the 9th International Conference on Neural Information Processing,* Vol. 5, 2444–2448.

Pontecorvo, C., & Bouzerdoum, A. (1997). Edge detection in multiplicative noise using the shunting inhibitory cellular neural network. *Proceedings of the International Conference on Engineering Applications of Neural Networks,* Stockholm, Sweden, 281–285.

Prescott, S.A., & Koninck, Y.D. (2003). Gain control of firing rate by shunting inhibition: Roles of synaptic noise and dendritic saturation. *Proceedings of the National Academy of Sciences of the United States of America, 100*(4), 2076–2081.

Randen, T., & Husøy, J.H. (1999). Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(4), 291-310.

Riedmiller, M. (1994). Advanced supervised learning in multilayer perceptrons—From backpropagation to adaptive learning algorithms. *Neural Networks, 5*, 265–278.

Rowley, H.A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(1), 23–38.

Shakhnarovich, G., Viola, P.A., & Moghaddam, B. (2002). A unified learning framework for real time face detection and classification. *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 14–21.

Simard, P.Y., Steinkraus, D., & Platt, J.C. (2003). Best practices for convolutional neural networks applied to visual documents analysis. *Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2*, 958–962.

Sun, Z., Yuan, X., Bebis, G., & Louis, S. (2002). Neural-network-based gender classification using genetic eigen-feature extraction. *Proceedings of the International Joint Conference on Neural Networks, 3*, 2433–2438.

Sung, K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence, 20*(1), 31–59.

Tivive, F.H.C., & Bouzerdoum, A. (2006). Application of SICoNNets to handwritten digit recognition. *International Journal of Computational Intelligence and Applications, 6*(1), 45–59.

Tollenaere, T. (1990). SuperSAB: Fast adaptive BP with good scaling properties. *Neural Networks, 3*, 561–573.

Tuceryan, M., & Jain, A.K. (1998). Texture analysis. In C.H. Chen, L.F. Pau, & P.S.P. Wang (Eds.), *The handbook of pattern recognition and computer vision* (2nd ed.) (pp. 207–248). Singapore: World Scientific.

Vida, I., Bartos, M., & Jonas, P. (2006). Shunting inhibition improves robustness of gamma oscillations in hippocampal interneuron networks by homogenizing firing rates. *Neuron, 49*(1), 107–117.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, 511–518.

Wu, B., Ai, H., & Huang, C. (2003). LUT-based adaboost for gender classification. *Proceedings of the Fourth International Conference on Audio- and Video-Based Biometric Person Authentication*, 104–110.

Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(1), 34–58.

ENDNOTES

1 The MNIST database was obtained from www.yann.lecun.com/exdb/mnist (access date: 2005).

2 The texture mosaics created by Randen and Husøy (1999) are available at www.ux.uis.no/~tranden/ (access date: 2006).

Chapter XII
Significance of Logic Synthesis in FPGA-Based Design of Image and Signal Processing Systems

Mariusz Rawski
Warsaw University of Technology, Poland

Henry Selvaraj
University of Nevada, USA

Bogdan J. Falkowski
Nanyang Technological University, Singapore

Tadeusz Łuba
Warsaw University of Technology, Poland

ABSTRACT

This chapter, taking FIR filters as an example, presents the discussion on efficiency of different implementation methodologies of DSP algorithms targeting modern FPGA architectures. Nowadays, programmable technology provides the possibility to implement digital systems with the use of specialized embedded DSP blocks. However, this technology gives the designer the possibility to increase efficiency of designed systems by exploitation of parallelisms of implemented algorithms. Moreover, it is possible to apply special techniques, such as distributed arithmetic (DA). Since in this approach, general-purpose multipliers are replaced by combinational LUT blocks, it is possible to construct digital filters of very high performance. Additionally, application of the functional decomposition-based method to LUT blocks optimization, and mapping has been investigated. The chapter presents results of the comparison of various design approaches in these areas.

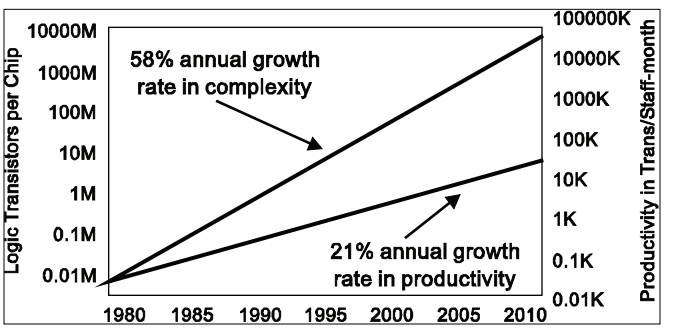
INTRODUCTION

The pattern recognition research field aims to design methods that allow recognition of patterns in data. It has important application in image analysis, character recognition, speech analysis, and many others. A pattern recognition system is composed of sensors gathering observations that have to be classified, a feature extraction part that provides specific information from gathered observation, and a classification mechanism that classifies observation on the basis of extracted features. Feature extraction methods are responsible for reducing the resources required to describe observation accurately. In the case of image analysis, character recognition, or speech analysis, various digital signal-processing (DSP) algorithms are used to detect desired features of digitalized image or speech signal. Efficient implementation of feature extraction-based DSP methods requires specific hardware solutions.

The commercial success of hardware implementations of image processing systems is due in large part to revolutionary development in microelectronic technologies. By taking advantage of the opportunities provided by modern microelectronic technology, we are in a position to build very complex digital circuits and systems at relatively low cost. There is a large variety of logic building blocks that can be exploited. The library of elements contains various types of gates, a lot of complex gates that can be generated in (semi-) custom CMOS design, and the field programmable logic families that include various types of (C)PLDs and FPGAs. The other no less important factors of the success are the automation of the design process and hardware description languages. Modern design tools have enabled us to move beyond putting together digital components in a schematic entry package to start writing code in an HDL specification. However, the opportunities created by modern microelectronic technology are not fully exploited because of weaknesses in traditional logic design methods. According to the International Technology Roadmap for Semiconductors (1997), the annual growth rate in design complexity is equal to 58%, while the annual growth rate in productivity is only 21% (Figure 1). This means that the number of logic gates available in modern devices grows faster than the ability to design them meaningfully. New methods are required to aid design process in a way that possibilities offered by modern microelectronics are utilized in the highest possible degree.

In recent years, digital filtering has been recognized as a primary digital signal processing (DSP) operation. With advances in technology, digital filters are rapidly replacing analogue filters, which were implemented with RLC components. Digital filters are used to modify attributes of signal in the time or frequency domain through a process

Figure 1. Difference in growth of device complexity and productivity



called linear convolution. Traditionally, digital signal filtering algorithms are being implemented using general-purpose programmable DSP chips. Alternatively, for high-performance applications, special-purpose fixed function DSP chipsets and application-specific integrated circuits (ASICs) are used. Typical DSP devices are based on the concept of RISC processors with an architecture that consists of fast array multipliers. In spite of using pipeline architecture, the speed of such implementation is limited by the speed of array multiplier. Digital filters are implemented in such devices as multiply-accumulate (MAC) algorithms (Lapsley, Bier, Shoham & Lee, 1997; Lee, 1988, 1989). However, the technological advancements in field programmable gate arrays (FPGAs) in the past decade have opened new paths for DSP design engineers.

Digital filtering plays an extremely important role in many signal and image processing algorithms. An excellent example is wavelet transform, which has gained much attention in recent years. Discrete wavelet transform (DWT) is one of the useful and efficient signal and image decomposition methods with many interesting properties (Daubechies, 1992; Falkowski, 2004; Falkowski & Chang, 1997; Rao & Bopardikar, 1998; Rioul & Vetterli, 1991). This transformation, similar to the Fourier transform, can provide information about frequency contents of signals. However, unlike Fourier transform, this approach is more natural and fruitful when applied to nonstationary signals, like speech, signal, and images. The flexibility offered by discrete wavelet transform allows researchers to develop and find the right wavelet filters for their particular application. For example, in fingerprints compression, a particular set of bio-orthogonal filters—Daubechies bio-orthogonal spine wavelet filters—is found to be very effective (Brislawn, Bradley, Onyshczak & Hopper, 1996). The computational complexity of the discrete wavelet transform is very high. Hence, efficient hardware implementation is required to achieve very good real-time performance. Application of the DWT requires convolution of the signal with the wavelet and scaling functions. Efficient hardware implementation of convolution is performed as a finite impulse response (FIR) filter. Two filters are used to evaluate a DWT: a high-pass and a low-pass filter, with the filter coefficients derived from the wavelet basis function.

Progress in the development of programmable architectures observed in recent years has resulted in digital devices that allow building very complex digital circuits and systems at relatively low cost in a single programmable structure. FPGAs are an array of programmable logic cells interconnected by a matrix of wires and programmable switches. Each cell performs a simple logic function defined by a designer's program. An FPGA has a large number (64 to more than 300,000) of these cells available to use as building blocks in complex digital circuits. The ability to manipulate the logic at the gate level means that a designer can construct a custom processor to efficiently implement the desired function. FPGA manufacturers have for years been extending their chips' ability to implement digital-signal processing efficiently; for example, by introducing low-latency carry-chain-routing lines that sped addition and subtraction operations spanning multiple logic blocks. Such a mechanism is relatively efficient when implementing addition and subtraction operations. However, it is not optimal in cost, performance, and power for multiplication and division functions. As a result, Altera (with Stratix), QuickLogic (with QuickDSP, now renamed Eclipse Plus), and Xilinx (with Virtex-II and Virtex-II Pro) embedded in their chips dedicated multiplier function blocks. Altera moved even further along the integration path, providing fully functional MAC blocks called the DSP blocks.

Programmable technology makes it possible to increase the performance of a digital system by implementing multiple, parallel modules in one chip. This technology allows also the application of special techniques such as distributed

arithmetic (DA) (Croisier, Esteban, Levilion & Rizo, 1973; Meyer-Baese, 2004; Peled & Liu, 1974). DA technique is extensively used in computing the sum of product in filters with constant coefficients. In such a case, partial product term becomes a multiplication with a constant (i.e., scaling). DA approach significantly increases the performance of an implemented filter by removing general-purpose multipliers and introducing combinational blocks that implement the scaling. These blocks have to be efficiently mapped onto FPGA's logic cells. This can be done with the use of such advanced synthesis methods as functional decomposition (Rawski, Tomaszewicz, & Łuba, 2004; Rawski, Tomaszewicz, Selvaraj, & Łuba, 2005; Sasao, Iguchi, & Suzuki, 2005).

In the case of applications targeting FPGA structures based on look-up tables (LUTs), the influence of advanced logic synthesis procedures on the quality of hardware implementation of signal and information processing systems is especially important. Direct cause of such a situation is the imperfection of technology mapping methods that are widely used at present, such as minimization and factorization of Boolean function, which are traditionally adapted to be used for structures based on standard cells. These methods transform Boolean formulas from a sum-of-products form into a multilevel, highly factorized form that is then mapped into LUT cells. This process is at variance with the nature of the LUT cell, which from the logic synthesis point of view is able to implement any logic function of limited input variables. For this reason, for the case of implementation targeting FPGA structure, decomposition is a much more efficient method. Decomposition allows synthesizing the Boolean function into a multilevel structure that is built of components, each of which is in the form of the LUT logic block specified by truth tables. Efficiency of functional decomposition has been proved in many theoretical papers (Brzozowski & Łuba, 2003; Chang, Marek-Sadowska & Hwang, 1996; Rawski, Jóźwiak & Łuba, 2001; Scholl, 2001). However, there are relatively few papers in which functional decomposition procedures were compared with analogous synthesis methods used in commercial design tools. The reason behind such a situation is the lack of appropriate interface software that would allow a transforming description of project structure obtained outside a commercial design system into a description compatible with its rules. Moreover, the computation complexity of functional decomposition procedures makes it difficult to construct efficient automatic synthesis procedures. These difficulties have been eliminated at least partially in so-called balanced decomposition (Łuba, Selvaraj, Nowicka & Kraśniewski, 1995; Nowicka, Łuba & Rawski, 1999).

BASIC THEORY

In this chapter, only such information necessary for an understanding of this chapter is reviewed. More detailed description of functional decomposition based on partition calculus can be found in Brzozowski and Łuba (2003).

Cube Representation of Boolean Functions

A Boolean function can be specified using the concept of cubes (e.g., input terms, patterns) representing some specific subsets of minterms. In a minterm, each input variable position has a well-specified value. In a cube, positions of some input variables can remain unspecified, and they represent "any value" or "don't care" (–). A cube may be interpreted as a p-dimensional subspace of the n-dimensional Boolean space or as a product of $n–p$ variables in Boolean algebra (p denotes the number of components that are "–"). Boolean functions are typically represented by truth tables. A truth table description of a function using minterms requires 2^n rows for a function of n variables. For function from Table 1, a truth

table with $2^6 = 64$ rows would be required. Since a cube represents a set of minterms, application of cubes allows for much more compact description in comparison with minterm representation. For example, cube 0101–0 from row 1 of the truth table from Table 1 represents a set of two minterms {010100, 010110}.

For pairs of cubes and for a certain input subset *B*, we define the compatibility relation COM as follows: each two cubes *S* and *T* are compatible (i.e., $S, T \in \mathrm{COM}(B)$) if and only if $x(S) \sim x(T)$ for every $x \subseteq B$. The compatibility relation ~ on {0, –, 1) is defined as follows: 0 ~ 0, – ~ –, 1 ~ 1, 0 ~ –, 1 ~ –, – ~ 0, – ~ 1, but the pairs (1, 0) and (0, 1) are not related by ~. The compatibility relation on cubes is reflexive and symmetric, but not necessarily transitive. In general, it generates a "partition" with nondisjoint blocks on the set of cubes representing a certain Boolean function *F*. The cubes contained in a block of the "partition" are all compatible with each other.

"Partitions" with nondisjoint blocks are referred to as blankets (Brzozowski & Łuba, 2003). The concept of blanket is a simple extension of ordinary partition, and typical operations on blankets are strictly analogous to those used in ordinary partition algebra.

Representation and Analysis of Boolean Functions with Blankets

A blanket on a set *S* is such a collection of (not necessarily disjoint) subsets B_i of *S*, called blocks, that:

$$\bigcup_i B_i = S$$

The product of two blankets β_1 and β_2 is defined as follows:

$$\beta_1 \bullet \beta_2 = \{ B_i \cap B_j \mid B_i \in \beta_1 \text{ and } B_j \in \beta_2 \}$$

For two blankets we write $\beta_1 \leq \beta_2$ if and only if for each B_i in β_1 there exists a B_j in β_2 such that $B_i \subseteq B_j$. The relation ≤ is reflexive and transitive.

Example 1: Blanket-Based Representation of Boolean Functions

For function *F* from Table 1, the blankets induced by particular input and output variables on the set of function *F*'s input patterns (cubes) are as follows:

$\beta_{x1} = \{\overline{1, 2, 3, 4, 5, 6, 8, 9}; \overline{3, 6, 7, 9, 10}\}$,
$\beta_{x2} = \{\overline{5, 6, 7, 8, 10}; \overline{1, 2, 3, 4, 6, 7, 9, 10}\}$,
$\beta_{x3} = \{\overline{1, 2, 3, 4, 8, 9}; \overline{5, 6, 7, 8, 10}\}$,
$\beta_{x4} = \{\overline{2, 3, 5, 8, 9, 10}; \overline{1, 2, 4, 5, 6, 7, 8}\}$,
$\beta_{x5} = \{\overline{1, 2, 3, 5, 6, 7, 8, 10}; \overline{1, 4, 5, 6, 8, 9, 10}\}$,
$\beta_{x6} = \{\overline{1, 2, 3, 4, 7, 8, 9, 10}; \overline{3, 4, 5, 6, 7, 9, 10}\}$,
$\beta_{y1} = \{\overline{1, 2, 3, 4, 5, 6, 7}; \overline{8, 9, 10}\}$,

The product of two blankets β_1 and β_2:

$\beta_{x2x4} = \beta_{x2} \bullet \beta_{x4} =$
$\{\overline{5, 8, 10}; \overline{5, 6, 7, 8}; \overline{2, 3, 9, 10}; \overline{1, 2, 4, 6, 7}\}$,
$\beta_{x2x4} \leq \beta_{x2}$.

Information on the input patterns of a certain function *F* is delivered by the function's inputs and

Table 1. Boolean function $F(x_1, x_2, x_3, x_4, x_5, x_6)$

	x_1	x_2	x_3	x_4	x_5	x_6	y_1
1	0	1	0	1	–	0	0
2	0	1	0	–	0	0	0
3	–	1	0	0	0	–	0
4	0	1	0	1	1	–	0
5	0	0	1	–	–	1	0
6	–	–	1	1	–	1	0
7	1	–	1	1	0	–	0
8	0	0	–	–	–	0	1
9	–	1	0	0	1	–	1
10	1	–	1	0	–	–	1

used by its outputs with respect to the blocks of the input and output blankets. Knowing the block of a certain blanket, one is able to distinguish the elements of this block from all other elements, but is unable to distinguish between elements of the given block. In this way, information in various points and streams of discrete information systems can be modeled using blankets.

Serial Decomposition

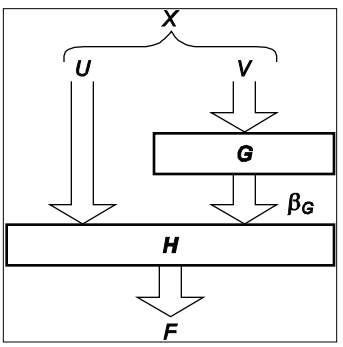
The set X of a function's input variable is partitioned into two subsets: *free variables U* and *bound variables V*, such that $U \cup V = X$. Assume that the input variables $x_1,...,x_n$ have been relabeled in such way that:

$U = \{x_1,...,x_r\}$ and
$V = \{x_{m-s+1},...,x_n\}$.

Consequently, for an n-tuple x, the first r components are denoted by x^U, and the last s components, by x^V.

Let F be a Boolean function, with $n > 0$ inputs and $m > 0$ outputs, and let (U, V) be as previously indicated. Assume that F is specified by a set $\mathcal{F}$ of the function's cubes. Let G be a function with s inputs and p outputs, and let H be a function with $r + p$ inputs and m outputs. The pair (G, H) represents a serial decomposition of F with respect to (U, V), if for every minterm b relevant to F, $G(b^V)$ is defined, $G(b^V) \in \{0, 1\}^p$, and $F(b) = H(b^U, G(b^V))$. G and H are called blocks of the decomposition (Figure 2).

Figure 2. Schematic representation of the serial decomposition



Theorem 1: Existence of Serial Decomposition (Brzozowski & Łuba, 2003)

Let β_V, β_U, and β_F be blankets induced on the function's F input cubes by the input subsets V and U, and outputs of F, respectively.

If there exists a blanket β_G on the set of function F's input cubes such that $\beta_V \leq \beta_G$, and $\beta_U \bullet \beta_G \leq \beta_F$, then F has a serial decomposition with respect to (U, V).

Proof of Theorem 1 can be found in Brzozowski and Łuba (2003).

As follows from Theorem 1, the main task in constructing a serial decomposition of a function F with given sets U and V is to find a blanket β_G that satisfies the condition of the theorem. Since β_G must be $\geq \beta_V$, it is constructed by merging blocks of β_V as much as possible.

Two blocks B_i and B_j of blanket β_V are compatible (merge able), if blanket γ_{ij} obtained from blanket β_V by merging B_i and B_j into a single block satisfies the second condition of Theorem 1; that is, if $\beta_U \bullet \gamma_{ij} \leq \beta_F$. Otherwise blocks B_i and B_j are incompatible (unmergeable). A subset δ of blocks of the blanket β_V is a compatible class of blocks if the blocks in δ are pairwise compatible. A compatible class is maximal if it is not contained in any other compatible class.

From the computational point of view, finding maximal compatible classes is equivalent to finding maximal cliques in a graph $\Gamma = (N, E)$, where the set N of nodes is the set of blocks of β_V and set E of edges is formed by set of compatible pairs.

The next step in the calculation of β_G is the selection of a set of maximal classes, with minimal cardinality, that covers all the blocks of β_V. The minimal cardinality ensures that the number of

blocks of β_G, and hence the number of outputs of the function *G*, is as small as possible.

In certain heuristic strategies, both procedures (finding maximal compatible classes and then finding the minimal cover) can be reduced to the graph coloring problem.

Calculating β_G corresponds to finding the minimal number *k* of colors for graph $\Gamma = (N, E)$.

Example 2

For the function from Table 1 specified by a set $\mathcal{F}$ of cubes numbered 1 through 10, consider a serial decomposition with $U = \{x_2, x_4, x_5\}$ and $V = \{x_1, x_3, x_6\}$.

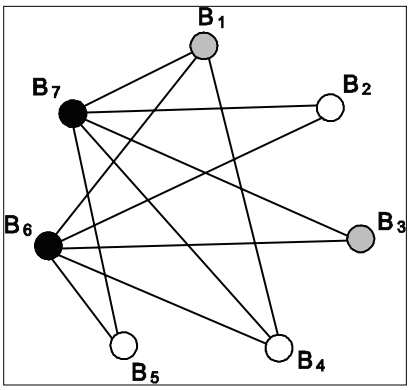
We find:

$$\beta_U = \beta_{x2\,x4\,x5} = \beta_{x2} \bullet \beta_{x4} \bullet \beta_{x5} =$$
$$\{\overline{5, 8, 10};\ \overline{5, 6, 7, 8};\ \overline{2, 3, 10};\ \overline{1, 2, 6, 7};\ \overline{9, 10};\ \overline{1, 4, 6}\},$$
$$\beta_V = \beta_{x1\,x3\,x6} = \beta_{x1} \bullet \beta_{x3} \bullet \beta_{x6} =$$
$$\{\overline{1, 2, 3, 4, 8, 9};\ \overline{3, 4, 9};\ \overline{8};\ \overline{5, 6};\ \overline{3, 9};\ \overline{7, 10};\ \overline{6, 7, 10}\},$$
$$\beta_F = \beta_{y1} = \{\overline{1, 2, 3, 4, 5, 6, 7};\ \overline{8, 9, 10}\},$$

For:

$$\beta_V = \{\overline{1, 2, 3, 4, 8, 9}^{B_1};\ \overline{3, 4, 9}^{B_2};\ \overline{8}^{B_3};\ \overline{5, 6}^{B_4};\ \overline{3, 9}^{B_5};\ \overline{7, 10}^{B_6};\ \overline{6, 7, 10}^{B_7}\}$$

Figure 3. Incompatibility graph of β_V's blocks



the following are the unmergeable pairs: (B_1, B_4), (B_1, B_6), (B_1, B_7), (B_2, B_6), (B_2, B_7), (B_3, B_4), (B_3, B_6), (B_3, B_7), (B_4, B_6), (B_4, B_7), (B_5, B_6), and (B_5, B_7). Using the graph coloring procedure, we find that three colors are needed here (Figure 3).

Nodes B_1, B_3 are assigned one color, nodes B_2, B_4, B_5 are assigned a second color, and a third color is assigned for nodes B_6, B_7. The sets of nodes assigned to different colors form the blocks of β_G.

$$\beta_G = \{\overline{1, 2, 3, 4, 8, 9};\ \overline{3, 4, 5, 6, 9};\ \overline{6, 7, 10}\}$$

It is easily verified that β_G satisfies the condition of Theorem 1. Thus, function *F* has a serial decomposition with respect to (*U*, *V*).

Since β_G has 3 blocks, to encode blocks of this blanket, two encoding bits g_1 and g_2 have to be used. Let us assume that we use the encoding:

$$\beta_G = \{\overline{1, 2, 3, 4, 8, 9}^{00};\ \overline{3, 4, 5, 6, 9}^{01};\ \overline{6, 7, 10}^{10}\}.$$

To define a function *G* by a set of cubes, we calculate all the cubes, $r(B_i)$, assigned to each block B_i of β_V. The relationship between blocks of β_V and their cube representatives, $r(B_i)$, relies on containment of block B_i in blocks of β_{xj} from $x_j \in V$.

Denoting blocks of β_V from Example 2 as B_1 through B_7, we have $r(B_1) = 000$. This is because $B_1 = \{1, 2, 3, 4, 8, 9\}$ is included in the first blocks of β_{x1}, β_{x3} and β_{x6}. For $B_2 = \{3, 4, 9\}$, we have: B_2 is included in the first block of β_{x1}, in the first block β_{x3} and in both blocks of β_{x6}. Hence, $r(B_2) = 00–$. Similarly, $r(B_3) = 0–0$, $r(B_4) = 011$, $r(B_5) = –0–$, $r(B_6) = 11–$, $r(B_7) = 111$.

Finally, the value of function *G* is obtained on the basis of containment of blocks B_i in blocks of β_G. Block B_1={1, 2, 3, 4, 8, 9} of blanket β_V is contained in block β_G that has been encoded with 00. Since $r(B_1) = 000$, we have $G(r(B_1)) = G(x_1 = 0, x_3 = 0, x_6 = 0) = 00$. Similarly, $G(r(B_3)) = 00$, $G(r(B_4)) = 01$, $G(r(B_6)) = 10$ and $G(r(B_7)) = 10$. However, block $B_2 = \{3, 4, 9\}$ is contained in two blocks of β_G (one

encoded "00" and the second "01"). The representative "00–" of this block has nonempty product with representative "000" of B_1 and representative "0–0" of B_3, which was assigned output "00." To avoid conflicts, we must subtract cubes "000" and "0–0" from cube "00–." The result is cube "001" that may be assigned output "01." The same applies to block B_5. The representative "–0–" of this block has nonempty product with representative of B_1 and B_3, which was assigned output "00." We must subtract cubes "000" and "0–0" from cube "0–0," and the result in the form of cube "10–" may be assigned output "01".

Truth table of function *G* is presented in Table 2a. To compute the cubes for function H, we consider each block of the product $\beta_U \bullet \beta_G$. Their representatives are calculated in the same fashion. Finally, the outputs of *H* are calculated with respect to β_F (Table 2b).

The process of functional decomposition consists of the following steps:

- Selection of an appropriate input support *V* for block *G* (input variable partitioning)
- Calculation of the blankets β_U, β_V and β_F
- Construction of an appropriate multiblock blanket β_G (corresponds to the construction of the multivalued function of block *G*)
- Creation of the binary functions *H* and *G* by representing the multiblock blanket β_G as the product of a number of certain two-block blankets (equivalent to encoding the multivalued function of block *G* defined by blanket β_G with a number of binary output variables)

In a multilevel decomposition, this process is applied to functions *H* and *G* repetitively, until each block in the obtained network in this way can be mapped directly to a logic block of a specific implementation structure (Łuba & Selvaraj, 1995).

The selection of an appropriate input variable partitioning is the main problem in functional decomposition (Rawski, Jóźwiak & Łuba, 1999a; Rawski, Selvaraj & Morawiecki, 2004). The choice of sets *U* and *V* from set *X* determines the construction of an appropriate blanket β_G, which satisfies Theorem 1. The existence of such a blanket β_G implies the existence of a serial decomposition. Blankets β_V, β_G, $\beta_U \bullet \beta_G$, and β_F constitute the basis for the construction of subfunctions *H* and *G* in serial decomposition. In other words, knowing β_V, β_U, and β_F, and having β_G, one can construct particular subfunctions *G* and *H*.

Table 2a. Function G of the serial decomposition

	x_1	x_3	x_6	g_1	g_2
1	0	0	0	0	0
2	0	0	1	0	1
3	0	–	0	0	0
4	0	1	1	0	1
5	1	0	–	0	1
6	1	1	–	1	0
7	1	1	1	1	0

Table 2b. Function H of the serial decomposition

	x_2	x_4	x_5	g_1	g_2	y_1
1	1	1	–	0	0	0
2	1	–	0	0	0	0
3	1	0	0	0	0	0
4	1	0	0	0	1	0
5	1	1	1	0	0	0
6	1	1	1	0	1	0
7	0	1	1	0	1	0
8	–	1	–	0	1	0
9	–	1	–	1	0	0
10	–	1	0	1	0	0
11	0	–	–	0	0	1
12	1	0	1	0	0	1
13	1	0	1	0	1	1
14	–	0	–	1	0	1

The input variables of block *G* and their corresponding blankets, and the output blanket β_G of block *G* define together the multivalued function of block *G*. The structure of β_G obviously influences the shape of the subfunctions *G* and *H* (Figure 2). Blanket β_G determines the output values of function *G*. Each value of this multivalued function corresponds to a certain block of the blanket β_G. Considering the number of values of the multivalued function of a certain subsystem in decomposition is therefore equivalent to considering the number of blocks in blanket β_G of this subsystem. A minimum of $\lceil \log_2 q \rceil$ binary variables is required for encoding *q* values. Thus, if *q* denotes the number of blocks in β_G, then the minimum required number of binary outputs from *G* is equal to $k = \lceil \log_2 q \rceil$.

Since function *H* is constructed by substituting in the truth table of function *F* the patterns of values of the primary input variables from set *V* (bound variables) with the corresponding values of function *G*, it is obvious that the choice of β_G influences the subfunction *H*. The outputs of *G* constitute a part of the input support for block *H*. Thus, the size of block *G* and the size of block *H* both grow with the number of blocks in blanket β_G. The minimum possible number of blocks in β_G strongly depends on the input support chosen for block *G*, because β_G is computed by merging some blocks of β_V, this being the blanket induced by the chosen support.

Function *H* is decomposed in the successive steps of the multilevel synthesis process. This is why blanket β_G has a direct influence on the next steps of the process. The structure of the blanket β_G determines the difficulty of the successive decomposition steps and influences the final result of the synthesis process (characterized by the number of logic blocks and number of logic levels). The number of blocks in blanket β_G is the most decisive parameter. The strong correlation of the number of blanket β_G's blocks with the decomposition's quality has been shown in Rawski, Jóźwiak, and Łuba (1999b), and this number can be used as a criterion for testing individual solutions.

In multilevel logic synthesis methods, the serial decomposition process is applied recursively to functions *H* and *G* obtained in the previous synthesis steps until each block of the resulting net can be mapped directly to a single logic block of a specific implementation structure (Łuba, 1995; Łuba & Selvaraj, 1995). In the case of look-up table FPGAs, the multilevel decomposition process ends when each block of the resulting net can be mapped directly into a configurable logic block (CLB) of a specific size (typically the CLB size is from 4 to 6 inputs and 1 or 2 outputs). Although algorithms of multilevel logic synthesis can also use parallel decomposition in order to assist the serial decomposition (Łuba et al., 1995), the final results of the synthesis process strongly depend on the quality of the serial decomposition.

Parallel Decomposition

Consider a multiple-output function *F*. Assume that *F* has to be decomposed into two components, *G* and *H*, with disjoint sets Y_G and Y_H of output variables. This problem occurs, for example, when we want to implement a large function using components with a limited number of outputs. Note that such a parallel decomposition can also alleviate the problem of an excessive number of inputs of *F*. This is because for typical functions, most outputs do not depend on all input variables. Therefore, the set X_G of input variables on which the outputs of Y_G depend may be smaller than *X*. Similarly, the set X_H of input variables on which the outputs of Y_H depend may be smaller than *X*. As a result, components *G* and *H* have not only fewer outputs but also fewer inputs than *F*. The exact formulation of the parallel decomposition problem depends on the constraints imposed by the implementation style. One possibility is to find sets Y_G and Y_H, such that the combined cardinality of X_G and X_H is minimal. Partitioning the set of outputs into only two disjoint subsets is not an important limitation of the method, because the procedure can be applied again for components *G* and *H*.

Example 3

Consider the multiple-output function given in Table 3. The minimal sets of input variables on which each output of *F* depends are:

y_1: $\{x_1, x_2, x_6\}$
y_2: $\{x_3, x_4\}$
y_3: $\{x_1, x_2, x_4, x_5, x_9\}$, $\{x_1, x_2, x_4, x_6, x_9\}$
y_4: $\{x_1, x_2, x_3, x_4, x_7\}$
y_5: $\{x_1, x_2, x_4\}$
y_6: $\{x_1, x_2, x_6, x_9\}$

An optimal two-block decomposition, minimizing the card X_G + card X_H (where card X is the cardinality of X), is $Y_G = \{y_2, y_4, y_5\}$ and $Y_H = \{y_1, y_3, y_6\}$, with $X_G = \{x_1, x_2, x_3, x_4, x_7\}$ and $X_H = \{x_1, x_2, x_4, x_6, x_9\}$. The truth tables for components *G* and *H* are shown in Table 4.

The algorithm itself is general in the sense that the function to be parallel decomposed can be specified in compact cube notation. Calculation of the minimal sets of input variables for each individual output can be a complex task. Thus, in practical implementation, heuristic algorithms are used, which support calculations with the help of so-called indiscernible variables.

Table 3. Function F

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	y_1	y_2	y_3	y_4	y_5	y_6
1	0	0	0	1	1	1	0	0	0	0	0	0	0	–	0
2	1	0	1	0	0	0	0	0	0	0	0	–	1	0	1
3	1	0	1	1	1	0	0	0	0	0	1	1	0	1	1
4	1	1	1	1	0	1	0	0	0	0	1	1	1	1	0
5	1	0	1	0	1	0	0	0	0	0	0	0	–	0	1
6	0	0	1	1	1	0	0	0	0	1	1	0	1	0	0
7	1	1	1	0	0	0	0	0	0	1	0	–	0	1	0
8	1	0	1	1	0	1	0	0	0	1	1	0	0	–	1
9	1	0	1	1	0	1	1	0	0	–	1	0	1	–	1
10	1	1	1	0	0	0	0	1	0	1	0	1	0	1	–
11	0	0	0	1	1	1	0	0	1	0	0	1	0	–	1
12	0	0	0	1	1	0	0	0	1	–	–	1	0	0	0

Table 4a. Function G of parallel decomposition

	x_1	x_2	x_3	x_4	x_7	y_2	y_4	y_5
1	0	0	0	1	0	0	0	0
2	1	0	1	0	0	0	1	0
3	1	0	1	1	0	1	0	1
4	1	1	1	1	0	1	1	1
5	0	0	1	1	0	1	1	0
6	1	1	1	0	0	0	0	1
7	1	0	1	1	1	–	1	–

Table 4b. Function H of parallel decomposition

	x_1	x_2	x_4	x_6	x_9	y_1	y_3	y_6
1	0	0	1	1	0	0	0	0
2	1	0	0	0	0	0	0	1
3	1	0	1	0	0	0	1	1
4	1	1	1	1	0	0	1	0
5	0	0	1	0	0	1	0	0
6	1	1	0	0	0	1	1	0
7	1	0	1	1	0	1	0	1
8	0	0	1	1	1	0	1	1
9	0	0	1	0	1	–	1	0

Balanced Functional Decomposition

The balanced decomposition is an iterative process in which, at each step, either parallel or serial decomposition of a selected component is performed. The process is carried out until all resulting subfunctions are small enough to fit blocks with a given number of input variables.

Example 4

The influence of the parallel decomposition on the final result of the FPGA-based mapping process will be explained with the function *F* given in Table 5, for which cells with four inputs and one output are assumed (this is the size of Altera's FLEX FPGAs).

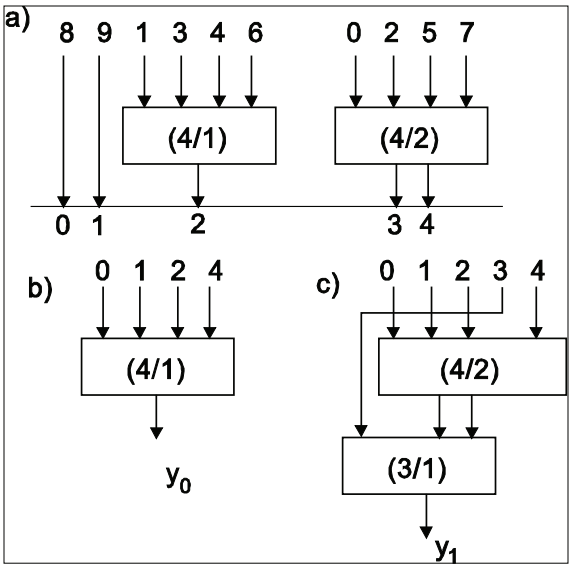
As *F* is a 10-input, two-output function, in the first step of the decomposition, particularly in automated mode, serial decomposition is performed. The algorithm extracts function *g* with inputs numbered 1, 3, 4, and 6; thus, the next step deals with seven-input function *H*, for which again serial decomposition is assumed, now resulting in block *G*, with four inputs and two outputs (implemented by two cells). It is worth noting that the obtained block *G* takes as its inputs variables denoted 0, 2, 5, and 7, which, fortunately, belong to primary variables, and therefore the number of levels is not increased in this step as it is shown in Figure 4a. In the next step, we apply parallel decomposition. Parallel decomposition generates two components, both with one output but four and five inputs, respectively. The first one forms a cell (Figure 4b). The second component is subject to two-stage serial decomposition as shown in Figure 4c. The obtained network can be built of seven (four to one) cells, where the number of levels in the critical path is three.

The same function decomposed with parallel decomposition, the first step shown in Figure 5, leads to a completely different structure. Parallel decomposition applied directly to function *F* generates two components, both with six inputs and one output. Each of them is subject to two-stage serial decomposition. For the first component, a disjoint serial decomposition with four inputs and one output can be applied (Figure 5a). The second component can be decomposed serially as well; however, with the number of outputs of the extracted block, *G* equals two. Therefore,

Table 5.

```
type fr
.i 10
.o  2
.p 25
0101000000  00
1110100100  00
0010110000  10
0101001000  10
1110101101  01
0100010101  01
1100010001  00
0011101110  01
0001001110  01
0110000110  01
1110110010  10
0111100000  00
0100011011  00
0010111010  01
0110001110  00
0110110111  11
0001001011  11
1110001110  10
0011001011  10
0010011010  01
1010110010  00
0100110101  11
0001111010  00
1101100100  10
1001110111  11
.e
```

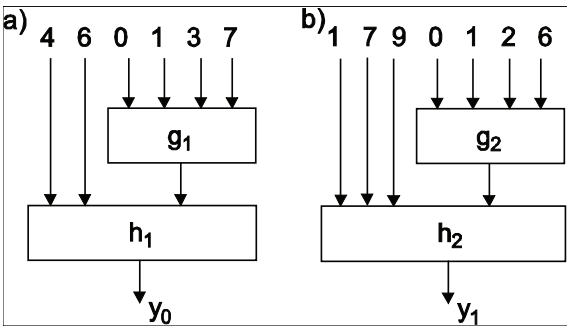
Figure 4. Decomposition of function F obtained with a strategy, where serial decomposition is performed at first



to minimize the total number of components, a nondisjoint decomposition strategy can be applied. The truth tables of the decomposed functions g_1, h_1, g_2, h_2, are shown in Table 6. The columns in the tables denote variables in the order shown in Figure 5; for example, the first left-hand side column in Table 6b denotes variable numbered 4, the second variable numbered 6, and the third denotes variable g_1. Such a considerable impact on the structure results from the fact that the parallel decomposition simultaneously reduces the number of inputs to both the resulting components, leading to an additional improvement in the final representation.

The idea of intertwining parallel and serial decomposition has been implemented in a program called DEMAIN. DEMAIN has two modes: automatic and interactive. It can also be used for the reduction of the number of inputs of a function when an output depends on only a subset of the inputs. From this point of view, DEMAIN is a tool specially dedicated to FPGA-oriented technology mapping.

Figure 5. Decomposition of function F with a strategy, where parallel decomposition is performed at first



Given a function F with n inputs and m outputs, and a logic cell (LC) with C_{in} inputs and C_{out} outputs, a decomposition process is carried out by the following steps:

1. If $n \leq m$, use parallel decomposition. Continue iteratively for each of the obtained components.
2. If $n > m$, try disjoint serial decomposition with the number of block G inputs equal to the number C_{in} of LC inputs, and the number of block G outputs equal to the number C_{out} of LC outputs. If such a serial decomposition is found, find the corresponding H. Continue iteratively with $F = H$. Otherwise, try to find G with fewer inputs than C_{in} and/or fewer outputs than C_{out}. If such a G is found, find H. Continue iteratively with $F = H$. In case a G that fits in one cell cannot be found, try a larger G. This step is repeated until decomposition with a function G larger than the cell exists. Find H and continue with $F = H$. Function G will have to be decomposed later.

The decomposition is carried out until all resulting subfunctions are small enough to fit into logic cells available in the assumed implementation technology.

Table 6.

a) function g_1	b) function h_1
0110 1	-01 0
1101 1	011 1
1000 1	111 0
0010 1	100 1
0000 0	0-0 0
0101 0	110 0
1100 0	
0100 0	
0011 0	
1011 0	
1111 0	
c) function g_2	d) function h_2
0110 1	10-1 0
0011 1	-101 1
0100 1	-111 1
1000 1	0011 0
0101 1	0001 1
1100 0	1-00 0
0010 0	0000 0
1010 0	1110 1
1110 0	1010 0
0001 0	0100 1
0111 0	0010 1
1111 0	

DIGITAL FILTERS

Digital filters are typically used to modify the attributes of a signal in the time or frequency domain through a process called linear convolution (Meyer-Baese, 2004). This process is formally described by the following formula:

$$y[n] = x[n] * f[n] = \sum_k x[k] \cdot f[n-k] = \sum_k x[k] \cdot c[k] \quad (1)$$

where the values $c[i] \neq 0$ are called the filter's coefficients.

There are only a few applications (e.g., adaptive filters) where general programmable filter architecture is required. In many cases, the coefficients do not change over time—linear time—invariant filters (LTI). Digital filters are generally classified as being finite impulse response (FIR) or infinite impulse response (IIR). As the names imply, an FIR filter consists of a finite number of sample values, reducing the previously presented convolution to a finite sum per output sample. An IIR filter requires that an infinite sum has to be performed. In this chapter, implementation of the LTI FIR filters will be discussed.

The output of an FIR filter of order (length) L, to an input time-samples $x[n]$, is given by a finite version of convolution sum:

$$y[n] = \sum_{k=0}^{L-1} x[k] \cdot c[k] \quad (2)$$

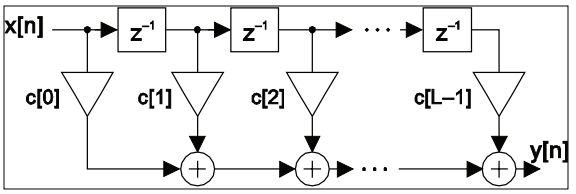
The L-th order LTI FIR filter is schematically presented in Figure 6. It consists of a collection of delay line, adders, and multipliers.

Much available digital filter software enables very easy computation of coefficients for a given filter. However, the challenge is mapping the FIR structure into suitable architecture. Digital filters are typically implemented as multiply-accumulate (MAC) algorithms with the use of special DSP devices.

Efficient hardware implementation of a filter's structure in programmable devices is possible by optimizing the implementation of multipliers and adders. In modern programmable structures, specialized embedded blocks can be used to implement multipliers, increasing the performance of the designed system. Moreover, in the case of Altera's devices, a whole MAC unit can be implemented in embedded DSP block, making the design methodology very similar to the one used in the case of DSP processors.

In the case of programmable devices, however, direct or transposed forms are preferred for maximum speed and lowest resource utilization.

Figure 6. Direct form FIR filter



This is because the approach enables exploitation of prevalent parallelism in the algorithm.

A completely different FIR architecture is based on the distributed arithmetic concept. In contrast to a conventional sum-of-products architecture, in the distributed arithmetic method, the sum of products of a specific bit of the input sample over all coefficients is computed in one step.

DISTRIBUTED ARITHMETIC METHOD

The distributed arithmetic method is a method of computing the sum of products. In many DSP applications, a general-purpose multiplication is not required. In the case of filter implementation, if filter coefficients are constant in time, then the partial product term $x[n] \cdot c[n]$ becomes a multiplication with a constant. Then, taking into account the fact that the input variable is a binary number:

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \cdot 2^b, \quad \text{where } [x_b \ \ n] \in [0,1] \tag{3}$$

The whole convolution sum can be described as shown next:

$$y[n] = \sum_{b=0}^{B-1} 2^b \cdot \sum_{k=0}^{L-1} x_b[k] \cdot c[k] = \sum_{b=0}^{B-1} 2^b \cdot \sum_{k=0}^{L-1} f(x_b[k], c[k]) \tag{4}$$

The efficiency of filter implementation based on this concept strongly depends on the implementation of the function $f(x_b[k],c[k])$. The preferred implementation method is to realize the mapping $f(x_b[k],c[k])$ as the combinational module with L inputs. The schematic representation of signed DA filter structure is shown in Figure 7, where the mapping f is presented as a lookup table that includes all the possible linear combinations of the filter coefficients and the bits of the incoming data samples (Meyer-Baese, 2004). The utility programs that generate the look-up tables for filters with given coefficients can be found in the literature.

In the experiments presented in this chapter, a variation of DA architecture has been used. It increases the speed of a filter at the expense of additional LUTs, registers, and adders. The basic DA architecture for computing the length L sum of products accepts one bit from every L input word. The computation speed can be significantly increased by accepting for the computation more bits per word. Maximum speed can be achieved with a fully pipelined parallel architecture, as shown in Figure 8. Such an implementation can outperform all commercially available programmable signal processors.

The HDL specification of the look-up table can be easily obtained for the filter described by its $c[i]$ coefficients. Since the size of look-up tables grows exponentially with the number of inputs, efficient implementation of these blocks becomes crucial to the final resource utilization of filter implementation. Here, advanced synthesis methods based on balanced decomposition can be successfully applied for technology mapping of DA circuits onto FPGA logic cells.

RESULTS

Experimental results for FIR filter implementation with different design methodologies are presented in this section. Filter with length (order) 15 has been chosen for the experiment. It has eight-bit signed input samples, and its coefficients can

Figure 7. DA architecture with look-up table (LUT)

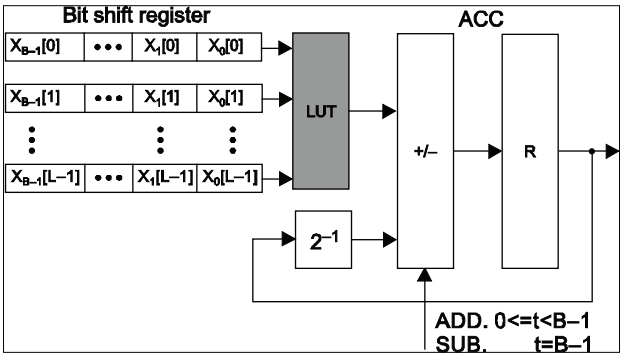
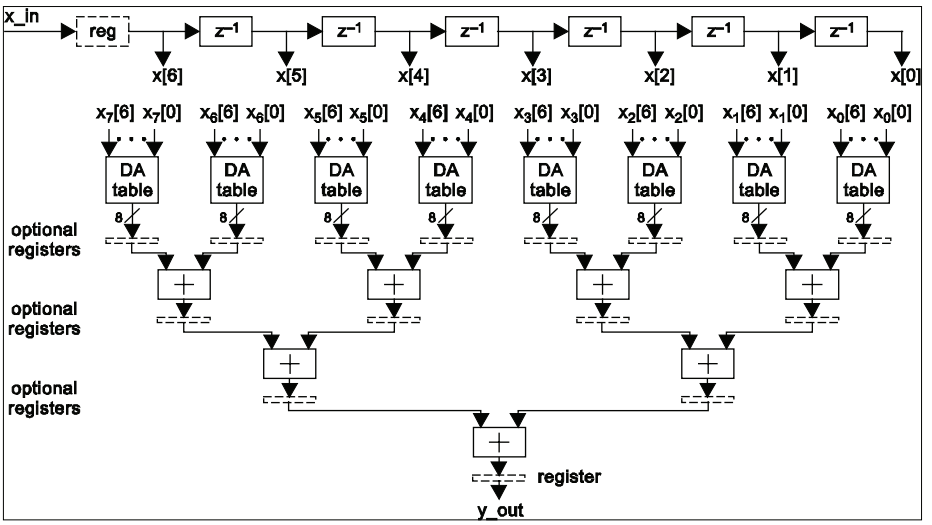


Figure 8. Parallel implementation of a distributed arithmetic scheme



be found in Goodman and Carey (1977). For comparison, the filter has been implemented in Stratix EP1S10F484C5, Cyclone EP1C3T100C6, and CycloneII EP2C5T144C6 structures using Altera QuartusII v5.1 SP0.15.

Table 7 presents the comparison of implementation results for different design methodologies. The column falling under the "MAC" label presents the results obtained by implementing the multiply-and-accumulate strategy with the use of logic cell resources; without utilization the embedded DSP blocks. Multipliers as well as accumulators were implemented in a circuit of logic cells. This implementation, due to its serial character, requires 15 clock cycles to compute the result. It requires a relatively large amount of resources, while delivering the worst performance in comparison to other implementations.

The next column, "MULT block," holds the implementation results of a method similar to "MAC" with a difference where multipliers were implemented in dedicated DSP-embedded blocks. It can be noticed that the performance of the filter increased at the cost of additional resources in the form of DSP-embedded blocks. Results in the column falling under "DSP block" were obtained by implementing the whole MAC unit in the embedded DSP block. Further increase in

Table 7. Implementation results for different design methodologies. Chip: S – Stratix EP1S10F484C5; C – Cyclone EP1C3T100C6, CII – CycloneII EP2C5T144C6 1) DSP blocks are not present in this device family.

Chip		MAC	MULT Block	DSP Block	Parallel	DA	DA Decomposed
S	LC	448	294	225	402	997	567
	DSP	0	2	4	30	0	0
	fmax [MHz]	74.6	85.06	107.30	53.3	65.14	78.75
C	LC	429	436	429	961	997	567
	DSP [1)]	–	–	–	–	–	–
	fmax [MHz]	77.85	79.31	77.85	57.71	72.56	70.87
CII	LC	427	294	275	670	952	567
	DSP	0	2	2	26	0	0
	fmax [MHz]	82.62	97.5	105.59	67.02	78.21	81.46

performance could be noticed, but still 15 clock cycles have to be used to compute the result.

Results given in the "Parallel" column were obtained by implementing the filter in a parallel manner. In this case, the results were obtained in a single clock cycle. Even though, the maximum frequency of this implementation is less than previous ones, it outperforms these implementations due to its parallel character.

Application of the DA technique results in the increased performance since the maximum frequency has increased. However, in this approach, more logic cell resource has been used since multipliers have been replaced by large combinational blocks and no DSP-embedded modules have been utilized.

Finally, results presented in the column "DA decomposed" demonstrate that the application of the DA technique combined with an advanced synthesis method based on balanced decomposition results in a circuit that not only outperforms any other implemented circuit but also reduces the necessary logic resource. The balanced decomposition technique was applied to decompose the combinational blocks of the DA implementation.

In Table 8, the experimental results of Daubechies' dbN, coifN, symN, and 9/7-tap biorthogonal filter banks are presented. Filters 9/7 are in two versions: (a) analysis filter and (s) synthesis filter. Filters dbN, coifN, symN are similar for analysis and synthesis (a/s). All filters have 16-bit signed samples and have been implemented with the use of distributed arithmetic concept in the fully parallel way. Balanced decomposition software was also added to increase efficiency of the DA tables' implementations.

Table 8 presents the result for filter implementations using Stratix EP1S10F484C5 device, with a total count of 10,570 logic cells. In the implementation without decomposing the filters, the new method was modeled in AHDL, and Quartus2v6.0SP1 was used to map the model into the target structure. In the implementation using decomposition, the automatic software was used to initially decompose DA tables, and then the Quartus system was applied to map the filters into FPGA.

The application of the balanced decomposition concept significantly decreased the logic cell resource utilization and at the same time increased the speed of the implementation.

Table 8. Implementation results of filters with and without decomposition

Filter	Order	Without Decomposition		With Decomposition	
		LC	f_{max} [MHz]	LC	f_{max} [MHz]
db3, a/s low-pass	6	1596	278,63	1345	254,26
db4, a/s low-pass	8	3747	212,9	2891	201,73
db5, a/s low-pass	10	10057	169,81	7377	119,39
db6, a/s low-pass	12	–**	–	31153	–*
9/7, a low-pass	9	3406	206,61	1505	212,86
9/7, s low-pass	7	1483	273,37	881	263,5
9/7, a high-pass	7	2027	253,29	1229	223,16
9/7, s high-pass	9	4071	180,93	1616	189,47
coif6, a/s low-pass	6	1133	283,45	1041	260,62
coif12, a/s low-pass	12	–**	–	1614	196,85
sym8, a/s low-pass	8	3663	212,72	2249	197,94
sym12, a/s low-pass	12	–**	–	2313	198,61
sym14, a/s low-pass	14	–**	–	2345	200,24
sym16, a/s low-pass	16	–**	–	2377	206,83

** does not fit in EP1S10F484C5*

*** too long compilation time (more than 24 hours)*

CONCLUSION

The modern programmable structures deliver the possibilities to implement DSP algorithms in dedicated embedded blocks. This makes designing of such an algorithm an easy task. However, the flexibility of programmable structures enables more advanced implementation methods to be used. In particular, exploitation of parallelism in the algorithm to be implemented may yield very good results. Additionally, the application of advanced logic synthesis methods based on balanced decomposition, which is suitable for FPGA structure, leads to results that cannot be achieved with any other method.

The presented results lead to the conclusion that if the designer decides to use the methodology known from DSP processor application, the implementation quality will profit from the utilization of specialized DSP modules embedded in the programmable chip. However, best results can be obtained by utilizing the parallelism in implemented algorithms and by applying advanced synthesis methods based on decomposition. Influence of the design methodology and the balanced decomposition synthesis method on the efficiency of practical digital filter implementation is particularly significant when the designed circuit contains complex combinational blocks. This is a typical situation when implementing digital filters using the DA concept.

The most efficient approach to logic synthesis of FIR filter algorithms discussed in this chapter relies on the effectiveness of the functional decomposition synthesis method. These methods were already used in decomposition algorithms; however, they were never applied together in a technology-specific mapper targeted at a look-up table FPGA structure. This chapter shows that it is possible to apply the balanced decomposition method for the synthesis of FPGA-based circuits directed toward area or delay optimization.

ACKNOWLEDGMENT

This work is supported by Ministry of Science and Higher Education financial grant for years 2006-2009 (Grant No. SINGAPUR/31/2006) as well as Agency for Science, Technology and Research in Singapore (Grant No. 0621200011).

REFERENCES

Brislawn, C.M., Bradley, C.B.J., Onyshczak, R., & Hopper, T. (1996). The FBI compression standard for digitized fingerprint images. *Proceedings of the SPIE Conference 2847,* Denver, Colorado, 344–355.

Brzozowski, J.A., & Łuba, T. (2003). Decomposition of Boolean functions specified by cubes. *Journal of Multiple-Valued Logic and Soft Computing, 9,* 377–417.

Chang, S.C., Marek-Sadowska, M., & Hwang, T.T. (1996). Technology mapping for TLU FPGAs based on decomposition of binary decision diagrams. *IEEE Trans on CAD, 15*(10), 1226–1236.

Croisier, A., Esteban, D., Levilion, M., & Rizo, V. (1973). *Digital filter for PCM encoded signals*. US Patent No. 3777130.

Daubechies, I. (1992). *Ten lectures on wavelets.* SIAM.

Falkowski, B.J. (2004). Compact representation of logic functions for lossless compression of grey scale images. *IEEE Proceedings, Computers and Digital Techniques, 151*(3), 221–230).

Falkowski, B.J., & Chang, C.H. (1997). Forward and inverse transformations between Haar spectra and ordered binary decision diagrams of Boolean functions. *IEEE Trans on Computers, 46*(11), 1271–1279.

Goodman, D.J., & Carey, M.J. (1977). Nine digital filters for decimation and interpolation. *IEEE Trans on Acoustics, Speech and Signal Processing, 25*(2), 121–126.

Lapsley, P., Bier, J., Shoham, A., & Lee, E. (1997). DSP processor fundamentals. New York: IEEE Press.

Lee, E. (1988). Programmable DSP architectures: Part I. *IEEE Transactions on Acoustics, Speech and Signal Processing Magazine,* 4–19.

Lee, E. (1989). Programmable DSP architectures: Part II. *IEEE Transactions on Acoustics, Speech and Signal Processing Magazine,* 4–14.

Łuba, T. (1995). Decomposition of multiple-valued functions. *Proceedings of the 25th International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, (pp. 256–261).

Łuba, T., & Selvaraj, H. (1995). A general approach to Boolean function decomposition and its applications in FPGA-based synthesis. *VLSI Design, Special Issue on Decompositions in VLSI Design, 3*(3-4), 289–300.

Łuba, T., Selvaraj, H., Nowicka, M., & Kraśniewski, A. (1995). Balanced multilevel decomposition and its applications in FPGA-based synthesis. In G. Saucier, & A. Mignotte (Eds.), *Logic and architecture synthesis.* Chapman & Hall.

Meyer-Baese, U. (2004). *Digital signal processing with field programmable gate arrays. Second edition.* Berlin: Springer Verlag.

Nowicka, M., Łuba, T., & Rawski, M. (1999). FPGA-based decomposition of Boolean functions: Algorithms and implementation. *Proceedings of the Sixth International Conference on Advanced Computer Systems,* Szczecin, Poland, 502–509.

Peled, A., & Liu, B. (1974). A new realization of digital filters. *IEEE Transactions on Acoustics, Speech and Signal Processing, 22*(6), 456–462.

Rao, R.M., & Bopardikar, A.S. (1998). *Wavelet transform: Introduction to theory and applications.* Addison-Wesley.

Rawski, M., Jóźwiak, L., & Łuba T. (1999a). Efficient input support selection for sub-functions in functional decomposition based on information relationship measures. *Proceedings of the EUROMICRO'99 Conference*, Milan, Italy.

Rawski, M., Jóźwiak, L., & Łuba, T. (1999b). The influence of the number of values in sub-functions on the effectiveness and efficiency of the functional decomposition. *Proceedings of the EUROMICRO'99 Conference*, Milan, Italy.

Rawski, M., Jóźwiak, L., & Łuba T. (2001). Functional decomposition with an efficient input support selection for sub-functions based on information relationship measures. *Journal of Systems Architecture, 47,* 137–155.

Rawski, M., Selvaraj, H., & Morawiecki, P. (2004). Efficient method of input variable partitioning in functional decomposition based on evolutionary algorithms. *Proceedings of the DSD 2004 Euromicro Symposium on Digital System Design, Architectures, Methods and Tools*, Rennes, France, (pp. 136–143).

Rawski, M., Tomaszewicz, P., & Łuba, T. (2004). Logic synthesis importance in FPGA-based designing of information and signal processing systems. *Proceedings of the International Conference on Signal and Electronics Systems*, Poznań, Poland, (pp. 425–428).

Rawski, M., Tomaszewicz, P., Selvaraj, H., & Łuba, T. (2005). Efficient implementation of digital filters with use of advanced synthesis methods targeted FPGA architectures. *Proceedings of the Eighth Euromicro Conference on DIGITAL SYSTEM DESIGN*, Portugal, (pp. 460–466).

Rioul, O., & Vetterli, M. (1991). Wavelets and signal processing. *IEEE Signal Processing Magazine*, 14–38.

Sasao, T., Iguchi, Y., & Suzuki, T. (2005). On LUT cascade realizations of FIR filters. *Proceedings of the Eighth Euromicro Conference on DIGITAL SYSTEM DESIGN*, Portugal, (pp. 467–474).

Scholl, C. (2001). *Functional decomposition with application to FPGA synthesis.* Kluwer Academic Publishers.

Chapter XIII
A Novel Support Vector Machine with Class-Dependent Features for Biomedical Data

Nina Zhou
Nanyang Technological University, Singapore

Lipo Wang
Nanyang Technological University, Singapore

ABSTRACT

This chapter introduces an approach to class-dependent feature selection and a novel support vector machine (SVM). The relative background and theory are presented for describing the proposed method, and real applications of the method on several biomedical datasets are demonstrated in the end. The authors hope this chapter can provide readers a different view of feature selection method and also the classifier so as to promote more promising methods and applications.

INTRODUCTION

Since the datasets we process are becoming increasingly larger in the number of patterns and the dimension of features or attributes, data preprocessing has already become a very necessary step for us to reduce the computational complexity, save computational cost, and also improve the efficiency of many learning algorithms used on the data. Data preprocessing techniques can be classified into two categories: sample selection (Blum & Langley, 1997) to reduce the number of patterns and data dimensionality reduction (Guyon & Elisseeff, 2003; Liu & Motoda, 1998; Wang & Fu, 2005) to reduce the dimensionality of the features (e.g., feature selection and feature extraction). In the majority of cases, since the data have no irrelevant samples, feature selection or feature extraction is our major challenge due to the prevalence of high-dimensional data with some irrelevant or redundant features.

The support vector machine (SVM) as a learning algorithm has been successfully used in the field of computational biology, such as microarray gene expression analysis (Weston, Mukherjee, Chapelle, Pontil, Poggio & Vapnik, 2000), cancer classification (Wang, Chu & Xie, 2006), and protein secondary structure prediction (Hua & Sun, 2001). In this chapter, we review the background of feature selection and state an application of the SVM on feature selection and classification for biomedical datasets with class-dependent feature subsets.

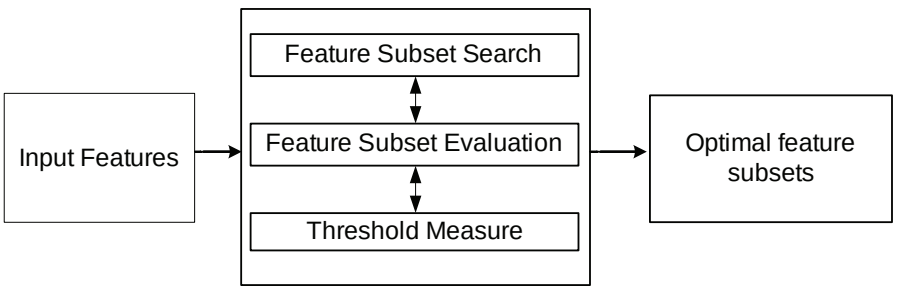
BACKGROUND

Feature extraction (Liu & Motoda, 1998; Wang & Fu, 2005) is the process of transforming the original features into a new form of feature space. Feature extraction does not delete any features but extracts a set of new features from the original set by a certain mapping. The most typical example of feature extraction is principal component analysis (PCA) (Liu & Motoda, 1998). PCA first calculates the covariance matrix from the input data and the eigenvalues and eigenvectors of the matrix, and ranks them in descending order with respect to the eigenvalues. Second, PCA takes a predefined number of components (eigenvectors) to form a transformation matrix. Although feature extraction (e.g., PCA) is an efficient data dimensionality reduction technique, the newly generated features are usually difficult to interpret. Therefore, we prefer feature selection.

Feature selection (Liu & Motoda, 1998; Wang & Fu, 2005) is the process of eliminating irrelevant or redundant features, leaving the best subset of features, which retains sufficient information so as to discriminate well among classes. It usually involves ranking the features first (Fu & Wang, 2003; Guyon & Elisseeff, 2003) and then deleting those irrelevant or redundant features. Hence, how to rank features and how to remove those irrelevant or redundant features are the main objectives of feature selection.

Based on the various measures used to find the best feature subset, feature selection usually can be classified into the two broad categories (Liu & Motoda, 1998): filter approaches (Almauallium & Dietterich, 1991; Kira & Rendell, 1992) and wrapper approaches (Devijver & Kittler, 1982; John, Kohavi & Pfleger, 1994). In addition, some authors refer to one more category—embedded approaches (see references cited in Blum & Langley, 1997)—which will not be mentioned in this chapter. Filter approaches (Figure 1) select features independent of any classifiers. Some filter approaches first evaluate the features' relevance in terms of the intrinsic properties of the data and then select features according to a predefined threshold. For example, the RELIEF algorithm (Kira & Rendell, 1992) and its extended version RELIEFF (Kononenko, 1994) assign a weight to each feature and then update the weight according

Figure 1. Filter approaches

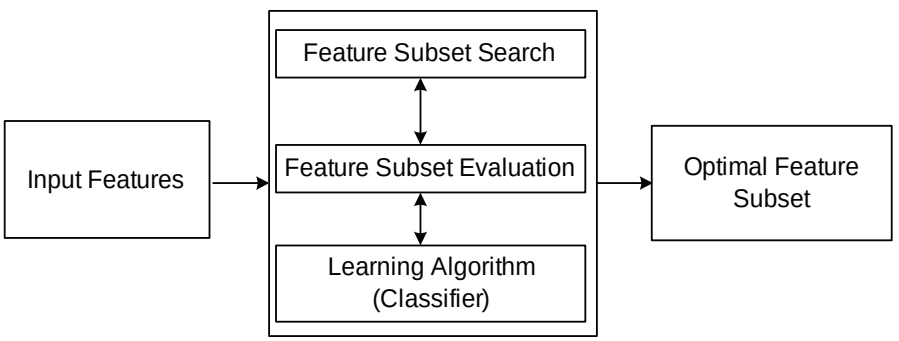


to training instances. These weights correspond to relevance of features. All features are ranked according to their weights and features; weights above a predefined threshold are selected, whereas other features with weights below the threshold are deleted. Not using the ranking strategy, some filter approaches attempt the greedy search for all feature subsets to find a proper feature subset (e.g., FOCUS) (Almauallium & Dietterich, 1991) and cross-entropy filter (Koller & Sahami, 1996). FOCUS stops search when a minimal feature subset is consistent with training data. The cross-entropy filter uses a predefined threshold to determine the proper feature subset.

Wrapper approaches (Figure 2) "wrap" feature selection around a learning algorithm. Wrapper approaches utilize some heuristic search techniques (e.g., sequential forward and backward search) (Langeley & Sage, 1994a; Wang & FU, 2005;), hill climbing (Caurana & Freitag, 1994), best-first search (Kohavi & John, 1998, cited in Blum & Langley, 1997), and beam search (Aha & Bankert, 1995, cited in Blum & Langley, 1997), to search for possible feature subsets and use a learning algorithm (classifier) to evaluate the feature subsets and determine the optimal one in terms of classification accuracy. For example, Langeley and Sage (1994a) adopted the search strategy of backward elimination and used the simple nearest neighbor classifier to find the proper feature subset. The algorithm begins with all features, each time removing one feature from the whole set and evaluating the performance of the current feature subset by the nearest neighbor classifier. The feature whose removal leads to the best learning performance is finally gotten rid of. This process is repeated until no improvement on accuracy can be achieved. Caurana and Freitag's (1994) wrapper approach uses hill climbing as the search strategy and the decision tree as the leaning algorithm. Compared to filter approaches, wrapper approaches are computationally more expensive due to the introduction of a learning machine (classifier). But for the same reason, wrapper approaches usually lead to better classification accuracy than filter approaches. In order to strike a balance between speed and accuracy for wrapper approaches, many researchers have tried to reduce evaluation time of learning algorithms. For example, Caurana and Freitag (1994) cached decision trees. Moore and Lee (1994) tried to reduce the number of training instances so as to speed up the evaluation process. In real applications, if the number of features or instances is not very large, the usual choice is a wrapper approach so as to achieve better classification accuracy. In this chapter, we choose a wrapper approach to process biomedical data.

Rather than the previous two categories of filter and wrapper approaches, feature selection methods may also be divided into another two categories: *class-independent* feature selection (Fu

Figure 2. Wrapper approaches



& Wang, 2003; Gilad-Bachrach, Navot & Tishby, 2004; Kira & Rendell, 1992), and *class-dependent* feature selection (Fu & Wang, 2002; Oh, Lee & Suen, 1998, 1999), which are based on whether the selected feature subset is related to classes. Class-independent feature selection chooses a feature subset without regard to the classes in a given classification problem. Class-dependent feature selection chooses feature subsets with regard to the classes (i.e., different feature subsets for different classes). Most of the feature selection methods, filters, or wrappers mentioned in the previous paragraph (Almauallium & Dietterich, 1991; Kira & Rendell, 1992; Kononenko, 1994)) belong to the category of class-independent feature selection. However, they have not considered the possibility that different groups of features may have different power in distinguishing different classes (Fu & Wang, 2002; Oh et al., 1998, 1999). Furthermore, class-dependent feature selection can theoretically improve upon or at least match the performance of class-independent feature selection because the latter can be considered a special case of the former. In order to take advantage of the possibility that different features have different classification power and demonstrate it in practice, we adopt class-dependent feature selection and use a *wrapper approach to selecting class-dependent features* for biomedical data (Newman, Hettich, Blake & Merz, 1998).

Many classifiers have been employed in feature selection (e.g., decision trees) (Doak, 1992; John et al., 1994), the naive Bayes classifier (Langeley & Sage, 1994b), neural networks, and the support vector machine (SVM) (Guyon, Gunn, Ben-Hur & Dror, 2004; Vapnik, 1998; Wang, 2005), to determine the optimal feature subset. In the NIPS 2003 feature selection challenge (Guyon et al., 2004), most winners chose the SVM as their classifiers. In addition, considering many attractive features (Hua & Sun, 2001) such as effectively avoiding overfitting and accommodating large feature spaces, we will use the SVM in our wrapper approach.

This chapter is organized as follows. In Section 2, we first review the two adopted feature importance ranking measures (i.e., the RELIEFF and class separability measure (CSM)) and then introduce our wrapper approach to class-dependent feature selection. We will also review the basic theory of the SVM. In Section 3, we present experiment results about our method on three datasets from the UCI machine learning repository databases (Newman et al., 1998) and make a comparison on classification accuracies between class-dependent and class-independent feature selection methods. In the end, we present conclusions about the present work and discussions for future work.

METHODOLOGY

In our wrapper approach to feature selection, the basic process is first to rank each feature by a ranking measure, and then, according to the ranking list, form different feature subsets. The formation process of feature subsets starts with the top one feature, followed by additions of the next top feature into the previous subset until we find the best feature subset that leads to the highest classification accuracy.

Feature Importance Ranking Measures

There are many measures that can be used to evaluate feature importance, such as the class separation measure (Devijver & Kittler, 1982; Oh et al., 1999), the information theoretic ranking criteria (Guyon & Elisseeff, 2003), the RELIEF (Kira & Rendell, 1992), and the separability-correlation measure (SCM) (a combination of a class separability measure and an attribute-class correlation measure) (Fu & Wang, 2003; Wang & Fu, 2005). In this chapter, we adopt the RELIEF and the class separability measure (CSM) proposed by Fu and Wang (2003). In the following,

we will give a brief description of the RELIEF and CSM.

RELIEF Ranking Measure

RELIEF, proposed by Kira and Rendell (1992), is a feature weight-based evaluation algorithm. For each feature, RELIEF calculates a relevance score according to the difference between the distance of each sample and the current sample's nearest hit and the distance of each sample and the current sample's nearest miss. *A nearest hit* is the sample closest to and with the same class as the current sample, and *a nearest miss* is the sample closest to but in different classes from the current sample. The relevance score is assigned to the feature as the feature's weight. RELIEF is a very efficient evaluation algorithm, but it was designed for two-class problems. Kononenko (1994) extended RELIEF to RELIEFF, which can handle multiclass problems. First, for each sample, RELIEFF finds the nearest hit of the sample in the same class and the nearest misses in all the other classes. Then the algorithm averages over the different misses by the weights of the *a priori* probability of each class (Bins, 2000).

The RELIEF evaluation algorithm can be described in the following five steps:

1. Given a weight vector with each element corresponding to one feature, set it to zero (i.e., $\{w_1,...,w_i,...,w_N\} = 0$). N is the number of features.
2. From the training instance set S, randomly select an input vector (e.g., the j-th sample $\vec{x}_j = \{x_{j1},...,x_{ji},...,x_{jN}\}$) and assume that it belongs to class c_1. Then find the *nearest hit* $\vec{x}_j^h$ that satisfies $Dist(\vec{x}_j{}^h, \vec{x}_j) = \arg\min Dist(\vec{x}_i{}^h, \vec{x}_j)$ (Bins, 2000), in which $\vec{x}_j^h$ and $\vec{x}_i^h \in S, \vec{x}_j^h$ and $\vec{x}_i^h \in c_1$, and the nearest miss $\vec{x}_j^m$ which satisfies $Dist(\vec{x}_j{}^m, \vec{x}_j) = \arg\min Dist(\vec{x}_i{}^m, \vec{x}_j)$ (Bins, 2000), in which $\vec{x}_i{}^m$ and $x_j{}^m \notin c_1$).
3. Calculate and update the weight w_i of the i-th feature in the case of sample $\vec{x}_j$:

   $$w_i = w_i - (x_{ji} - x_{ji}^h)^2 + (x_{ji} - x_{ji}^m)^2,$$
   $$i = 1, 2, ..., N$$

   x_{ji}^m and x_{ji}^h represent the i-th elements of $\vec{x}_j^m$ and $\vec{x}_j^h$, respectively. The updating (Bins, 2000) means the weight will increase if the hit difference (the difference between the feature and its hit (i.e., $(x_{ji} - x_{ji}^h)^2$) is less than the miss difference (the difference between the feature and its miss (i.e., $(x_{ji} - x_{ji}^m)^2$).
4. Repeat step 2 and 3 over the training instance set S.
5. Rank the features according to their weights. The basic rationale is the larger the weights, the more important (relevant) the features.

Class Separability Measure

Class separability measure has been used by many people with different versions. For example, Devijver and Kittler (1982) defined the class separability for a dataset in the form of $trace(M_b{}^{-1}M_w)$. Here, M_b is the between-class scatter matrix, and M_w is the within-class scatter matrix. The detailed formula descriptions about the two matrices are available in Devijver and Kittler (1982). Oh, et al. (1999) also defined a class separation to measure how well two classes are separated by a feature vector $\vec{x}$. The class separation proposed by Oh, et al. (1999) is represented by $S^{cc}(c_i, c_j, \vec{x})$, where c_i and c_j represent the i-th class and the j-th class of the dataset, respectively. $S^{cc}(c_i, c_j, \vec{x})$ calculates the difference of two classes c_i and c_j using estimated class distributions for the feature vector $\vec{x}$. Each feature's class separation is calculated individually (e.g., $S^{cc}(c_i,c_j,x_p)$) for the p-th feature, and features are ranked according to their class separation values. Fu and Wang (2003) defined another class separation measure to rank each feature's classification capability. This CSM (Fu & Wang, 2003) includes two distance elements: the

within-class distance (distance between patterns within each class) and the between-class distance (the distance between patterns among different classes). Usually the smaller the within-class distance and the greater the between-class distance, the smaller the ratio of the within-class distance to the between-class distance, then the more accurate the classification. Therefore, according to Fu and Wang (2003), the ratio of the within-class distance to the between-class distance is used to measure the feature's classification ability; that is, each time, we have one feature removed from the training data and calculate the ratio. If one feature is very important (relevant), its removal will lead to the increase of the ratio compared to the one without removing any features, and vice versa. Therefore, we say that the greater the ratio after removing the feature, the more important the removed feature for the classification. Let us see more details about the algorithm.

For the whole training data, the within-class distance S_w (Fu & Wang, 2003) is calculated as:

$$S_w = \sum_{c=1}^{C} P_c \sum_{j=1}^{n_c} (\vec{x}_{cj} - \vec{\bar{m}}_c)(\vec{x}_{cj} - \vec{\bar{m}}_c)^T \quad (1)$$

and the between-class distance S_b (Fu & Wang, 2003) is calculated as:

$$S_b = \sum_{c=1}^{C} P_c (\vec{\bar{m}}_c - \vec{\bar{m}})(\vec{\bar{m}}_c - \vec{\bar{m}})^T \quad (2)$$

Here, C refers to the number of classes, and P_c refers to the probability of the c-th class. n_c refers to the number of samples in the c-th class, and $\vec{x}_{cj}$ refers to the j-th sample in the c-th class. $\vec{\bar{m}}_c$ refers to the mean vector of the c-th class, and $\vec{\bar{m}}$ refers to the mean vector of all the training samples. As mentioned previously, the smaller the ratio S_w / S_b, the better the separability. When evaluating each feature's classification ability, we use S_w'/S_b' to represent the ratio after removing a feature. The greater $S_w'/S_b'^{(r)}$, the more important the removed feature (the r-th) is. For example, assume $S_w'/S_b'^{(1)}$ represents the ratio after removing the first feature and $S_w'/S_b'^{(2)}$ represents the ratio after removing the second feature. If $S_w'/S_b'^{(1)}$ is greater than $S_w'/S_b'^{(2)}$, then the first feature is more important than the second feature. Hence, we may evaluate the importance level of the features according to the ratio S_w'/S_b' (Fu & Wang, 2003) with an attribute deleted each time in turn.

Although the two measures cannot detect redundancy in features, the RELIEF and CSM perform well on detecting relevance of features even in the presence of features interaction, which is the reason why the two measures are widely used in feature selection.

The Classifier: The Support Vector Machine

As a classifier, the SVM has already been used successfully in many fields and proves to have significantly better performance than or at least the same as traditional machine learning approaches, including neural networks (Hua & Sun, 2001). The basic idea of the SVM refers to a space transformation with a nonlinear mapping (i.e., transforming the linearly inseparable data in the original feature space into a new linearly separable space). The dimensionality of the new feature space is much higher than the original one so that the optimal separating hyperplane (i.e., the best decision function) can be found in the new feature space (Vert, 2001).

To make this more concrete, let us begin with a linearly separable case with two classes. Assume that the whole training dataset is $S = \{(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)\}$, in which $y_i \in \{-1, +1\}$ ($i = 1,2,...m$) is the class label of the input sample $\vec{x}_i$. The SVM tries to find an optimal hyperplane:

$$\vec{w}^T \vec{x}_i + b = 0 \quad (3)$$

to maximize the margin of the separation between the two classes of data. In Equation (3), $\vec{w}$ is a weight vector connecting the input to the

hyperplane, and *b* is called a bias. The optimal hyperplane is obtained by solving the following constrained optimization problem:

$$\text{minimizing } \frac{1}{2}\|\vec{w}\|^2 \tag{4}$$

$$\text{subject to } y_i(\vec{w}^T\vec{x}_i + b) \geq 1 \, (i = 1,2,...m) \tag{5}$$

If the data are linearly inseparable, the constrained optimization problem will be different. For the linearly inseparable case, a set of non-negative slack variables $\xi_1, \xi_2, ..., \xi_m$ are introduced to penalize the training error, and a regularization parameter ς is introduced to control the trade-off between the training error and the margin. The corresponding constrained optimization problem is changed into:

$$\text{minimizing } \frac{1}{2}\|\vec{w}\|^2 + \varsigma\sum_{i=1}^{m}\xi_i \tag{6}$$

$$\text{subject to } y_i(\vec{w}^T\vec{x}_i + b) \geq 1 - \xi_i \, (\xi_i \geq 0, i = 1,2,...m) \tag{7}$$

When using the Lagrangian function to solve the constrained minimization problem, the problems become:

$$\text{maximizing } \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\,\alpha_j y_i y_j \vec{x}_i^T\vec{x}_j \tag{8}$$

$$\text{subject to } \sum_{i=1}^{m}\alpha_i y_i = 0 \;\; (0 \leq \alpha_i \leq \varsigma) \tag{9}$$

α_i and α_j are the Lagrangian multipliers within the range of [0, ς] (ς is the regularization parameter). For this linearly inseparable problem, a space transformation is needed. Suppose ϕ is a vector of transformation functions, which can realize the transformation from a lower dimensional input space into a higher dimensional space. The transformation of the input vector $\vec{x}$ (with the dimension *N*) from the input space to the transformed space is defined as:

$$\phi(\vec{x}) = [\phi_1(\vec{x}), \phi_2(\vec{x}), ..., \phi_N(\vec{x})]^T \tag{10}$$

The decision function in the transformed space:

$$\sum_{j=1}^{N} w_j\phi_j(\vec{x}) + b = 0 \tag{11}$$

can be further simplified as:

$$\sum_{j=0}^{N} w_j\phi_j(\vec{x}) = 0 \Rightarrow w^T\phi(\vec{x}) = 0 \tag{12}$$

where w_0 is a given bias and $\phi_0(\vec{x}) = 1$. w_j (j = 1,2,...,N) is a given weight connecting the feature space to the output space. The connecting weight can be obtained through a quadratic programming (QP) solver (Vapnik, 1998; Wang, 2005) (note that $\vec{w} = \{w_1, w_2, ..., w_N\}$ does not include the bias):

$$\vec{w} = \sum_{i=1}^{m}\alpha_i y_i \phi(\vec{x}_i) \tag{13}$$

Applying Equation (13) to Equation (12), we obtain:

$$\sum_{i=1}^{m}\alpha_i y_i \phi^T(\vec{x}_i)\phi(\vec{x}) = 0 \tag{14}$$

Instead of calculating the transformation function ϕ, an inner product kernel function *K* is introduced to realize the mapping:

$$K(\vec{x}, \vec{x}_i) = K(\vec{x}_i, \vec{x}) = \phi^T(\vec{x}_i)\phi(\vec{x}) \tag{15}$$

The optimal hyperplane is found by solving:

$$\text{maximizing } \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\,\alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j) \tag{16}$$

$$\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0 \ \ (0 \le \alpha_i \le \varsigma) \qquad (17)$$

The inner product kernel function represents the transformation function ϕ and makes the inseparable data separable in new feature space. In summary, the introduction of kernel function can help avoid the curse of dimensionality during optimization in high-dimensional spaces. Hence, choosing an appropriate kernel function is very important to the SVM's performance. There are three popular types of kernels (Hsu, Chang & Lin, 2003; Vert, 2001) used in most SVM packages (i.e., the polynomial learning machine, the radial-basis function (RBF) network, and the two-layer perceptron. Because the RBF kernel has many advantages compared with other kernels (e.g., with less parameters), we will choose the RBF kernel in our application.

Our Class-Dependent Feature Selection Approach

To realize the class-dependent feature selection for multiclass problems, we will adopt the strategy of "one-against-all" (Bottou et al., 1994; Hsu & Lin, 2002) (i.e., the current class vs. all the other classes. If the number of classes and the number of features are *C* and *N*, respectively, the computation complexity of the class-dependent ranking measure will be $O(C \bullet N)$. We describe the proposed wrapper approach to class-dependent feature selection in three steps.

In step one, according to the idea of one-against-all, we convert a *C*-class classification problem to *C* 2-class classification problems (i.e., problem 1, …, problem *i*, …, and problem *C*). Each problem has only two classes: one is the original class, and the other one includes all the classes except the original class.

In step two, for each 2-class problem, we adopt the RELIEF ranking measure and CSM to evaluate the importance of the features. *C* 2-class problems will have *C* different feature importance ranking lists. For example, for problem 1, the importance ranking is to measure how the features classify class 1 from the other classes; thus, the feature importance ranking list is specific to class 1. Likewise for class 2, class 3, …, and class *C*. Therefore, we call the feature importance ranking thus obtained *class-dependent feature importance ranking.* In the experiment, we compare the two ranking measures' performance.

In step three, for each feature, based on the class-dependent feature importance ranking list, we form different feature subsets by sequentially adding one feature into the previous subset. The optimal feature subset (i.e., the one with the highest classification accuracy) is determined by the classifier SVM. For each class, we start with the feature with the top ranking value as the first subset and then each time add one feature into the previous subset from the ranking list to form a new feature subset until all the features in this class are added or the highest classification accuracy is reached. We evaluate each feature subset with a classifier, and the feature subset corresponding to the highest classification accuracy will be our choice for this class.

For convenience, we introduce a feature mask to represent the state of each feature. The feature mask has only two elements, '0' and '1', in which '0' represents the absence of a particular feature, and '1' represents the presence of the feature. For example, consider a dataset with five features $\{x_1, x_2, x_3, x_4, x_5\}$ if the optimal feature subset obtained is $\{x_1, x_3, x_5\}$ with the second and fourth features deleted, the feature mask for this feature subset should be {1, 0, 1, 0, 1}.

After finishing these three steps, we obtain the class-dependent feature subset, which cannot be classified by any single classifier except the RBF neural network (Fu & Wang, 2002, 2003). Here, we attempt to use the SVM instead, considering the advantages of the SVM.

The Constructed Novel SVM Classifier for Class-Dependent Features

The SVM was originally designed for the binary classification, although most problems are multiclass. In order to solve those multiclass problems, some methods have been proposed to effectively extend the binary SVM to the multiclass classification (e.g., the one-against-all method (Bottou et al., 1994), the one-against-one method (Kressel, 1999), and the DAGSVM method (Platt, Cristianini & Shawe-Taylor, 2000). With regard to those methods, Hsu and Lin (2002) made a comparison and pointed out that the one-against-one and DAGSVM methods are more suitable for practical use, and one-against-all is a good method whose performance is comparable to one-against-one. For classification, all these SVM classifiers mentioned in Bottou, et al. (1994), Kressel (1999), and Platt, et al. (2000) cannot directly accommodate class-dependent feature subsets. Considering the characteristic of class-dependent feature subsets and the process of the feature selection, we use one-against-all in the construction of the novel SVM classifier. The novel SVM classifier is constructed by building several SVM models and then combining them together for accommodating class-dependent feature subsets. Each model is a binary classifier and is specific for one class. In the following, we will introduce the construction process:

1. **The training process.** In this process, we use training patterns to construct *C* SVM models. Each class has its own model according to its specific feature subset. The model *i* is trained with all the training examples in the *i*-th class having positive labels and all the examples in other classes having negative labels. Specifically, all the training examples need to be filtered by a feature mask of the *i*-th class before they are input for training. For instance, if the feature mask of the *i*-th class has $n^{(i)}$ '1', all the training examples to form the *i*-th model will have the corresponding $n^{(i)}$ features as the input, and those features corresponding to '0' in the feature mask are removed. The output y_i has the value of '+1' and '-1'. If the input pattern $\vec{x}_i$ belongs to the *i*-th class, we consider it a positive sample ('+1'), or we consider it a negative sample, denoted as '-1'.
   The *i*−th SVM model solves the following problem:

$$\begin{aligned} &\min_{\vec{w}^i, b^i, \xi^i} \frac{1}{2}(\vec{w}^i)^T \vec{w}^i + \varsigma^i \sum_{j=1}^{m} \xi_j^i \\ &(\vec{w}^i)^T \phi(\vec{x}_j^i) + b^i \geq 1 - \xi_j^i, \text{ if } y_j = i; \\ &(\vec{w}^i)^T \phi(\vec{x}_j^i) + b^i \leq 1 - \xi_j^i, \text{ if } y_j \neq i; \\ &\xi_j^i \geq 0 \end{aligned} \tag{18}$$

2. **The testing process.** After the class-dependent models are constructed, we will use them to test unlabeled patterns. Same as the training process, each testing pattern is filtered with one class's feature mask before input into the corresponding SVM model (i.e., the original attributes corresponding to '0' in the feature mask are removed). Among the *C* outputs, the testing pattern (e.g., $\vec{x}$) belongs to the class with the largest output value:

$$\text{Class of } \vec{x} \equiv \arg\max_{i=1,2,\ldots,C}((\vec{w}^i)^T \phi(\vec{x}^i) + b^i) \tag{19}$$

EXPERIMENTS AND ANALYSIS

We experiment on three biomedical datasets from the UCI machine learning repository databases (Newman et al., 1998). Our major goal is to analyze whether the class-dependent feature selection can obtain more efficient feature subsets so as to improve the classification accuracy more than the class-independent feature selection. Therefore, we make the comparison between the two methods in terms of the number of deleted features and the classification accuracy.

Experimental Data

The first benchmark data is the breast cancer dataset from the University of Wisconsin Hospital. The breast cancer dataset has 699 samples, nine attributes, and two classes (benign and malignant), in which benign has 458 samples and malignant has 241 samples. The second dataset is the E. coli dataset. It has seven attributes (localization sites of the protein) and eight classes. The number of instances is 336.

The third dataset is called the processed Cleveland dataset. It mainly concerns heart disease diagnosis and is collected from the Cleveland Clinic Foundation. There are originally 303 samples, 13 features, and five classes. Because there are six samples with unknown feature values, we remove the six samples from our experiment.

Software Preparations

Since there are many kinds of SVM software packages available, we choose LIBSVM 2.8 (Chang & Lin, 2001) with the RBF kernel functions in our experiment. We use the 10 fold-cross validation method to train and test the three datasets; that is, we separate the whole dataset into 10 equal subsets, each time having one subset as the testing part and the rest of nine subsets as the training part.

The whole training process can be described in the following steps (Hsu et al., 2003):

1. Format the datasets to make them suit the functions and normalize the datasets.
2. Set the regularization parameter ς and the kernel parameter (the radial basis function [RBF] kernel) σ. Use the cross-validation and grid-search to find the best pair of the parameter (ς,σ). Grid-search involves trying pairs of (ς,σ), and the one with the best cross-validation accuracy is chosen, in which ς and σ have 15 values, respectively. ς has the value from $[2^4, 2^3, ..., 2^0, 2^{-1}, ..., 2^{-10}]$, and σ has the value from $[2^{12}, 2^{11}, ..., 2^0, ..., 2^{-2}]$.
3. Use the best pair of parameters (ς,σ) to train the whole training set and obtain a trained model.
4. Use the trained model to test the testing set.

Experiment Results and Analysis

First, we do the test on the breast cancer dataset, respectively with the RELIEF (Table 1) and CSM (Table 2) to rank the features. Since the breast cancer dataset has only two classes, the results with class-independent and class-dependent feature selection methods are the same. The second column in Tables 1 and 2 lists the number of

Table 1. The breast cancer dataset with the RELIEF measure

Feature selection approach	The number of features deleted in each of the 10 simulations (RELIEF)	Average (RELIEF)
Class-independent	3 3 4 3 3 0 0 4 1 1	2.2
Class-dependent	3 3 4 3 3 0 0 4 1 1	2.2

Table 2. The breast cancer datasets with the CSM ranking measure

Feature selection approach	The number of features deleted in each of the 10 simulations (CSM)	Average (CSM)
Class-independent	0 0 0 0 0 1 1 1 0 0	0.3
Class-dependent	0 0 0 0 0 1 1 1 0 0	0.3

deleted features in each of the 10 simulations. The third column lists the average number of deleted features.

The results in Tables 1 and 2 show us that the breast cancer dataset has very few irrelevant or redundant features. The RELIEF measure makes it have on average two features removed, while CSM on average does not make any features deleted. Second, we present the results for the E. coli dataset in Tables 3 and 4. We can see that the E. coli dataset has very different numbers of features deleted for different classes with class-dependent feature selection. For example, in Table 3, for classes 1, 3, 4, 5, and 6, the average number of features eliminated is few, while for classes 7 and 8, the number of deleted features is on average six. At last, we make the test on the Cleveland dataset. The result shows us that different classes have very different feature subsets. Class 1 has few features removed (i.e., on average 1.2 for the RELIEF (Table 4) and on average 1.9 for the CSM (Table 6)), while for classes 2, 3, 4, and 5, the number of features deleted in the 10 simulations is on average within the range of [9,12]. Compared with class-independent feature selection, the class-dependent feature selection method can show us that different classes have different feature subsets that can retain the most characteristic information.

Table 3. The E. coli datasets with the RELIEFF ranking measure

Feature selection approach	Classes	The number of features deleted in each of the 10 simulations	Average (RELIEF)
Class-independent	All classes	0 0 0 0 1 0 0 1 0 0	0.2
Class-dependent	Class 1	2 2 2 2 3 2 2 2 3 2	2.2
	Class 2	3 5 5 3 5 3 1 2 5 4	3.6
	Class 3	0 1 1 1 0 2 0 0 1 1	0.7
	Class 4	1 3 2 4 5 2 2 2 2 1	2.4
	Class 5	3 4 3 2 1 4 0 4 4 1	2.6
	Class 6	0 0 0 4 2 2 5 1 0 0	1.4
	Class 7	6 6 6 6 6 5 6 6 6 6	5.9
	Class 8	6 6 6 6 6 6 6 6 6 6	6.0

Table 4. The E. coli dataset with the CSM ranking measure

Feature selection approach	Classes	The number of features deleted in each of the 10 simulations (CSM)	Average (CSM)
Class-independent	All classes	0 0 0 0 0 1 0 0 0 0	0.1
Class-dependent	Class 1	0 0 0 0 1 1 0 0 1 0	0.3
	Class 2	0 0 0 0 0 0 0 0 0 0	0
	Class 3	0 0 1 1 1 1 0 1 1 1	0.7
	Class 4	2 4 1 4 2 4 4 5 4 4	3.4
	Class 5	2 1 3 0 0 3 1 2 2 3	1.7
	Class 6	4 4 4 4 4 2 4 4 4 4	3.8
	Class 7	6 6 6 6 6 4 6 6 6 6	5.8
	Class 8	6 6 6 6 6 6 6 6 6 6	6.0

Table 5. The Cleveland dataset with the RELIEF ranking measure

Feature selection approach	Classes	The number of features deleted in each of the 10 simulations (RELIEF)	Average
Class-independent	All classes	1 6 11 0 1 6 1 0 3	2.0
Class-dependent	Class 1	6 4 0 0 0 0 0 1 0 1	1.2
	Class 2	12 12 12 12 12 12 12 12 12 12	12
	Class 3	12 12 7 5 12 12 7 12 6 12	9.7
	Class 4	5 12 7 12 12 12 12 12 12 12	10.8
	Class 5	12 12 5 12 12 12 12 12 12 12	11.3

Table 6. The Cleveland dataset with the CSM ranking measure

Feature selection approach	Classes	The number of features deleted in each of the 10 simulations (CSM)	Average (CSM)
Class-independent	All classes	1 3 7 1 0 7 4 7 3 2	3.5
Class-dependent	Class 1	1 7 1 4 1 1 1 1 1 1	1.9
	Class 2	12 12 12 12 12 2 12 12 12 12	11
	Class 3	12 12 9 4 12 12 0 12 12 12	9.7
	Class 4	8 5 1 12 5 12 12 12 12 11	9.0
	Class 5	12 12 12 12 12 12 12 12 12 12	12

Table 7. Classification accuracies' comparison among different feature selection methods for the three data sets

Data method	Breast cancer dataset	E. coli dataset	Cleveland dataset
Without feature selection	96.49%	86.81%	55.56%
Class-independent feature selection with RELIEF	96.34%	85.71%	56.23%
Class-dependent feature selection with RELIEF	96.34%	86.31%	58.93%
Class-independent feature selection with CSM	96.49%	86.61%	56.23%
Class-dependent feature selection with CSM	96.49%	86.91%	58.61%

In Table 7, we present the classification accuracies, without feature selection, with two ranking measures and two feature selection methods (class-dependent and class-independent). The proposed feature selection method and the constructed SVM classifier have improved the accuracy on average 2% on the Cleveland dataset. For the other two biomedical datasets, there are few improvements on the classification accuracies. This can be understood in two ways: one is because the two datasets have less noise themselves, and the other reason is that those removed features in fact do not disturb the classification.

Although it may be unfair to compare our results with published results because of different classifiers, training methods, training data, and so forth, we still list some published results on the breast cancer dataset (Table 8) from Ruiz,

Table 8. A published result for the breast cancer datasets

Method	Without feature selection (C4.5)	SOAP with C4.5	RELIEF with C4.5
Classification accuracy	95.01%	94.84%	95.02%

Aguilar-Ruiz, and Riquelme (2002). They used selection of attributes by projection (SOAP) and the RELIEF method to select features (class-independent) and used C4.5 as the classifier. The accuracy is the average value of the 10 tenfold cross-validation runs. There are some published results available for the Cleveland dataset, but those classification results are about discriminating only two classes (Jin & Zhang, 2005) of the original five classes.

CONCLUSION

In this chapter, we demonstrated the proposed novel SVM with class-dependent feature selection and classification. For feature selection, we adopt the RELIEF weight measure and class separability measure (Fu & Wang, 2003) to evaluate the features' importance and select the optimal feature subset from the ranking list for each class through the classifier SVM. For the classification on the class-dependent feature subsets, we constructed a novel SVM classifier that can accommodate class-dependent feature subsets. The experimental results for the three biomedical datasets (Newman et al., 1998) show that each class has different feature subsets that keep the representative features for classifying this class from the other classes, and the novel SVM can improve or at least maintain the classification accuracy after doing the feature selection. Since the three datasets we experimented on have few noises, the proposed class-dependent feature selection cannot completely show its advantage, nor can the novel SVM classifier. However, we believe that in real life, there will be some data that have many noises. For those data, our class-dependent feature selection method will more effectively select class-dependent feature subsets than class-independent feature selection methods so the novel SVM classifier can realize a good classification.

We selected features in terms of the RELIEF and CSM ranking measure and did not deal with the redundant features. Although the existence of redundant features in the feature subset will not degrade the classification, the feature subset obtained will not be the simplest (minimum) one. Hence, in the future work, we need to consider about how to determine the redundancy of the features and further simplify the feature subsets. At the same time, we should notice that selecting class-dependent feature subsets involves the necessity of considering a feature subset for each class, which is computationally more expensive than selecting class-independent feature subsets. However, the extra computational cost may be worthwhile in certain applications where improvements of accuracy are very important and meaningful.

REFERENCES

Almauallium, H., & Dietterich, T.G. (1991). Learning with many irrelevant features. *Proceedings of Ninth National Conference on Artificial Intelligence*, 2, 547–552.

Bins, J. (2000). *Feature selection from huge sets in the context of computer vision* [doctoral dissertation]. Colorado State University.

Blum, A.L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence,* 245–271.

Bottou, L., et al., (1994). Comparison of classifier methods: A case study in handwriting digit recognition. *International Conference of Pattern Recognition (ICPR-94)*, 77–87.

Caurana, R.A., & Freitag, D. (1994). Greedy attribute selection. *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, 28–36.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: A library for support vector machines. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libsvm

Devijver, P.A., & Kittler, J. (1982). *Pattern recognition: A statistical approach*. New York: Prentice Hall.

Doak, J. (1992). *An evaluation of feature-selection methods and their application to computer security* (Tech. Rep. CSE-92-18). Davis: University of California, Department of Computer Science.

Fu, X.J., & Wang, L.P. (2002). *A GA-based novel RBF classifier with class-dependent features*. Proceedings of the 2002 Congress on Evolutionary Computation, 2, 1890–1894.

Fu, X.J., & Wang, L.P. (2003). Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, 33*(3), 399–400.

Gilad-Bachrach, R., Navot, A., & Tishby, N. (2004). Margin based feature selection—theory and algorithms. *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Gunn, S.R., Ben-Hur, A., & Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Proceedings of Advances in Neural Information Processing Systems (NIPS 2004)*, Vancouver, Canada.

Hsu, C.W., Chang, C.C., & Lin, C.J. (2003). *A practical guide to support vector classification*. Taipei, Taiwan: National Taiwan University, Department of Computer Science and Information Engineering.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Network, 13*, 415–425.

Hua, S., & Sun, Z. (2001). A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology, 308*, 397–407.

Jin, B., & Zhang, Y.Q. (2005). Support vector machines with evolutionary feature weights optimization for biomedical data classification. *Proceedings of the Soft Computing for Real World Applications, Annual Meeting of the North America Fuzzy Information Processing Society (NAFIPS05)*, Ann Arbor, Michigan, 177–180).

John, G.H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, Portland, Oregon, 367–370.

Kira, K., & Rendell, L.A. (1992). The feature selection problem: Traditional methods and a new algorithm. *Proceedings of 10th National Conference on Artificial Intelligence*, Park, California, 129–134.

Kohavi, R. & John, G. H. (1998) The wrappers approach. In Huan Liu & Hiroshi Motoda (Eds.), *Feature extraction, construction and selection: A data mining perspective* (pp. 35-46). Springer.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proceedings of the 13th International Conference on Machine Learning (ML),* Bari, Italy, 284–292.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. *Proceedings of the European Conference on Machine Learning (ECML_94),* Berlin, Heidelberg, 171–182.

Kressel, U. (1999). Pair-wise classification and support vector machines. *Advances in kernel methods: Support vector learning* (pp. 255–268). Cambridge, MA: MIT Press.

Langeley, P., & Sage, S. (1994a). *Oblivious decision trees and abstract cases.* Working Notes of the AAAI94 Workshop on Case-Based Reasoning, Seattle, Washington, 113–117.

Langeley, P., & Sage, S. (1994b). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence,* Seattle, Washington, 399–406.

Liu, H., & Motoda, H. (1998). Feature extraction, construction, and selection. *A Data Mining Perspective.* Kluwer Academic Publisher.

Moore, A.W., & Lee, M.S. (1994). Efficient algorithms for minimizing cross validation error. *Proceedings of the Eleventh International Conference on Machine Learning,* New Brunswick, New Jersey, 190–198.

Newman, D.J., Hettich, S., Blake, C.L., & Merz, C.J. (1998). *UCI repository of machine learning databases.* Irvine, CA: University of California, Department of Information and Computer Science. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Oh, I.S., Lee, J.S., & Suen, C.Y. (1998). Using class separation for feature analysis and combination of class-dependent features. *Proceedings of the Fourteenth International Conference on Pattern Recognition, 1,* 453–455.

Oh, I.S., Lee, J.S., & Suen, C.Y. (1999). Analysis of class separation and combination of class-dependent features for handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21,* 1089–1094.

Platt, J.C., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin DAGs for multi-class classification. *Advances in Neural Information Processing Systems, 12,* 547–553.

Ruiz, R., Aguilar-Ruiz, J.S., & Riquelme, J.C. (2002). SOAP: Efficient feature selection of numeric attributes. *Proceedings of the Iberoamerican Conference on Artificial Intelligence, IBERAMIA'02,* 233–242.

Vapnik, V. (1998). *Statistical learning theory.* New York: Wiley.

Vert, J.–P. (2001). Introduction to support vector machines and applications to computational biology. *Proceedings of the Seminar on SVM in Bioinformatics.*

Wang, L.P. (Ed.). (2005). *Support vector machines: Theory and applications.* New York: Springer.

Wang, L.P., Chu, F., & Xie, W. (2007). Accurate cancer classification using expressions of very few genes. *IEEE Transactions on Computational Biology and Bioinformatics, 1*(1), 40-53.

Wang, L.P., Chu, F., & Xie, W. (Accepted 2006). Accurate cancer classification using expressions of very few genes. *IEEE Transactions on Bioinformatics and Computational Biology.*

Wang, L.P., & Fu, X. J. (2005). *Data mining with computational intelligence.* Berlin: Springer.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *Advances in Neural Information Processing Systems, 13,* 668–674.

Chapter XIV
A Unified Approach to Support Vector Machines

Alistair Shilton
The University of Melbourne, Australia

Marimuthu Palaniswami
The University of Melbourne, Australia

ABSTRACT

This chapter presents a unified introduction to support vector machine (SVM) methods for binary classification, one-class classification, and regression. The SVM method for binary classification (binary SVC) is introduced first, and then extended to encompass one-class classification (clustering). Next, using the regularized risk approach as a motivation, the SVM method for regression (SVR) is described. These methods are then combined to obtain a single unified SVM formulation that encompasses binary classification, one-class classification, and regression (as well as some extensions of these), and the dual formulation of this unified model is derived. A mechanical analogy for binary and one-class SVCs is given to give an intuitive explanation of the operation of these two formulations. Finally, the unified SVM is extended to implement general cost functions, and an application of SVM classifiers to the problem of spam e-mail detection is considered.

INTRODUCTION

Support vector machines (SVMs) are a class of nonlinear learning algorithms that may be applied to many problems, including but not limited to binary classification, one-class classification (anomaly detection), and regression (function approximation). Over the last decade, SVMs have gained popularity due to their ability to tackle complex, highly nonlinear problems in a consistent, structured manner, while simultaneously avoiding problems of overfitting on more simple problems (see, for example, Wang, 2005).

SVMs work by implicitly mapping inputs into what is known as feature space, in which a linear max-margin classification or linear, flattened

regression method is applied to the problem. By using the kernel trick, SVMs are able to avoid any direct use of the feature map itself, allowing them to utilize extremely high (or even infinite) dimensional implicitly defined feature spaces without difficulty.

The aim of this chapter is to provide a unifying framework for the support vector classifier (C-SVC), one-class support vector classifier (one-class C-SVC), and support vector regressor (ε-SVR) framework, highlighting the similarities and connections among them. In so doing, we are incidentally able to introduce a simple extension to the standard ε-SVR method, whereby inequality constraints may be included directly and naturally in the training set of the unified SVM model. To give a more intuitive feel for the dual SVM training problem, we describe a mechanical analogy due to Burges (1998), which may be used to understand the role of the dual variables $\boldsymbol{\alpha}$ in the C-SVC training problem. After introducing this analogy, we show how it may be extended to describe the one-class C-SVC in an intuitive manner in terms of forces acting on a *decision sheet* in feature space.

Throughout this chapter, column vectors will be written in lower-case bold (e.g., $\mathbf{a}$,$\mathbf{g}$) and matrices in uppercase bold (e.g., $\mathbf{K}$); $\mathbf{1}$ will be used to indicate a vector every element of which is 1, and $\mathbf{I}$ is defined to be the identity matrix. We use $\mathbb{Z}_N$ to indicate the integers modulo N (so $\mathbb{Z}_N = \{0,1,\ldots,N-1\}$), $\mathbb{R}^1$ the set of positive real numbers, and $\mathbb{Z}^1$ the set of positive integers. Indices are assumed to range from 0,1,… unless otherwise stated.

SUPPORT VECTOR CLASSIFIERS

Let us begin by considering the problem of binary classification (Cortes & Vapnik, 1995; Burges, 1998). Suppose we have some set of objects X (e.g., images of faces, e-mail content, or MRI images of a human brain) that for technical and practical reasons (not least of which is the requirement that objects x be in a format a computer can process) we shall assume is Lebesgue measurable. We will call this set *input space*.

Suppose that each object $x \in X$ in our set belongs to a class $d \in D$. Such classes may include "this e-mail is spam," "this student is fast asleep," or "this is an image of the Scarlet Pimpernel," but more conventionally, and as we are dealing only with binary (two-class) classification, are (rather prosaically) defined to be $D = \{-1, +1\}$.

Our task is to construct a machine (by which we mean a computer program) that, given an object $x \in X$, can assign a class $d \in D$ to this object such that the assignment is correct, if not always, then as often as possible or practical. To attempt this, however, some information about the underlying objects is required. In the context of support vector classifiers (SVCs), we assume that this information is in the form of a training set (Haykin, 1999):

$$\Theta = \{(x_0, d_0), (x_1, d_1), \ldots, (x_{N-1}, d_{N-1})\}$$
$$x_i \in X \ \ \forall i \in \mathbb{Z}_N$$
$$d_i \in D \ \ \forall i \in \mathbb{Z}_N$$

wherein each pair (x_i,d_i) represents an object $x_i \in X$ and a classification $d_i \in D$ for this object, which has been provided to us by an expert of some description. If, for example, we are dealing with images of rocks and our classes –1 and +1 represent, respectively, "igneous rock" and "sedimentary rock," then our expert might be a friendly geologist. In any case, based on this training set, our aim is to construct a classifier that is able to classify objects that may or may not have been contained in the training set, and to do so with the best possible (or practical) accuracy.

The approach to this problem, which we will be describing in this section, is known as the support vector machine method for binary classification, or support vector classification (C-SVC, where the prefix C is used to differentiate between this

formulation and a reasonably common variant known as *v*-SVC (Herbrich & Weston, 1999) for short. The title support vector machine actually describes a range of algorithms that may be applied to many problems, not just binary classification, some of which will be covered later in this chapter. However, most (if not all) SVM methodologies share two main characteristics; namely:

1. The actual work of constructing the classifier is done in *feature space*, not input space, where feature space and input space are linked by the (implicit)[1] *feature map* $\varphi : X \to \mathbb{R}^{d_H}$ from input space X to a (possibly infinite dimensional) feature space $\mathbb{R}^{d_H}$.
2. The *trained machine* g is defined by a linear function g of position in feature space, having *universal approximator* form:

   $$g(x) = \mathbf{w}^T \varphi(x) + b$$

   where the weight vector $\mathbf{w} \in \mathbb{R}^{d_H}$ and bias $b \in \mathbb{R}$ characterize the trained machine.

Now it may seem odd that we are interested in functions of the form $g:X \to \mathbb{R}$ rather than functions of the form $f:X \to D$. However, a function of the latter type may be obtained from the former by simply taking the sign of the result, so there is no need to worry. Given a training set Θ and an implicitly defined feature map $\varphi : X \to \mathbb{R}^{d_H}$, our aim is then to find $\mathbf{w}$ and b so that g may take the place of our friendly (but not always available) expert when classifying objects $x \in X$. The resulting classifier is called the *trained machine*, and the process of obtaining it *training.*

The Separable Case

We will begin by assuming that it is possible to achieve perfect classification of the training set using some function $g:X \to \mathbb{R}$ in universal approximator form. This is known as the *separable case*, and while it is clearly not always achievable, it is nevertheless a useful starting point, as it allows us to better illustrate the underlying principles of the C-SVC method.

Note that the trained machine $g:X \to \mathbb{R}$ specified by the pair $(\mathbf{w},b)$ defines a decision surface $S(\mathbf{w},b) \cong \mathbb{R}^{d_H - 1}$ in feature space given by:

Figure 1: Examples of possible decision surfaces in feature space ($d_H = 2$). Key: class +1 training vectors = ○, class –1 training vectors = □.

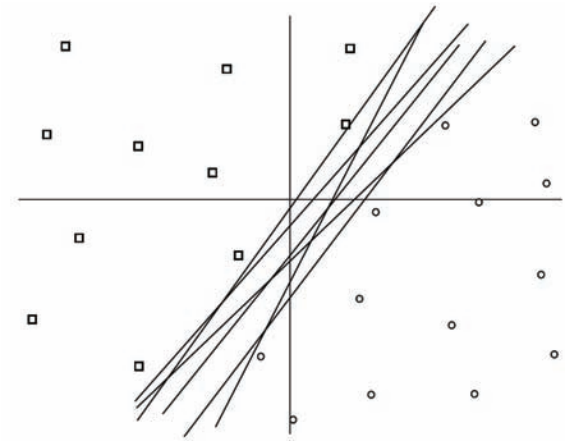
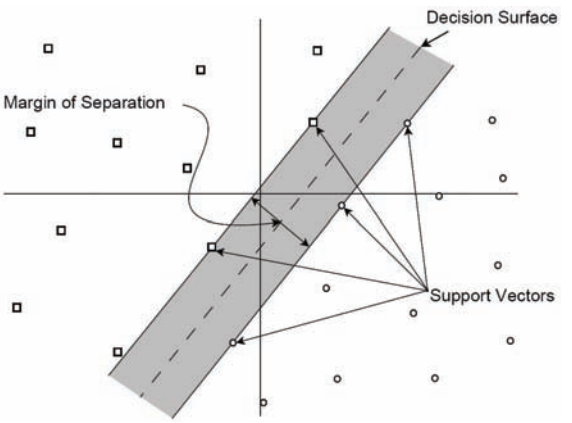


Figure 2. Optimal hyperplane selection via max-margin. Key: class +1 training vectors = ○, class –1 training vectors = □.



$$S(\mathbf{w},b)=\left\{\phi\in\mathbb{R}^{d_H}\,\middle|\,\mathbf{w}^T\phi+b=0\right\}$$

which is a hyperplane in feature space. This decision surface is said to separate the training set Θ if $d_i g(x_i)>0\ \forall(x_i,d_i)\in\Theta$, as illustrated in Figure 1.

Now it is clear from these expressions that for all $\kappa\in\mathbb{R}^+$, the decision surfaces $S(\kappa\mathbf{w},\kappa b)$ and $S(\mathbf{w},b)$ will be identical; moreover, if $S(\mathbf{w},b)$ separates the training set, then so too will $S(\kappa\mathbf{w},\kappa b)$ So, without loss of generality, we may restrict our search to those parameters $(\mathbf{w},b)$ for which:

$$d_i g(x_i)\geq 1\ \forall i\in\mathbb{Z}_N$$

Of course, if the training set is separable, there will still be infinitely many decision surfaces satisfying these constraints. In the C-SVC theory, the optimal decision surface is defined as the decision surface that has the maximum perpendicular distance between itself and the images of those training vectors lying closest to it in feature space (called the support vectors) that is equidistant from the support vectors of both classes, as shown in Figure 2. It may be seen that the shortest Euclidean distance $r(x_i)$ between the image of any training vector x_i and a decision surface in feature space is:

$$r(x_i)=\frac{|g(x_i)|}{\|\mathbf{w}\|_2}$$

and hence, the perpendicular distance between the decision surface and the training point(s) of a particular class lying closest to it is:

$$\tau_{\geq}=\min_{i\in\mathbb{Z}_N,d_i=+1} r(x_i)$$
$$\tau_{\leq}=\min_{i\in\mathbb{Z}_N,d_i=-1} r(x_i)$$

The *margin of separation* is the sum of these distances, $\rho=\tau_{\geq}+\tau_{\leq}$. The condition that the decision surface be equidistant from the support vectors of either class implies that $\tau_{\geq}+\tau_{\leq}$. It follows that for any decision surface satisfying the equidistance requirement, $(\mathbf{w},b)$ may be scaled so that:

$$\rho=\frac{2}{\|\mathbf{w}\|_2} \qquad (1)$$

Formally, the problem of finding the classifier $g:X\rightarrow\mathbb{R}$ that maximizes the margin of separation (1) is equivalent to finding a pair $(\mathbf{w},b)$ that solves the optimization problem:

$$\min_{\mathbf{w},b} R_0(\mathbf{w},b)=\frac{1}{2}\mathbf{w}^T\mathbf{w} \qquad (2)$$
$$\text{such that:}\quad d_i\left(\mathbf{w}^T\boldsymbol{\varphi}(x_i)+b\right)\geq 1\ \forall i\in\mathbb{Z}_N$$

which is called the *primal form* of the C-SVC training problem (separable case). It should be noted that one does not normally attempt to solve this problem directly, as it has a complex constraint set and may be very high dimensional if d_H is large. Instead, the so-called dual form of this problem is solved, as will be described later in this chapter.

Example: XOR Gate

Let us briefly consider a trivial yet instructive example; namely, the standard XOR gate example. In this case, our C-SVC is required to mimic a 2 input exclusive OR logic gate. We define our input space to be $X=\mathbb{R}^2$, where each axis represents one input to the gate (negative being logic 0, positive logic 1), and define our class –1 and +1 to be an outputs of logic level 0 and logic level 1, respectively.

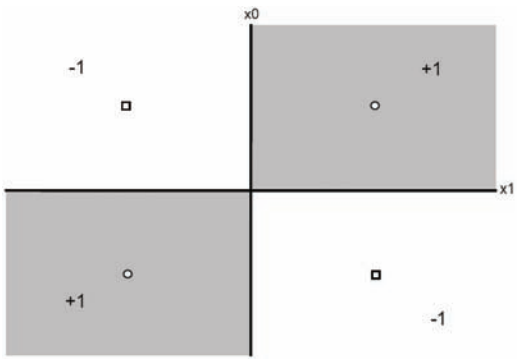
Our training set is shown in Box 1. Using the feature map:

$$\boldsymbol{\varphi}(\mathbf{x})=\begin{bmatrix}x_0\\x_1\\x_0x_1\end{bmatrix}$$

Box 1.

$$\Theta = \left\{\left([-1,-1],+1\right),\left([-1,+1],-1\right),\left([+1,-1],-1\right),\left([+1,+1],+1\right)\right\}$$

Figure 3. XOR gate example, viewed from input space. Key: class +1 training vectors = ○, class –1 training vectors = □



the primal C-SVC training problem becomes:

$$\min_{\mathbf{w},b} R_0(\mathbf{w},b) = \frac{1}{2}w_0^2 + \frac{1}{2}w_1^2 + \frac{1}{2}w_2^2$$

such that:
$$-w_0 - w_1 + w_2 + b \geq 1$$
$$+w_0 - w_1 + w_2 + b \geq 1$$
$$-w_0 + w_1 + w_2 + b \geq 1$$
$$+w_0 + w_1 + w_2 + b \geq 1$$

which has a unique solution $\mathbf{w} = [0,0,1]$, $b = 0$, giving a trained machine:

$$g(\mathbf{x}) = x_0 x_1$$

which, in this case, exactly duplicates the real XOR gate (recalling that we are only interested in the *sign* of *g*, not the magnitude), as shown in Figure 3.

Of course, this example is somewhat artificial insofar that if we had selected a different feature map there is no guarantee that the optimization problem would have had any solution at all. Furthermore, the training set for this example trivially covers all possible combinations and is noiseless and free from errors. In general, neither of these assumptions may be taken for granted. This motivates us to extend the C-SVC method in the following section.

The Inseparable Case

If the training classes are not separable, it is necessary to relax the inequalities in (2) using slack variables and modify the cost function to penalize any failure to meet the original (strict) inequalities. Using the standard (linear) penalty function (Cortes & Vapnik, 1995), the primal form of the C-SVC training problem becomes:

$$\min_{\mathbf{w},b,\xi} R_1(\mathbf{w},b,\xi) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\mathbf{1}^T\xi \qquad (3)$$

such that:
$$d_i\left(\mathbf{w}^T\boldsymbol{\varphi}(x_i) + b\right) \geq 1 - \xi_i \quad \forall i \in \mathbb{Z}_N$$
$$\xi \geq \mathbf{0}$$

where the constant $C \in \mathbb{R}^+$ controls the trade-off between empirical risk minimization (and potential overfitting if C is large) and margin maximization (and potential underfitting if C is small), as will be discussed shortly. This is the primal form of the C-SVC training problem in the nonseparable (general) case. Once again, it

should be noted that the dual form of this problem described later, not the primal form already presented, is usually solved.

Figure 4 shows a graphical representation of this formulation. As seen in this diagram, the training vectors x_i may be split into four categories; namely:

1. $d_i\, g(x_i)>1$: Nonsupport vector
2. $d_i\, g(x_i)=1$: Boundary support vector
3. $d_i\, g(x_i)\in(0,1)$: Correctly classified error (support) vector
4. $d_i\, g(x_i)\leq 0$: Incorrectly classified error (support) vector

Support Vector Machines and the Risk Bound

The previous section contained some rather cryptic allusions to the connection between the parameter C and the problems of overfitting and underfitting, which we will now clarify. The discussion in this section centers on the concept of the *capacity* of a particular set of classifiers and, in particular, the set of possible functions $g:X\rightarrow\mathbb{R}$ from which the trained machine is selected. The following section is by no means a complete introduction to the theory of risk bounds, structural risk minimization methods (of which the SVM is but one example), and related subjects, but rather attempts to summarize the more salient points. For a more detailed introduction, see, for example, Kecman (2001).

Before proceeding, we must introduce some terms. Let us suppose that objects $x\in X$ are always drawn according to some distribution $P(x)$ in an independent, identically distributed manner. The (actual) *risk* associated with a classifier $g:X\rightarrow\mathbb{R}$ characterized by $\mathbf{w}$ and b is defined to be:

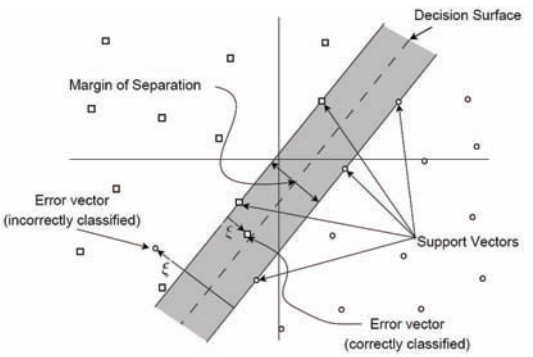
$$R(\mathbf{w},b)=\int_{x\in X} c\left(g(x),\hat{g}(x)\right)dP(x)$$

where $c:\mathbb{R}^2\rightarrow\mathbb{R}^+\cup\{0\}$ is some measure of the "goodness" of the estimate provided by g known as the cost function, and $\hat{g}:X\rightarrow D$ maps an object x to its true classification d. The cost function used in standard C-SVC theory is:

$$c(\delta,d)=\max(0,1-d\delta)$$

Now, for any classifier parameterized by $\mathbf{w}$ and b, the risk $R(\mathbf{w},b)$ provides a direct measure of the performance of that classifier—essentially, the smaller the risk, the better the classifier. Ideally, we would like to directly minimize this risk. However, as we know neither $P(x)$ nor $\hat{g}$, we must rely on the training set Θ to approximate $R(\mathbf{w},b)$. This approximation is known as the empirical risk:

Figure 4. Optimal hyperplane selection via max-margin (inseparable case). Key: class +1 training vectors = ○, class –1 training vectors = □.



$$R_{emp}\left(\mathbf{w},b\middle|\Theta\right)=\frac{1}{N}\sum_{i\in\mathbb{Z}_N}c\left(g\left(x_i\right),d_i\right)$$
$$=\frac{1}{N}\mathbf{1}^T\boldsymbol{\xi}$$

which is precisely equivalent to the penalty term in our primal (3), neglecting the scale factor C.

However, as we will see shortly, direct minimization of this empirical risk $R_{emp}(\mathbf{w},b|\Theta)$ is not guaranteed to lead to minimization of the actual risk $R(\mathbf{w},b)$ due to what is known as *overfitting*. This is closely linked to the concept of the *capacity* of the set of possible classifiers *g*. Loosely speaking, the capacity of a set of functions is a measure of their ability to describe complex relationships. If we start with a set of functions with too much capacity, our C-SVC, once trained, may be able to classify the vectors from its training set with near perfect accuracy. However, it may do so by learning inconsequential information. So, for example, it may differentiate spam and nonspam e-mails by simply recalling the precise wording of all the spam e-mails in the training set, and then classifying any e-mail not containing this text as nonspam e-mail. On the other hand, if we start with a set of functions without enough capacity, the resulting classifier may not be able to differentiate between the two classes at all well, as it may not even be able to pick up on the more relevant differences between messages, let alone the more minor, inconsequential differences. In the former case, we say that our classifier is *overfitted* to the training set, and in the latter case, we say it is *underfitted*. But in either case, the performance of the trained machine is likely to be poor. Ideally, we want to arrive at a compromise between these two extremes, whereby the set of possible classifiers has enough capacity to encapsulate the necessary information required for its task, but not so much that it will encapsulate not only this information but also irrelevant and misleading details.

One measure of the capacity of a set of functions is the Vapnik-Chervonenkis (VC) dimension. Before defining the VC dimension, however, it is necessary to define the concept of shattering of points in *X*. A set of *l* distinct points (vectors) is said to be *shattered* by a set of functions if, for each of the 2^l possible ways of labeling the points in an arbitrary binary manner, there exists a member of the set of functions that can correctly assign these labels. Based on this, the *VC dimension h* of a set of binary functions is defined to be the maximum number of distinct points *h*, which can be shattered by that set of functions (which, it is important to note, does not imply that any arbitrary set of *h* points in *X* may be shattered by this set of functions). The VC dimension is important in the present context due to the following theorem (the so-called risk bound) (Vapnik, 1995):

Theorem 1: For any $0<\eta\leq 1$ there is a probability of $1-\eta$ that the following bound will hold:

$$R\left(\mathbf{w},b\right)=R_{emp}\left(\mathbf{w},b\middle|\Theta\right)+\sqrt{\frac{h\left(\log\left(\frac{2N}{h}\right)-\log\left(\frac{\eta}{4}\right)\right)}{N}}$$

The second term on the right in this expression is called the *VC confidence*. If *h* is too small, then the empirical risk $R_{emp}(\mathbf{w},b|\Theta)$ may be large, and so the actual risk $R(\mathbf{w},b)$ may be large. Conversely, if *h* is too large, the VC confidence will be large, and so the previous bound will be ineffectual in limiting the size of the actual risk. Ideally, *h* should be somewhere between these two extremes and moreover chosen to minimize the risk bound.

For the C-SVC, the VC dimension *h* and the margin of separation ρ are connected by the following theorem (Burges, 1998):

Theorem 2: Let *D* denote the diameter of the smallest ball in feature space containing all training vector images $\boldsymbol{\varphi}(x_i)$. Then the subset of possible classifiers *g* with a margin of separation of at least ρ has a VC dimension *h* bounded previously by:

$$h \le \min\left(\left\lceil \frac{D^2}{\rho^2} \right\rceil, d_H\right) + 1$$

Using this, and assuming that $\lceil D^2 / \rho^2 \rceil \le d_H$, the risk bound may be rewritten in terms of the minimum margin of separation ρ as follows in Box 2.

Now, the primary cost function (3) may be rewritten:

$$R_1(\mathbf{w}, b, \boldsymbol{\xi}) = CR_{emp}(\mathbf{w}, b|\Theta) + \frac{2}{\rho^2}$$

from which it may be seen that $R_1(\mathbf{w}, b, \boldsymbol{\xi}) \approx CR_{emp}(\mathbf{w}, b|\Theta)$ for very large values of C; and $R_1(\mathbf{w}, b, \boldsymbol{\xi}) \approx \frac{2}{\rho^2}$ for very small values of C. It follows that for very large values of C minimization of the primal cost is essentially equivalent to empirical risk minimization. Subsequently, there is nothing to stop the second term in the risk bound from "blowing up" (becoming very large), resulting in poor generalization performance through overfitting. Likewise, if C is very small, then minimizing the primal cost will maximize ρ, which will minimize the VC confidence, but with nothing to rein in the empirical risk, the total risk bound is once again prone to "blowing up," resulting in poor generalization performance, this time due to underfitting.

If, however, we chose an appropriate value for C (which is typically found using some form of cross-validation), then we may trade off empirical risk minimization (overfitting) and VC dimension minimization (underfitting) to achieve minimization of the total risk bound, thereby minimizing the bound on the actual risk to give optimal performance. The upshot of this is that C-SVCs are able, within reason, to overcome the usual capacity selection problem encountered with many other classifiers. The inclusion of the max-margin term allows us to use feature maps with large feature space dimension d_H (and hence potentially large capacity) without worrying overly about the problems of overfitting.

ONE-CLASS SUPPORT VECTOR CLASSIFIERS

The C-SVC, as described in the previous section, is a remarkably useful classification technique *if* we have a good training set containing training samples from both the +1 and –1 classes. Consider, however, a network intrusion detection scheme. In this case, we will likely have a good range of "condition normal" training vectors, which we will arbitrarily label class +1. However, there is a good chance that we will have very few (if any) training vectors for our "intruder alert" (-1) class for both practical and theoretical reasons. For example, it would be foolish indeed to allow hackers to enter a non-honey-pot computer system simply to gather training data on what hacks "look like," and in any case, new attacks surface on a regular basis, so even if we have "–1" training samples, there is a good chance they will not match new attacks.

For this reason, it is often useful to have a classifier that only requires points from one class for training purposes. Let us arbitrarily suppose that all training points in our training set Θ are of class +1. The question is, if we wish to apply C-SVC methods in this situation, how may we modify

Box 2.

$$R(\mathbf{w}, b) = R_{emp}(\mathbf{w}, b|\Theta) + \sqrt{\frac{\left(\left\lceil \frac{D^2}{\rho^2} \right\rceil + 1\right)\left(\log\left(\frac{2N}{\left\lceil \frac{D^2}{\rho^2} \right\rceil + 1}\right) - \log\left(\frac{\eta}{4}\right)\right)}{N}}$$

the C-SVC primal problem (3) to make up for our lack of any training samples of class –1?

The answer to this question lies with the bias b in the universal approximator form of g. Recall that:

$$g(x) = \mathbf{w}^T \boldsymbol{\varphi}(x) + b$$

If our training set Θ contains only samples from class +1, then it is not hard to see that the solution to (3) is simply $\mathbf{w}=\mathbf{0}$, $\boldsymbol{\xi}=\mathbf{0}$, and b=1, and hence, $g(x)=b=+1$ for all $x \in X$. This does not actually define a decision surface in the usual sense and is clearly not a satisfactory solution to the one-class C-SVC problem. However, if we add a new term b to the primal cost, then in combination with usual factors of empirical risk minimization and margin maximization, the one-class C-SVC will attempt to choose the most negative (or least positive) allowable bias b.

Specifically, the primal form of the one-class C-SVC training problem is (Manevitz & Yousef, 2001; Scholkopf, Platt, Shawe-Taylor, Smola, & Williamson, 1999):

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} R_1(\mathbf{w},b,\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\mathbf{1}^T\boldsymbol{\xi} + b \\ &\text{such that:} \quad d_i\left(\mathbf{w}^T\boldsymbol{\varphi}(x_i) + b\right) \geq -\xi_i \;\; \forall i \in \mathbb{Z}_N \\ &\qquad\qquad \boldsymbol{\xi} \geq \mathbf{0} \end{aligned} \tag{4}$$

where it is customary to set $C = \frac{1}{\nu}$ and also to replace b with $\rho = -b$ (where ρ is *not* the margin of separation in this case). However, none of these conventions is strictly necessary, and by leaving the form of the one-class C-SVC as similar as possible to the usual C-SVC, we hope to underline the fact that this is just a simple variation of the C-SVC primal. For technical reasons, we must in this case restrict our choice of C to the range $C \geq 1$.

An alternative approach to the one-class SVC is to attempt to construct a sphere of minimal radius in feature space (Lee & Lee, 2005). However, while there are some advantages to this approach, it does not fit into the unified method we are constructing in this chapter, and so we will not go into detail here.

SUPPORT VECTOR REGRESSORS

The final type of SVM under consideration here is the support vector regressor (Drucker, Burges, Kaufman, Smola & Vapnik, 1997; Smola, 1996; Smola & Scholkopf, 1998; Vapnik, Golowich & Smola, 1997) (or ε-SVR for short, where the prefix ε is included to differentiate between the formulation introduced here and a variant known as the ν-SVR (Scholkopf, Bartlett, Smola & Williamson, 1999). ε-SVRs tackle the problem of function approximation and find application in a huge variety of areas, including weather prediction, stock market trend analysis, and adaptive control systems, to name a few.

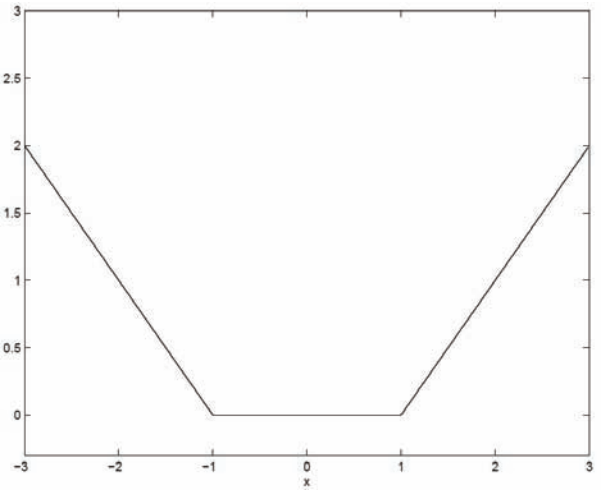
To be specific, ε-SVR is concerned with the following problem. Suppose we have some system with a measurable input $x \in X$ (where we will once again assume that X is Lebesgue measurable) and a measurable real valued output $z \in \mathbb{R}$. For example, our input might be a series of stock prices for a company over the previous financial year, and z the projected growth for the next. In this case, we may wish to estimate z in advance, allowing us to make a more informed decision about whether or not the company in question would be a good investment.

Let us write the "true" output z given input x as $z=\hat{g}(x)$. The aim of ε-SVR is to construct a function of the familiar universal approximator form:

$$g(x) = \mathbf{w}^T \boldsymbol{\varphi}(x) + b$$

which approximates $\hat{g}$, noting that this is a linear function of the position of the image of $x \in X$ in feature space.

Figure 5. Vapnik's ε-insensitive cost function (ε=1 in this case)



As for the binary classification case, we are given a training set Θ:

$$\Theta = \{(x_0, z_0), (x_1, z_1), \ldots (x_{N-1}, z_{N-1})\}$$
$$x_i \in X \ \ \forall i \in \mathbb{Z}_N$$
$$z_i \in \mathbb{R} \ \ \forall i \in \mathbb{Z}_N$$

where, in this case, each pair (x_i, z_i) represents an input $x_i \in X$ to our system and the measured output $z_i \in \mathbb{R}$ of our system given this input. Based on this training set, the aim of ε-SVR is to construct a regressor that is able to mimic $\hat{g}$ as accurately as possible (or practical).

Continuing our analogy with the C-SVC, assuming objects $x \in X$ are selected according to some distribution $P(x)$, we would like to minimize the actual risk:

$$R(\mathbf{w}, b) = \int_{x \in X} c\left(g(x), \hat{g}(x)\right) dP(x)$$

where $c : \mathbb{R}^2 \to \mathbb{R}^+ \cup \{0\}$ is known as the cost function and is a measure of the error in our approximation $g(x_i)$. Traditionally, ε-SVR uses the cost function:

$$c(\delta, z) = |\delta - z|_\varepsilon$$
$$= \max(0, |\delta - z| - \varepsilon)$$

which is known as Vapnik's ε-insensitive cost (Haykin, 1999) ($\varepsilon \in \mathbb{R}^+ \cup \{0\}$ being a constant), as shown in Figure 5. However, since we know neither $P(x)$ nor $\hat{g}$, we must once again resign ourselves to dealing instead with the empirical risk, namely:

$$R_{emp}(\mathbf{w}, b | \Theta) = \frac{1}{N} \sum_{i \in \mathbb{Z}_N} c\left(g(x_i), z_i\right)$$

But if we attempt to select **w** and b by minimizing the empirical risk directly, we are once again faced with the problem of overfitting if the set of possible trained machines $g: X \to \mathbb{R}$ has too much capacity for the problem at hand. In this particular case, overfitting will manifest itself in the "bumpiness" (i.e., the rate of change of $g(x)$ as x is varied) of the trained machine g. Loosely speaking, the more rapidly g varies in response to small changes in x, the more complex the surface and the more likely that overfitting has occurred.

Motivated by these observations, rather than minimize the empirical risk directly to find **w**

and *b*, instead we minimize the *regularized* risk, which in general is defined to be:

$$R_{reg}\left(\mathbf{w},b\middle|\Theta\right)=CR_{emp}\left(\mathbf{w},b\middle|\Theta\right)+\phi\left(\mathbf{w},b\right)$$

where the *regularization term* $\phi(\mathbf{w},b)$ is some measure of the "bumpiness" of the trained machine *g*, and $C \in \mathbb{R}^+$ is a constant. Noting that if $X = \mathbb{R}^{d_L}$, then $\left|\nabla_{\mathbf{x}} g\left(\mathbf{x}\right)\right| \propto \frac{1}{2}\mathbf{w}^T\mathbf{w}$, ε-SVR use the regularization term $\phi\left(\mathbf{w},b\right)=\frac{1}{2}\mathbf{w}^T\mathbf{w}$ giving the regularized risk function:

$$R_{reg}\left(\mathbf{w},b\middle|\Theta\right)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+CR_{emp}\left(\mathbf{w},b\middle|\Theta\right)$$

In much the same way that the constant $C \in \mathbb{R}^+$ controls the trade-off between capacity minimization and empirical risk minimization in the binary classification case, the same constant $C \in \mathbb{R}^+$ in the previous expression controls the trade-off between empirical risk minimization (and possible overfitting) if C is large, and function flattening (and potential underfitting) if C is small.

The primal form of the ε-SVR training problem may be expressed in terms of slack variables as follows:

$$\min_{\mathbf{w},b,\xi,\xi^*} R_1\left(\mathbf{w},b,\xi,\xi^*\right)=\frac{1}{2}\mathbf{w}^T\mathbf{w}+\frac{C}{N}\mathbf{1}^T\left(\xi+\xi^*\right)$$

$$\text{such that:}\quad \mathbf{w}^T\boldsymbol{\varphi}\left(x\right)+b-z_i \geq -\varepsilon-\xi_i \;\; \forall i \in \mathbb{Z}_N$$

$$\mathbf{w}^T\boldsymbol{\varphi}\left(x\right)+b-z_i \leq +\varepsilon+\xi_i^* \;\; \forall i \in \mathbb{Z}_N$$

$$\xi \geq \mathbf{0}$$

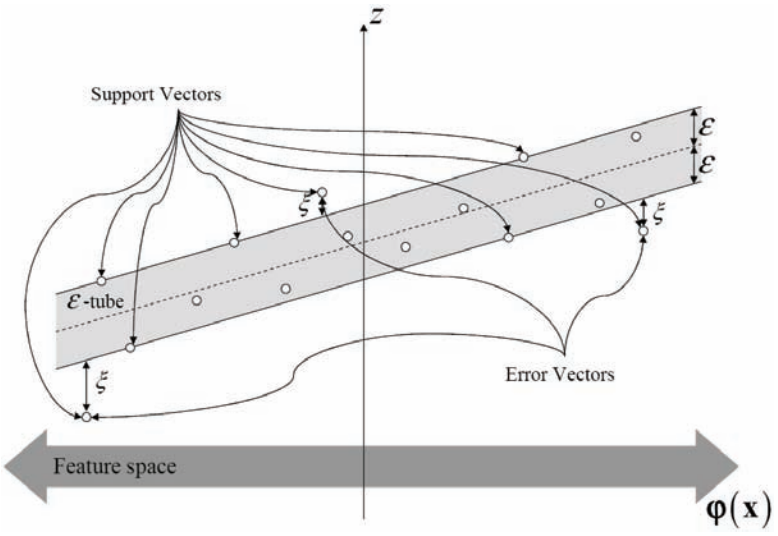
$$\xi^* \geq \mathbf{0} \qquad (5)$$

As for binary classification, it is important to note that one does not normally attempt to solve this problem directly to find **w** and *b* (the dual is solved instead). Note also that $\xi_i\xi_i^* = 0 \;\; \forall i \in \mathbb{Z}_N$ and furthermore:

$$\xi_i+\xi_i^* = \left|g\left(x_i\right)-z_i\right|_{\varepsilon} \;\; \forall i \in \mathbb{Z}_N$$

Geometrically, the ε-SVR formulation may be visualized, as shown in Figure 6. In this interpretation, feature space is extended by one dimension, namely *z*. The training points in this (extended) feature space are $(\boldsymbol{\varphi}(x_i), z_i)$, and one aims to construct an ε-tube (a tube whose width along the *z* axis is 2ε, as shown in the figure. The "tube" is made up of two parallel hyperplanes separated by a distance of 2ε along the *z* axis) such that all

Figure 6. Geometric interpretation of the standard SV regressor



(extended) training points lie inside this tube (or as close to the tube as possible if this is not achievable) and the tube itself is as flat as possible with respect to the *z* axis. The empirical risk term of the cost function works toward the former goal, and the regularization term the latter.

Example: XOR Gate Revisited

We now revisit our earlier XOR gate example. Once again, we aim to mimic a 2-input exclusive OR logic gate, assuming that –1 represents logic 0 and +1 represents logic 1, and our feature map is $\boldsymbol{\varphi}(x)=[x_0, x_1, x_0x_1]$. Let us further assume that ε=0.1 and that C is very large (effectively the same as the separable case in the C-SVC insofar as it ensures that $\boldsymbol{\xi}$=0.

Our training set is the same as found in Box 1, and hence, the ε-SVR training problem is:

$$\min_{\mathbf{w},b} R_0(\mathbf{w},b)=\frac{1}{2}w_0^2+\frac{1}{2}w_1^2+\frac{1}{2}w_2^2$$

such that:
$$-w_0-w_1+w_2+b\in[0.9,1.1]$$
$$+w_0-w_1+w_2+b\in[0.9,1.1]$$
$$-w_0+w_1+w_2+b\in[0.9,1.1]$$
$$+w_0+w_1+w_2+b\in[0.9,1.1]$$

The mathematics is slightly more complicated in this case (we will not go into details). Suffice it to say that it can be shown that this has a unique solution $\mathbf{w}$=[0,0,0.9], b=0, giving the trained machine:

$$g(\mathbf{x})=0.9x_0x_1$$

This seems reasonably close to the answer achieved previously, and indeed, if we neglect the magnitude of the output, it is precisely the same. However, as this is a regression problem, we *are* interested in the magnitude, so this solution is different. In general, it is not wise to formulate pattern classification problems as regression problems, although it is clearly possible to do so by simply setting $z_i=d_i$. The reason for this may be seen by considering what would happen if one were to include an additional training vector ([–1.5,+1],–1) in the previous training set. This should clearly not change the decision function and indeed does not in the C-SVC case. In the ε-SVR case, however, it will distort the decision function, which, while desirable in a regressor, is clearly not desirable in a classifier.

UNIFICATION AND DUALITY

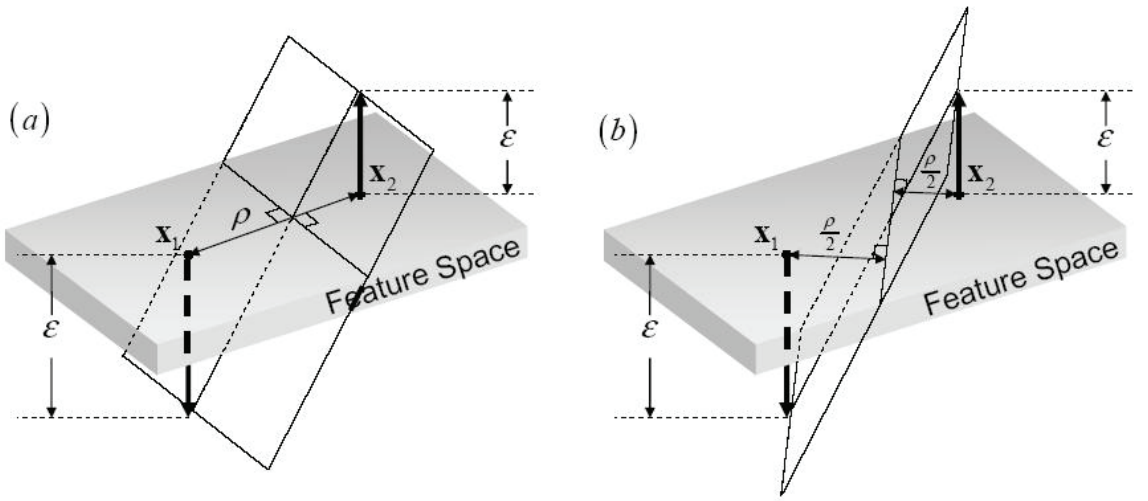
Consider all of the SVM formulations considered in this chapter up until the present. In each case, our aim is to construct a *trained machine* $g:X\rightarrow\mathbb{R}$ of the form:

$$g(x)=\mathbf{w}^T\boldsymbol{\varphi}(x)+b$$

parameterized by $\mathbf{w}$ and b. The parameters $\mathbf{w}$ and b may (in principle) be found by solving the appropriate primal training problem ((3) for C-SVC, (4) for one-class C-SVC, (5) for ε-SVR), which in all cases is a linearly constrained d_H-dimensional convex quadratic programming problem. In the present section, we take the obvious similarities between the various formulations to its logical conclusion and construct an SVM framework that encompasses binary classification, one-class classification, and regression as simply special cases of a more general, unified SVM formulation. We then construct the dual form of this unified formulation.

To understand the connection between regression and pattern classification, consider Figure 7, which shows the graph of $g(x)$ as distance above (or below) feature space in an augmented (extended by one dimension) feature space. The decision surface in this case is the intersection between the graph of $g(x)$ and feature space. For the two boundary vectors $\boldsymbol{\varphi}(x_i)$ and $\boldsymbol{\varphi}(x_j)$ shown, the perpendicular (to feature space) distance between $\boldsymbol{\varphi}(x_i)$ (or $\boldsymbol{\varphi}(x_i)$) in feature space and the graph is ε=1. If the margin of separation is ρ, then the gradient of the graph

Figure 7. Relationship between the concepts of max-margin (for the pattern recognition case) and flattening (for the regression case). (a) shows the max-margin case, and (b) an alternative, nonoptimal margin. Note that the gradient of the surface is smaller (i.e., it is "flatter")



must be $\frac{2\varepsilon}{\rho}$, and so minimizing this gradient (flattening the graph) will maximize the margin of separation ρ, and vice versa.

Unification

Let us begin by defining a universal training set:

$$\Theta = \{(x_0, d_0, z_0), (x_1, d_1, z_1), \ldots (x_{N-1}, d_{N-1}, z_{N-1})\}$$
$$x_i \in X \;\; \forall i \in \mathbb{Z}_N$$
$$d_i \in \{-1, 0, +1\} \;\; \forall i \in \mathbb{Z}_N$$
$$z_i \in \mathbb{R} \;\; \forall i \in \mathbb{Z}_N$$

and associating with it the (universal) primal SVM training problem (see Box 3) where $C \in \mathbb{R}^+$, $\varepsilon \in \mathbb{R}^+ \cup \{0\}$, $\zeta \in \mathbb{R}$ and $\chi \in \{-1,+1\}$ are constants, and we require that $\chi=-1$ if $d_i=0$ for any $i \in \mathbb{Z}_N$.

It is not too hard to see that this will reduce to the standard primal C-SVC (3), one-class C-SVC (4), or ε-SVR (5) training problem under the following special conditions:

- **C-SVC:** $\chi=1$, $\varepsilon=1$, $\zeta=0$ and $z_i = 0, d_i \in \{-1,+1\} \;\; \forall i \in \mathbb{Z}_N$.
- **One-class C-SVC:** $\chi=1$, $\varepsilon=0$, $\zeta=1$ and $z_i = 0, d_i = +1 \;\; \forall i \in \mathbb{Z}_N$.
- ε-SVR: $\chi = -1$, $\zeta=0$ and $d_i = 0 \;\; \forall i \in \mathbb{Z}_N$.

Box 3.

$$\min_{\mathbf{w},b,\xi,\xi^*} R_1(\mathbf{w}, b, \xi, \xi^*) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\left(\sum_{i\in\mathbb{Z}_N : d_i \geq 0} \xi_i + \sum_{i\in\mathbb{Z}_N : d_i \leq 0} \xi_i^*\right) + \zeta b$$

such that:
$$\mathbf{w}^T\boldsymbol{\varphi}(x) + b - z_i \geq +\chi\varepsilon - \xi_i \;, \forall i \in \mathbb{Z}_N \;\; d_i \geq 0$$
$$\mathbf{w}^T\boldsymbol{\varphi}(x) + b - z_i \leq -\chi\varepsilon + \xi_i^* \;, \forall i \in \mathbb{Z}_N \;\; d_i \leq 0$$
$$\boldsymbol{\xi} \geq \mathbf{0} \;, \forall i \in \mathbb{Z}_N \;\; d_i \geq 0$$
$$\boldsymbol{\xi}^* \geq \mathbf{0} \;, \forall i \in \mathbb{Z}_N \;\; d_i \leq 0 \qquad (6)$$

Of course, so long as we satisfy the requirement $\chi = -1$ if $d_i = 0$ for any $i \in \mathbb{Z}_N$, there is no reason to restrict ourselves to these three standard problems. For example, if we wish to include inequality constraints in the ε-SVR formulation, we may do so by including training tuples $(x_i, +1, z_i)$ to add lower bound constraints of the type $g(x_i) \geq z_i - \varepsilon$ to the problem, and likewise by including training tuples $(x_i, -1, z_i)$ to add upper-bound constraints of the type $g(x_i) \leq z_i - \varepsilon$.[2]

The Dual Formulation

As has been mentioned previously, there are two difficulties with the primal form (6) that make it difficult to solve directly. The first is the relative complexity of the constraint set (especially when compared with the dual form). The second, more serious problem is the fact that the primal problem is a d_H-dimensional quadratic problem, which places direct, practical limitations on our choice of feature map. For this reason, we construct the dual optimization problem (Fletcher, 1981), which, as we will show, is a much more convenient optimization problem.

To begin, note that (6) is a linearly constrained convex quadratic programming problem and, hence, amenable to Lagrangian methods. Hence, for each of the constraints:

$$\mathbf{w}^T \boldsymbol{\varphi}(x_i) + b - z_i \geq +\chi\varepsilon - \xi_i \text{ if } d_i \geq 0$$

we associate a Lagrange multiplier $\beta_i \geq 0$. Similarly, for each of the constraints:

$$\mathbf{w}^T \boldsymbol{\varphi}(x_i) + b - z_i \leq +\chi\varepsilon - \xi_i \text{ if } d_i \leq 0$$

we associate a Lagrange multiplier $\beta_i^* \leq 0$. For convenience, we also define $\beta_i = 0$ for all training tuples for which $d_i < 0$, and similarly $\beta_i^* \leq 0$ for all training tuples for which $d_i > 0$. Finally, we define a Lagrange multiplier $\gamma_i \geq 0$ for each constraint $\xi_i \geq 0$ (with $\gamma_i = 0$ otherwise) and a Lagrange multiplier $\gamma_i^* \geq 0$ for each constraint $\xi_i^* \geq 0$ (with $\gamma_i^* \geq 0$ otherwise). So, for each training tuple $(x_i, d_i, z_i) \in \Theta$, we have four corresponding Lagrange multipliers β_i, β_i^*, γ_i and γ_i^*.

Using this notation, the Lagrangian of (6) is:

$$\begin{aligned} L_1 = {} & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\mathbf{1}^T\boldsymbol{\xi} + \frac{C}{N}\mathbf{1}^T\boldsymbol{\xi}^* + \zeta b \\ & - \sum_{i \in \mathbb{Z}_N : d_i \geq 0} \left(\gamma_i \xi_i + \beta_i \left(\mathbf{w}^T \boldsymbol{\varphi}(x_i) + b - z_i - \chi\varepsilon + \xi_i \right) \right) \\ & - \sum_{i \in \mathbb{Z}_N : d_i \geq 0} \left(\gamma_i^* \xi_i^* + \beta_i^* \left(\mathbf{w}^T \boldsymbol{\varphi}(x_i) + b - z_i + \chi\varepsilon - \xi_i^* \right) \right) \end{aligned} \tag{7}$$

and the Wolfe dual of (6) is:

$$\begin{aligned} & \min_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \max_{\boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*} L_1 \\ & \text{such that: } \nabla_{\mathbf{w}} L_1 = \mathbf{0} \\ & \qquad \nabla_b L_1 = 0 \\ & \qquad \nabla_{\xi_i} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0 \\ & \qquad \nabla_{\xi_i^*} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0 \\ & \qquad \boldsymbol{\beta}, -\boldsymbol{\beta}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^* \geq \mathbf{0} \\ & \qquad \beta_i = \gamma_i = 0 \ \forall i \in \mathbb{Z}_N : d_i < 0 \\ & \qquad \beta_i^* = \gamma_i^* = 0 \ \forall i \in \mathbb{Z}_N : d_i > 0 \end{aligned} \tag{8}$$

which has the KKT optimality conditions:

$$\beta_i \nabla_{\beta_i} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0 \tag{9}$$
$$\beta_i^* \nabla_{\beta_i^*} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0 \tag{10}$$
$$\gamma_i \nabla_{\gamma_i} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0 \tag{11}$$
$$\gamma_i^* \nabla_{\gamma_i^*} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0 \tag{12}$$
$$\nabla_{\mathbf{w}} L_1 = \mathbf{0} \tag{13}$$
$$\nabla_b L_1 = 0 \tag{14}$$
$$\nabla_{\xi_i} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0 \tag{15}$$
$$\nabla_{\xi_i^*} L_1 = 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0 \tag{16}$$
$$\boldsymbol{\beta}, -\boldsymbol{\beta}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^* \geq \mathbf{0} \tag{17}$$
$$\beta_i = \gamma_i = 0 \ \forall i \in \mathbb{Z}_N : d_i < 0 \tag{18}$$
$$\beta_i^* = \gamma_i^* = 0 \ \forall i \in \mathbb{Z}_N : d_i > 0 \tag{19}$$

Conditions (16) - (19) imply that:

$$\mathbf{w} = \sum_{i \in \mathbb{Z}_N} \left(\beta_i + \beta_i^*\right)\boldsymbol{\varphi}\left(x_i\right) \tag{20}$$

$$\zeta = \mathbf{1}^T \left(\boldsymbol{\beta} + \boldsymbol{\beta}^*\right) \tag{21}$$

$$\beta_i = \frac{C}{N} - \gamma_i \;\; \forall i \in \mathbb{Z}_N : d_i \geq 0 \tag{22}$$

$$-\beta_i^* = \frac{C}{N} - \gamma_i^* \;\; \forall i \in \mathbb{Z}_N \;\; d_i \leq 0 \tag{23}$$

and hence, $0 \leq \beta_i \leq \frac{C}{N} \;\; \forall i \in \mathbb{Z}_N : d_i \geq 0$ and $-\frac{C}{N} \leq \beta_i^* \leq 0 \;\; \forall i \in \mathbb{Z}_N : d_i \leq 0$. Consider some training tuple (x_i, d_i, z_i) for which $d_i = 0$ (and hence, $\chi = -1$ by definition). Suppose $\varepsilon > 0$. If $\beta_i > 0$, then using the previous conditions, it follows that $\mathbf{w}^T\boldsymbol{\varphi}\left(x_i\right) + b \leq z_i - \varepsilon$, and hence, $\beta_i^* = 0$. Using similar reasoning, it may be seen that if $\beta_i^* > 0$, then $\beta_i = 0$. So if $\varepsilon > 0$, then $\beta_i\beta_i^* = 0$ and $\left|\beta_i - \beta_i^*\right| = \left|\beta_i + \beta_i^*\right|$ for all $i \in \mathbb{Z}_N$. Hence, defining $\boldsymbol{\alpha} = \boldsymbol{\beta} + \boldsymbol{\beta}^*$, we may rewrite the Wolfe dual in the following dual unified SVM training problem form (see Box 4), where:

$$l_i = \begin{cases} -\frac{C}{N} & \text{if } 0 \leq d_i \\ 0 & \text{otherwise} \end{cases}$$

$$u_i = \begin{cases} \frac{C}{N} & \text{if } 0 \geq d_i \\ 0 & \text{otherwise} \end{cases}$$

and $\mathbf{K} \in \mathbb{R}^{N \times N}$, $K_{i,j} = K(x_i, x_j)$ and $K(x,y) = \boldsymbol{\varphi}^T(x)\boldsymbol{\varphi}(y)$ is the *kernel function*, which we will discuss in detail shortly. The trained machine g may be expressed solely in terms of $\boldsymbol{\alpha}$ and b using (20) as follows:

$$g\left(y\right) = \sum_{i \in \mathbb{Z}_N} \alpha_i K\left(x_i, y\right) + b \tag{25}$$

We have chosen to leave the bias b in the previous statement of the dual problem (24). This has certain advantages (Shilton, Palaniswami, Ralph, & Tsoi, 2005), not least of which is that it results in a very simple constraint set and removes the necessity of calculating the bias b after the optimization process has been completed. However, if so desired we may make direct use of (21) to obtain the alternative dual formulation:

$$\min_{\boldsymbol{\alpha}} Q\left(\boldsymbol{\alpha}\right) = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\mathbf{z} - \chi\varepsilon\sum_{i \in \mathbb{Z}_N}\left|\alpha_i\right|$$

$$\text{such that:} \quad \mathbf{l} \leq \alpha \leq \mathbf{u}$$

$$\mathbf{1}^T\boldsymbol{\alpha} = \zeta \tag{26}$$

Where the bias b may be calculated postoptimization using:

$$b = d_i - \sum_{j \in \mathbb{Z}_N} K_{i,j}\,\alpha_j$$

This formulation is, in fact, more common in the literature. However, for the reasons stated previously (namely, that the constraint set is more complex and that the bias b must be calculated postoptimization), we favor the use of the previous, partially dual formulation (24).

The optimality conditions for (24), reexpressed from the KKT conditions (9) - (19), may now be stated. Defining:

$$\begin{bmatrix} \mathbf{g} \\ h \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix}$$

Box 4.

$$\min_{\boldsymbol{\alpha}} \max_{b} Q\left(\boldsymbol{\alpha}, b\right) = \frac{1}{2}\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix}^T \begin{bmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix}^T \begin{bmatrix} \mathbf{z} \\ \zeta \end{bmatrix} - \chi\varepsilon\sum_{i \in \mathbb{Z}_N}\left|\alpha_i\right|$$

$$\text{such that:} \quad \mathbf{l} \leq \alpha \leq \mathbf{u} \tag{24}$$

and noting that $g_i = g(x_i)\ \forall i \in \mathbb{Z}_N$, the optimality conditions of the dual SVM training problem (24) are:

$$l_i \le \alpha_i \le u_i \ \forall i \in \mathbb{Z}_N \quad (27)$$
$$g_i \ge z_i - \chi\varepsilon \ \forall i \in \mathbb{Z}_N : \alpha_i = l_i < 0 \quad (28)$$
$$g_i = z_i - \chi\varepsilon \ \forall i \in \mathbb{Z}_N : l_i < \alpha_i < 0 \quad (29)$$
$$g_i \le z_i - \chi\varepsilon \ \forall i \in \mathbb{Z}_N : \alpha_i = 0 > l_i \quad (30)$$
$$g_i \ge z_i + \chi\varepsilon \ \forall i \in \mathbb{Z}_N : \alpha_i = 0 < u_i \quad (31)$$
$$g_i = z_i + \chi\varepsilon \ \forall i \in \mathbb{Z}_N : 0 < \alpha_i < u_i \quad (32)$$
$$g_i \le z_i + \chi\varepsilon \ \forall i \in \mathbb{Z}_N : \alpha_i = u_i > 0 \quad (33)$$
$$h = \zeta \quad (34)$$

Properties of the Dual

A number of properties of the dual unified SVM training problem (24) is noteworthy. First, the dual is a convex quadratic programming problem (the convexity of the dual following directly from the convexity of the primal), meaning that it has no nonglobal minima.[3] Moreover, the quadratic form is particularly straightforward to solve, especially as we do not have to worry about becoming "stuck" in local minima.

Second, each training tuple (x_i, d_i, z_i) is uniquely associated with a single dual variable α_i. It may be seen from the optimality conditions (28)-(33) that we can classify the training vectors x_i into nonsupport, boundary (support), and error (support) vectors using the corresponding α_i value directly as follows:

- $\alpha_i = 0$ for nonsupport vectors
- $0 < |\alpha_i| \le \frac{C}{N}$ for boundary (support) vectors
- $|\alpha_i| = \frac{C}{N}$ for error (support) vectors

It follows from this that, assuming $\varepsilon > 0$, the solution $\boldsymbol{\alpha}$ is likely to be sparse, by which we mean that a proportion of the dual variables α_i will be zero. This follows from the fact that only support vectors have corresponding nonzero α_i values, and geometrically it is unlikely that all (or even a significant proportion of) training vectors will be support vectors.

Finally, let us return to the kernel function $K: X \times X \to \mathbb{R}$ defined previously. It may be noted that we do not actually need to know the exact form of the feature map $\boldsymbol{\varphi} : X \to \mathbb{R}^{d_H}$, so long as we know that it exists. It can be shown (Mercer, 1909; Cochran, 1972) that for any symmetric function $K: X \times X \to \mathbb{R}$ for all functions $\phi: X \to \mathbb{R}$ for which:[4]

$$\int_{x \in X} \phi^2(x)\,dx < \infty$$

satisfies:

$$\int_{x \in X} \int_{y \in X} \phi(x) K(x, y) \phi(y)\,dx\,dy \ge 0$$

there must exist a map $\boldsymbol{\varphi} : X \to \mathbb{R}^{d_H}$ such that:

$$K(x, y) = \sum_{n \in \mathbb{Z}_{d_H}} \varphi_n(x) \varphi_n(y)$$

(where we define $\mathbb{Z}_\infty = \mathbb{Q}$ to encompass the infinite dimensional case) is valid and converges absolutely and uniformly. This is known as Mercer's condition (Mercer, 1909).

The implication of this result is that we do not actually need to know the feature map $\boldsymbol{\varphi} : X \to \mathbb{R}^{d_H}$ or to retrieve $\mathbf{w}$ in order to use our trained machine. Indeed, given a training set Θ (be it binary classification training set, a one-class training set, a regression training set, or some hybrid) and a Mercer kernel $K: X \times X \to \mathbb{R}$, we can find the optimal decision function g by solving the dual training problem (24) to obtain $\boldsymbol{\alpha}$ and b, and then make use of the subsequent trained machine (25), all without knowledge of the feature map $\boldsymbol{\varphi} : X \to \mathbb{R}^{d_H}$ or the weight vector $\mathbf{w}$. This decoupling of the training problem dimension N and the dimensionality (and hence, to some extent, the complexity) of the implicit feature map d_H is one of the great strengths of the SVM approach. In essence, it allows us to

use very complex, potentially very high capacity feature maps without overfitting or increasing the complexity of the training algorithm.

Mercer kernels for the case $X = \mathbb{R}^{d_L}$ include:

- **Linear kernel:** $K(\mathbf{x},\mathbf{y})=\mathbf{x}^T\mathbf{y}$
- **Polynomial kernel:** $K(\mathbf{x},\mathbf{y})=(1+\mathbf{x}^T\mathbf{y})^p$, where $p \in \mathbb{Z}^+$.
- **Gaussian RBF kernel:** $K(\mathbf{x},\mathbf{y})=e^{\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\gamma}}$, where $\gamma \in \mathbb{R}^+$.
- **Multilayer perceptron:** $K(\mathbf{x},\mathbf{y}) = \tanh(\mathbf{x}^T\mathbf{y} + \beta)$, where $\beta \in \mathbb{R}$ is selected such that $K(\mathbf{x},\mathbf{y})$ satisfies Mercer's condition.
- **Vovk's real polynomial kernel:** $K(\mathbf{x},\mathbf{y})=\dfrac{(1+\mathbf{x}^T\mathbf{y})^p}{1+\mathbf{x}^T\mathbf{y}}$, where $p \in \mathbb{Z}^+$.

It may be shown that if $K_1,K_2: X \times X \rightarrow \mathbb{R}$ are Mercer kernels, $L_1,L_2: X \times X \rightarrow \mathbb{R}$ are symmetric functions, and $c, d \in \mathbb{R}^+ \cup \{0\}$, then the following will be Mercer kernels:

$$K_a(x,y) = cK_1(x,y) + dK_2(x,y)$$
$$K_b(x,y) = K_1(x,y)K_2(x,y)$$
$$K_c(x,y) = \int_{z \in X} L_1(x,z)L_2(z,y)dz$$

Special Cases

At this point, it may be useful to step back briefly from the unified SVM to consider the special cases; namely, C-SVC, one-class C-SVC, and ε-SVR, and how the dual problem and its optimality conditions look in each case.

The C-SVC

This is equivalent to the unified case with $\chi = 1$, $\varepsilon = 1$, $\zeta = 0$ and $z_i = 0, d_i \in \{-1,+1\}\ \forall i \in \mathbb{Z}_N$. Noting that $\alpha_i \geq 0\ \forall i \in \mathbb{Z}_N : d_i = +1$ and $\alpha_i \leq 0\ \forall i \in \mathbb{Z}_N : d_i = -1$ (as $l_i = 0\ \forall i \in \mathbb{Z}_N : d_i = +1$, and $u_i = 0\ \forall i \in \mathbb{Z}_N : d_i = -1$), the dual training problem may be seen to be:

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha},b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - \boldsymbol{\alpha}^T\mathbf{d}$$
$$\text{such that: } 0 \leq \alpha_i \leq \tfrac{C}{N}\ \forall i \in \mathbb{Z}_N : d_i = +1$$
$$-\tfrac{C}{N} \leq \alpha_i \leq 0\ \forall i \in \mathbb{Z}_N : d_i = -1 \quad (35)$$

which has the optimality conditions:

$$0 \leq d_i\alpha_i \leq \tfrac{C}{N}\ \forall i \in \mathbb{Z}_N \quad (36)$$
$$d_i g_i \geq +1\ \forall i \in \mathbb{Z}_N : \alpha_i = 0 \quad (37)$$
$$d_i g_i = +1\ \forall i \in \mathbb{Z}_N : 0 < |\alpha_i| < \tfrac{C}{N} \quad (38)$$
$$d_i g_i \leq +1\ \forall i \in \mathbb{Z}_N : |\alpha_i| = \tfrac{C}{N} \quad (39)$$
$$h = 0 \quad (40)$$

The One-Class C-SVC

This is equivalent to the unified case with $\chi = 1$, $\varepsilon = 1$, $\zeta = 0$ and $z_i = 0, d_i = +1\ \forall i \in \mathbb{Z}_N$. Noting that $\alpha_i \geq 0\ \forall i \in \mathbb{Z}_N$ (as $l_i = 0\ \forall i \in \mathbb{Z}_N : d_i = +1$), the dual may be seen to be:

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha},b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - b$$
$$\text{such that: } \mathbf{0} \leq \boldsymbol{\alpha} \leq \tfrac{C}{N}\mathbf{1} \quad (41)$$

which has the optimality conditions:

$$0 \leq \alpha_i \leq \tfrac{C}{N}\ \forall i \in \mathbb{Z}_N \quad (42)$$
$$g_i \geq +1\ \forall i \in \mathbb{Z}_N : \alpha_i = 0 \quad (43)$$
$$g_i = +1\ \forall i \in \mathbb{Z}_N : 0 < |\alpha_i| < \tfrac{C}{N} \quad (44)$$
$$g_i \leq +1\ \forall i \in \mathbb{Z}_N : |\alpha_i| = \tfrac{C}{N} \quad (45)$$
$$h = 0 \quad (46)$$

The ε-SVR

This is equivalent to the unified case with $\chi = -1$, $\zeta = 0$ and $d_i = 0\ \forall i \in \mathbb{Z}_N$. In this case, the dual is:

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha}, b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - \boldsymbol{\alpha}^T\mathbf{z} + \varepsilon\sum_{i\in\mathbb{Z}_N}|\alpha_i|$$

such that: $-\frac{C}{N}\mathbf{1} \le \boldsymbol{\alpha} \le \frac{C}{N}\mathbf{1}$

(47)

which has the optimality conditions:

$$-\tfrac{C}{N} \le \alpha_i \le \tfrac{C}{N} \;\; \forall i \in \mathbb{Z}_N \quad (48)$$

$$g_i \ge z_i + \varepsilon \;\; \forall i \in \mathbb{Z}_N : \alpha_i = -\tfrac{C}{N} \quad (49)$$

$$g_i = z_i + \varepsilon \;\; \forall i \in \mathbb{Z}_N : -\tfrac{C}{N} < \alpha_i < 0 \quad (50)$$

$$g_i \in \left[z_i - \varepsilon, z_i + \varepsilon\right] \;\; \forall i \in \mathbb{Z}_N : \alpha_i = 0 \quad (51)$$

$$g_i = z_i - \varepsilon \;\; \forall i \in \mathbb{Z}_N : 0 < \alpha_i < \tfrac{C}{N} \quad (52)$$

$$g_i \le z_i - \varepsilon \;\; \forall i \in \mathbb{Z}_N : \alpha_i = \tfrac{C}{N} \quad (53)$$

$$h = 0 \quad (54)$$

A MECHANICAL ANALOGY

For the C-SVC, there is a very useful mechanical analogy (Burges, 1998), which may be used to give an intuitive interpretation of the dual variables $\boldsymbol{\alpha}$. Recall that the trained machine $g: X \to \mathbb{R}$ defines a decision surface $S(\boldsymbol{\alpha}, b)$ in feature space given by:

$$S(\boldsymbol{\alpha}, b) = \left\{\boldsymbol{\phi} \,\middle|\, \sum_{i\in\mathbb{Z}_N} \alpha_i \boldsymbol{\varphi}^T(x_i)\boldsymbol{\phi} + b = 0\right\}$$

which is a hyperplane in feature space.

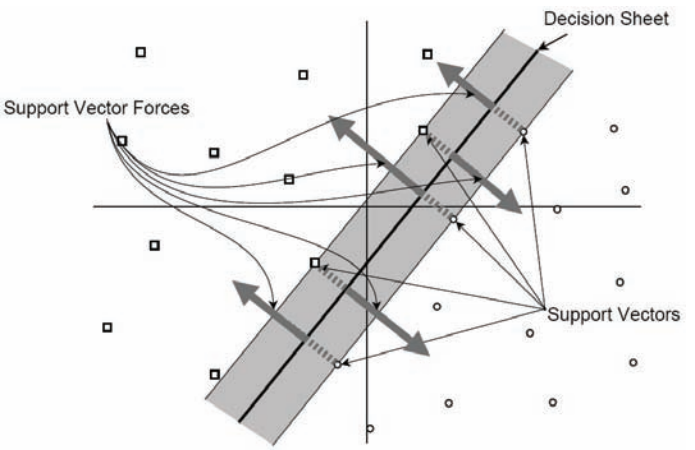
Let us suppose that we have a rigid, flat sheet lying on this hyperplane, which we will call the *decision sheet*. Let us further suppose that the image of each support vector x_i exerts a force $F_i = \alpha_i d_i \hat{\mathbf{w}}$ perpendicularly onto the decision sheet (i.e., by means of a rod set perpendicular to the decision sheet between the decision sheet and the support vector image $\boldsymbol{\varphi}(x_i)$), as shown in Figure 8. The conditions for mechanical equilibrium for this system are then (Burges, 1998):

$$\sum \text{Forces} = \sum_{i\in\mathbb{Z}_N} \alpha_i d_i \hat{\mathbf{w}} = 0$$

$$\sum \text{Torques} = \sum_{i\in\mathbb{Z}_N} \Gamma_{i;\mu_0\mu_1\ldots\mu_{d_H-3}} = 0$$

The first of these is automatically satisfied by any C-SVC decision surface by virtue of the optimality condition $h = \mathbf{1}^T\boldsymbol{\alpha} = 0$. For the second, note that the torque exerted on the decision sheet by any training vector (support or otherwise) is displayed in Box 5, where $\varepsilon_{\mu_0\mu_1\ldots\mu_{d_H-1}}$ is the com-

Figure 8. Mechanical analogy for the C-SVC



pletely antisymmetric tensor.[5] Combined with (20), what is displayed in Box 6 may be seen.

From these results, we see that the decision sheet lies at a point of mechanical equilibrium. So in the separable case, all support vectors exert some force on the decision sheet or surface, with the "most important" support vectors exerting the most force. In the nonseparable case, the same is again true, except that in this case, we place a limit on the magnitude of the force, which any one support vector may exert, effectively limiting the amount of influence any given training vector may have on the position of the decision surface. In this analogy, the support vectors actually physically "support" the decision sheet at a position of mechanical equilibrium, which justifies the name.

It is not difficult to extend this analogy to the one-class C-SVC. Indeed, the only difference is that all support vectors are exerting force in the same direction, and we add a uniform (i.e., distributed evenly across the decision sheet) opposing force ζ on the decision sheet. So:

$$\sum \text{Forces} = \sum_{i\in\mathbb{Z}_N} \alpha_i d_i \hat{\mathbf{w}} - \zeta\hat{\mathbf{w}} = 0$$

$$\sum \text{Torques} = \sum_{i\in\mathbb{Z}_N} \Gamma_{i;\mu_0\mu_1\ldots\mu_{d_H-3}} = 0$$

as shown in Figure 9. Once again, the optimality conditions ensure that both conditions are met.

We could also trivially extend this analogy directly to the ε-SVR case. However, in this case, the decision surface is less meaningful than in the C-SVC case, and hence, the analogy is less useful.

GENERAL COST FUNCTIONS IN THE UNIFIED SVM

It is good to consider for a moment the issue of cost functions and their selection. It may be asked precisely why we have chosen the cost functions that we have for the problems of classification and regression. In particular, the cost function used for C-SVC:

$c(\delta,d)= \max(0,1-d\delta)$

appears questionable. If we are only interested in the sign of the trained machine $g(x)$, then why are we penalizing errors based on both the sign *and* the magnitude of $g(x)$? Surely, a more sensible cost function for binary classification would be:

$$c(\delta,d) = \begin{cases} 1 & \text{if } \operatorname{sgn}(d) = \operatorname{sgn}(\delta) \\ 0 & \text{otherwise} \end{cases}$$

We may also ask why we have chosen the cost function:

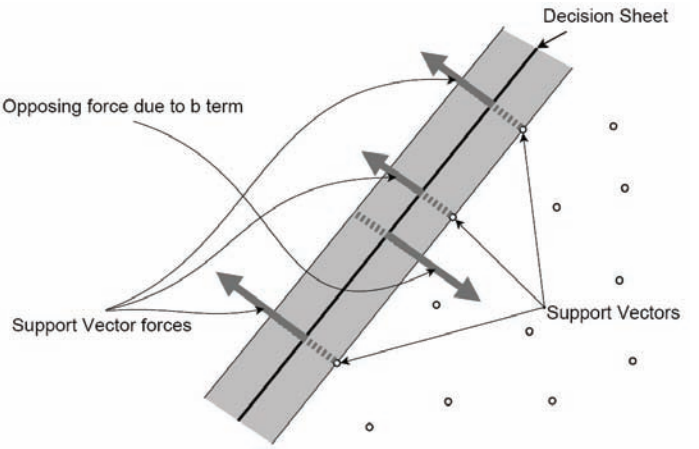
Box 5.

$$\begin{aligned}\Gamma_{i;\mu_0\mu_1\ldots\mu_{d_H-3}} &= \sum_{\mu_{d_H-2},\mu_{d_H-1}\in\mathbb{Z}_{d_H}} \varepsilon_{\mu_0\mu_1\ldots\mu_{d_H-1}} \varphi_{\mu_{d_H-2}}(x_i) F_{i\mu_{d_H-1}} \\ &= \sum_{\mu_{d_H-2},\mu_{d_H-1}\in\mathbb{Z}_{d_H}} \varepsilon_{\mu_0\mu_1\ldots\mu_{d_H-1}} \varphi_{\mu_{d_H-2}}(x_i) \alpha_i d_i \hat{w}_{d_H-1}\end{aligned}$$

Box 6.

$$\sum_{i\in\mathbb{Z}_N} \Gamma_{i;\mu_0\mu_1\ldots\mu_{d_H-3}} = \sum_{\mu_{d_H-2},\mu_{d_H-1}\in\mathbb{Z}_{d_H}} \|\mathbf{w}\| \varepsilon_{\mu_0\mu_1\ldots\mu_{d_H-1}} \hat{w}_{\mu_{d_H-2}} \hat{w}_{\mu_{d_H-1}} = 0$$

Figure 9. Mechanical analogy for the one-class C-SVC



$$c(\delta, d) = \max\left(0, |\delta - z| - \varepsilon\right)$$

for the ε-SVR. After all, there is no guarantee that an alternative cost function might not give better results.

In either case, the answer is a mixture of pragmatism and historical accident. Essentially, these cost functions both result in particularly simple dual optimization problems that have been shown to perform quite well in many practical applications. Nevertheless, some work has been done looking at alternative cost functions (Smola & Scholkopf, 1998; Smola, Scholkopf & Muller, 1998a, 1998b) for SVMs, which we now discuss.

In general, let $t,t^*:\mathbb{R}\rightarrow\mathbb{R}$ be differentiable, nondecreasing penalty functions such that $t(0) = t^*(0) = 0$ (we will see shortly that these functions define a general cost function). Based on these penalty functions, the generalized unified SVM primal training problem is defined to be (Shilton, 2006; Shilton & Palaniswami, 2004) (see Box 7), where once again $C \in \mathbb{R}^+$, $\varepsilon \in \mathbb{R}^+ \cup \{0\}$, $\zeta \in \mathbb{R}$ and $\chi \in \{-1,+1\}$ are constants; and we require that $\chi = -1$ if $d_i = 0$ for any $i = \mathbb{Z}_N$. Examination of (55) will show that the cost function associated with (55) in the case of classification ($\chi = +1, \varepsilon = 1$ and $z_i = 0, d_i \in \{-1,+1\}\ \forall i \in \mathbb{Z}_N$) is:

Box 7.

$$\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} R_1\left(\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*\right) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{N}\left(\sum_{i\in\mathbb{Z}_N : d_i \geq 0} t\left(\xi_i\right) + \sum_{i\in\mathbb{Z}_N : d_i \leq 0} t\left(\xi_i^*\right)\right) + \zeta b$$

such that:
$$\mathbf{w}^T\boldsymbol{\varphi}\left(x\right) + b - z_i \geq +\chi\varepsilon - \xi_i \ ,\forall i \in \mathbb{Z}_N \ \ d_i \geq 0$$
$$\mathbf{w}^T\boldsymbol{\varphi}\left(x\right) + b - z_i \leq -\chi\varepsilon + \xi_i^* \ ,\forall i \in \mathbb{Z}_N \ \ d_i \leq 0$$
$$\boldsymbol{\xi} \geq \mathbf{0} \ ,\forall i \in \mathbb{Z}_N \ \ d_i \geq 0$$
$$\boldsymbol{\xi}^* \geq \mathbf{0} \ ,\forall i \in \mathbb{Z}_N \ \ d_i \leq 0 \qquad (55)$$

$$c(\delta, d) = \begin{cases} t(1 - d\delta) & \text{if } d > 0 \text{ and } d\delta < +1 \\ t^*(1 - d\delta) & \text{if } d < 0 \text{ and } d\delta < +1 \\ 0 & \text{otherwise} \end{cases}$$

and similarly, the cost function associated with (55) in the case of regression ($\chi = -1$, $\zeta = 0$ and $d_i = 0 \ \forall \mathrm{i} \in \mathbb{Z}_{\mathrm{N}}$) is:

$$c(\delta, z) = \begin{cases} t(|\delta - z|_\varepsilon) & \text{if } \delta < z - \varepsilon \\ t^*(|\delta - z|_\varepsilon) & \text{if } \delta > z + \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

We will not give a detailed derivation of the dual form of this primal training problem for reasons of brevity (the derivation is not overly different to what was given previously, and moreover, may be found in (Shilton, 2006; Shilton & Palaniswami, 2004; Smola & Scholkopf, 1998; Smola et al., 1998a, 1998b). Suffice it to say that the dual generalized unified SVM training problem is shown in Box 8, where we have defined:

$$T(\xi) = t(\xi) - \xi \frac{\partial t}{\partial \xi}(\xi)$$

$$T^*(\xi^*) = t^*(\xi^*) - \xi^* \frac{\partial t^*}{\partial \xi^*}(\xi^*)$$

One example of an alternative penalty function is the quadratic function $t(\xi) = t^*(\xi) = \xi^2$. Using this penalty function, the generalized unified SVM training problem (56) becomes what is shown in Box 9, which should provide a better measure of risk (in the maximum likelihood sense) in the case of regression training data affected by Gaussian noise. It also has the advantage that the matrix $\mathbf{K} + \frac{N}{C}\mathbf{I}$ is positive definite, whereas $\mathbf{K}$ on its own is only guaranteed to be positive semidefinite. Note that in the regression case, if we set $\varepsilon = 0$,

Box 8.

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha}, b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{z}\\ \zeta\end{bmatrix} - \chi\varepsilon \sum_{i \in \mathbb{Z}_N} |\alpha_i| - \frac{C}{N}\sum_{i \in \mathbb{Z}_N} T(\xi_i) - \frac{C}{N}\sum_{i \in \mathbb{Z}_N} T^*(\xi_i^*)$$

such that:

$$\alpha_i \geq 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0$$

$$\alpha_i \leq 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0$$

$$\xi_i = \begin{cases} \inf\left\{\xi \middle| \frac{C}{N}\frac{\partial t}{\partial \xi}(\xi) > |\alpha_i|\right\} & \text{if } \alpha_i \geq 0 \\ 0 & \text{if } \alpha_i < 0 \end{cases}$$

$$\xi_i^* = \begin{cases} 0 & \text{if } \alpha_i > 0 \\ \inf\left\{\xi^* \middle| \frac{C}{N}\frac{\partial t^*}{\partial \xi^*}(\xi^*) > |\alpha_i|\right\} & \text{if } \alpha_i \leq 0 \end{cases}$$

$$\boldsymbol{\xi}, \boldsymbol{\xi} \geq \mathbf{0} \qquad (56)$$

Box 9.

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha}, b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K} + \frac{N}{C}\mathbf{I} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{z}\\ \zeta\end{bmatrix} - \chi\varepsilon \sum_{i \in \mathbb{Z}_N} |\alpha_i|$$

such that:

$$\alpha_i \geq 0 \ \forall i \in \mathbb{Z}_N : d_i \geq 0$$

$$\alpha_i \leq 0 \ \forall i \in \mathbb{Z}_N : d_i \leq 0$$

this simplifies to Suykens' least-squares support vector machine (LS-SVM) form (Suykens, Van Gestel, De Brabanter, De Moor & Vandewalle, 2002); namely:

$$\min_{\boldsymbol{\alpha}} \max_{b} Q(\boldsymbol{\alpha}, b) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{K}+\frac{N}{C}\mathbf{I} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} - \begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix}^T \begin{bmatrix}\mathbf{z}\\ 0\end{bmatrix}$$

and so:

$$\begin{bmatrix}\boldsymbol{\alpha}\\ b\end{bmatrix} = \begin{bmatrix}\mathbf{K}+\frac{N}{C}\mathbf{I} & \mathbf{1}\\ \mathbf{1}^T & 0\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{z}\\ 0\end{bmatrix}$$

More generally, though, the generalized dual form (55) allows us to experiment with very general cost functions. Such functions need not be smooth or symmetrical, so long as they satisfy the very minor requirements stated previously. However, there is a price: rather than having to deal with a simple, readily solvable dual training problem, we instead may be faced with a complicated dual optimization problem (a major exception to this being the quadratic function described previously). The dual can be solved in general (Smola & Scholkopf, 1998; Smola et al., 1998a, 1998b), but even so, the generic training process will usually take longer than the standard training process, which could be a serious impediment when dealing with larger training sets.

A PRACTICAL EXAMPLE: SPAM DETECTION

To get a better feel for the steps involved in training an SVM, we consider the problem of spam e-mail detection. In this case, we are given objects $x \in X$, where X is the set of all possible e-mails, and asked to classify these as either spam e-mail (class +1) or nonspam e-mail (class –1).

For the purposes of this experiment, we used the spam-base dataset from the UCI repository (Blake & Merz, 1998). The spam-base dataset consists of 4,601 training pairs, 1,813 of which are labeled as spam (class –1), the remainder being nonspam (class +1). Input space is $X = \mathbb{R}^{57}$.

We will consider two approaches here. Our first, more conventional approach uses a C-SVC trained on the complete spam-base dataset. We test a range of kernels and C values, using fivefold cross-validation[6] to find the optimal parameters for this training set. Our second, less conventional approach also uses the spam-base dataset, but in this case, we attempt to solve the problem by training a one-class C-SVC using only nonspam e-mails to train our SVM.

Let us first consider the results achieved for the C-SVC trained with the full spam-base dataset. In any SVM training process, two steps must be completed:

1. **Data normalization.** Vectors in the training set Θ must be normalized. This ensures that the matrix **K** is well scaled and prevents preferential bias toward particular attributes. Normalization is done on an attribute-by-attribute basis by shifting by $\mathbf{m} \in \mathbb{R}^{d_L}$ and scaling by $\mathbf{s} \in \mathbb{R}^{d_L}$. Typically, **m** and **s** are selected to ensure that each attribute either lies in some predetermined range (usually [–1,+1]) or that the attribute has zero mean and unit variance over the entire training set. We used the latter in this experiment. Mathematically, the normalization operation is:

$$\mathbf{x}_i := \left(\operatorname{diag}(\mathbf{s})\right)^{-1}(\mathbf{x}_i - \mathbf{m}) \;\forall i \in \mathbb{Z}_N$$

2. **Parameter selection.** The kernel and tradeoff parameter C must be selected. The most common way of doing this is to select a grid of potential kernels and C values and then test each in turn. Testing is typically done using cross-validation, and the pair kernel C with the smallest cross-validation error is selected. Some human interaction is required during this process, both in selecting appro-

priate ranges for kernel C and adjusting these appropriately as results come to hand.

As is common practice, we have selected out kernel from a set of "common" kernels; namely:

- **Linear:** $K(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{y}$
- **Polynomial:** $K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}^T\mathbf{y})^p$, where $p \in \{2,3\}$
- **Gaussian RBF:** $K(\mathbf{x},\mathbf{y}) = e^{-\frac{1}{\gamma}\|\mathbf{x}-\mathbf{y}\|^2}$, where $\gamma \in \{1,3,10,30,100,300\}$

The process of selecting C must be done on a kernel-by-kernel basis, and the range from which it is selected may need to be adjusted as the experiment progresses. In this case, however, we chose to select C from the range $\frac{C}{N} \in \{0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000, 3000\}$. One may always use a finer grid for both kernel and C, or use a recursive approach to hone in on the true "best" value. However, such a process can be very time-consuming, and it is important to know when to desist from "fine tuning" and accept that results are "good enough."

The results achieved for this dataset are shown in Table 1, where for brevity we have included only best C value for each kernel. Best results were achieved using a Gaussian RBF kernel with γ = 100 and $\frac{C}{N} = 100$ with 93.92% of all e-mails correctly labeled on average during the fivefold cross-validation process.

For comparative purposes, we also ran the same tests using an alternative penalty function:

$$t(\zeta) = t^*(\zeta) = \tanh(\zeta)$$

which is designed to apply a uniform penalty to all wrongly classified error vectors, thereby ignoring the magnitude of these errors. Using the same set of tests, the best result achieved for this cost function was 94.22 of e-mail correctly classified for an RBF Gaussian kernel, again with γ = 100 and $\frac{C}{N} = 100$. It should be noted, however, that while we were able to achieve better results using this cost function, the overall training process in this case took marginally longer to complete. As this is a relatively small dataset, the slowdown experienced was not too serious, but for significantly larger datasets, this time factor must be taken into account.

For our one-class C-SVC experiment, we are attempting to test the trained machine's ability to differentiate spam essentially blind, based only on nonspam e-mail samples. This experiment is somewhat artificial, although one may seek

Table 1. Results summary for the spam-base dataset using various kernel functions; only results for the optimal C parameter are given in each case

Kernel Type	***C* / *N***	N_S	N_E	**Accuracy**
Linear Kernel	30	892	835	93.13%
Quadratic Kernel	0.1	799	255	92.87%
Cubic Kernel	0.3	786	40	92.43%
Gaussian RBF γ = 1	10	3919	31	80.78%
Gaussian RBF γ = 3	10	3326	49	85.80%
Gaussian RBF γ = 10	3	2179	248	91.85%
Gaussian RBF γ = 30	10	1301	323	93.50%
Gaussian RBF γ = 100	**10**	**984**	**601**	**93.92%**
Gaussian RBF γ = 300	10	975	829	93.89%

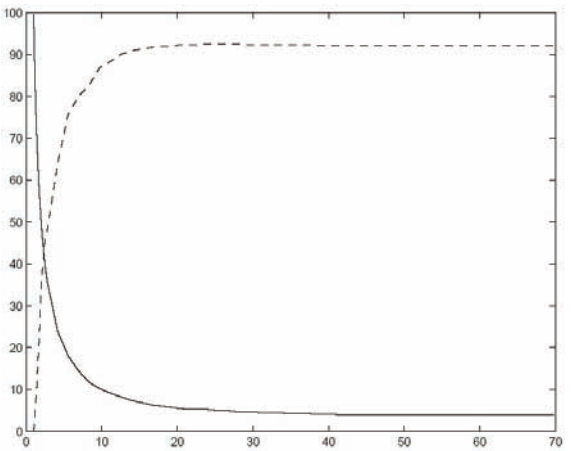
to justify it by arguing that the ever-changing nature of spam e-mail content and presentation will quickly render our "spam" training vectors redundant.

For simplicity, we test only one kernel; namely, the RBF kernel with $\gamma = 100$ with C varied over a much smaller yet more fine-grained region (chosen experimentally to give a good sample of the results that are achievable). In this experiment, we have two types of errors; namely, *false positives* (nonspam e-mails incorrectly labeled as spam) and *false negatives* (spam e-mails incorrectly labeled as nonspam), where in general, the former is considered to be more important than the latter. A graph of these results over a range of C values is shown in Figure 10.

CONCLUSION

In this chapter, we have introduced a unified SVM formulation that incorporates binary classification, one-class classification, and regressions. In so doing, we have endeavored to highlight the connections between the various formulations, which are often introduced entirely separately. In particular, we have discussed the equivalence between margin maximization in the classification case and function flattening in the regression case. Using a mechanical analogy, we have given an intuitive interpretation of both binary and one-class classification, and the connection between them. Finally, we have extended the SVM model to cover general, asymmetric cost functions, and given an example of an application of SVMs to spam e-mail detection to highlight the practical issues involved in their use.

Figure 10. One-class C-SVC results for the spambase dataset as a function of C. The percentage of correctly classified nonspam e-mails (obtained from cross-validation) is shown by the solid line, while the dashed line shows the percentage of correctly classified spam e-mails



REFERENCES

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases.* University of California, Department of Information and Computer Science. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining, 2*(2), 121–167.

Cochran, J.A. (1972). *The analysis of linear integral equations.* McGraw-Hill.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*(3). 273–297.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In M.C. Mozer, M.I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, vol. 9* (p. 155). MIT Press.

Fletcher, R. (1981). *Practical methods of optimisation, volume 2: Constrained optimisation.* Chichester: John Wiley and Sons.

Haykin, S. (1999). *Neural networks—A comprehensive foundation*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Herbrich, R., & Weston, J. (1999). *Adaptive margin support vector machines for classification learning.* Proceedings of the Ninth International Conference on Artificial Neural Networks, 880–885.

Kecman, V. (2001). *Learning and soft computing, support vector machines, neural networks and fuzzy logic models.* Cambridge, MA: MIT Press.

Lee, J., & Lee, D. (2005). An improved cluster labeling method for support vector clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(3), 461–464.

Manevitz, L.M., & Yousef, M. (2001). One-class {SVM}s for document classification. *Journal of Machine Learning Research, 2.* 139–154.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Transactions of the Royal Society of London, 209(A).*

Scholkopf, B., Bartlett, P., Smola, A., & Williamson, R. (1999). Shrinking the tube: A new support vector regression algorithm. *Advances in Neural Information Processing Systems, 11,* 330–336.

Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., & Williamson, R.C. (1999). *Estimating the support of a high-dimensional distribution* (Tech. Rep MSR-TR-99-87). Redmond: Microsoft Research.

Shilton, A. (2006). *Design and training of support vector machines* [doctoral dissertation]. Melbourne: The University of Melbourne.

Shilton, A., & Palaniswami, M. (2004). A modified nu-SV method for simplified regression. *Proceedings of the International Conference on Intelligent Sensing and Information Processing,* 422–427.

Shilton, A., Palaniswami, M., Ralph, D., & Tsoi, A.C. (2005). Incremental training of support vector machines. *IEEE Transactions on Neural Networks, 16*(1), 114–131.

Smola, A. (1996). *Regression estimation with support vector learning machines* [master's thesis]. Technische Universitat Munschen.

Smola, A., & Scholkopf, B. (1998). *A tutorial on support vector regression* (Tech. Rep. NeuroCOLT2 Technical Report Series, NC2-TR-1998-030). London: University of London, Royal Holloway College.

Smola, A., Scholkopf, B., & Muller, K. (1998). Convex cost functions for support vector regression. *Proceedings of the 8th ICANN, Perspectives in Neural Computing,* 99–104.

Smola, A., Scholkopf, B., & Muller, K. (1998). General cost functions for support vector regression. *Proceedings of the Ninth Australian Conference on Neural Networks,* 79–83.

Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines.* New Jersey: World Scientific Publishing.

Vapnik, V. (1995). *Statistical learning theory.* New York: Springer-Verlag.

Vapnik, V., Golowich, S., & Smola, A. (1997). Support vector methods for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems, 9,* 281–187.

Wang, L.P. (Ed.). (2005). *Support vector machines: Theory and application.* Berlin: Springer.

ENDNOTES

[1] It will be seen later that the feature map $\varphi : X \to \mathbb{R}^{d_H}$ is never actually defined or

used explicitly. Instead, a *kernel function K*: $X \times X \rightarrow \mathbb{R}$ satisfying Mercer's (1909) condition and hence *implicitly* defining such a map will be used. As a result, the feature map or kernel function used may be as complex or as simple as required to fit the problem without direct penalty in terms of training time or implementation difficulty.

[2] As an aside, it may be noted that each tuple $(x,0,z) \in \Theta$ may be removed and replaced by two tuples $(x,-1,z)$ and $(x,+1,z)$ without changing the constraint set of the primal (6) in any nontrivial manner. Hence, we do not strictly require the pseudo-class 0 to achieve regression in our unified SVM. From a practical standpoint, however, applying this reasoning to an ε-SVR problem would double the size of the training set, which would, as we will see shortly, double the dimensionality of the dual training problem. So, while theoretically possible, this approach is inadvisable.

[3] There may be local minima, but all local minima will be global.

[4] The integrals here are defined by the Lebesgue measure on *X*.

[5] Specifically (see Box 10).

[6] *n*-fold cross-validation involves splitting the training set into *n* equally sized subsets. The SVM is trained on $n - 1$ of these and tested on the remaining subset; results are recorded, and the process is repeated *n* times, leaving out a different subset each time. The results are then averaged to give the *n*-fold cross-validation error.

Box 10.

$$\varepsilon_{\mu_0\mu_1\cdots\mu_{d_H-1}} = \begin{cases} +1 & \text{if } (\mu_0,\mu_1,\ldots,\mu_{d_H-1}) \text{ is an even perm. of } (0,1,\ldots,d_H-1) \\ -1 & \text{if } (\mu_0,\mu_1,\ldots,\mu_{d_H-1}) \text{ is an odd perm. of } (0,1,\ldots,d_H-1) \\ 0 & \text{otherwise} \end{cases}$$

Chapter XV
Cluster Ensemble and Multi-Objective Clustering Methods

Katti Faceli
Federal University of São Carlos, Brazil

Andre C.P.L.F. de Carvalho
University of São Paulo, Brazil

Marcilio C.P. de Souto
Federal University of Rio Grande do Norte, Brazil

ABSTRACT

Clustering is an important tool for data exploration. Several clustering algorithms exist, and new algorithms are frequently proposed in the literature. These algorithms have been very successful in a large number of real-world problems. However, there is no clustering algorithm, optimizing only a single criterion, able to reveal all types of structures (homogeneous or heterogeneous) present in a dataset. In order to deal with this problem, several multi-objective clustering and cluster ensemble methods have been proposed in the literature, including our multi-objective clustering ensemble algorithm. In this chapter, we present an overview of these methods, which, to a great extent, are based on the combination of various aspects of traditional clustering algorithms.

INTRODUCTION

Clustering is an important tool for the exploration of datasets with no or very little prior information (Jain & Dubes, 1988). Several clustering algorithms have been proposed in the literature. They have been very successful to solve a large number of real problems in areas as diverse as biology, medicine, engineering, marketing, and remote sensing. A recent area in which cluster analysis has contributed with very important results is bioinformatics (Narayanan, 2005; Wang, Zaki, Toivonen & Shasha, 2003). In bioinformatics, researches related to functional genomics and gene

expression data analysis are employing clustering techniques to find clusters of genes (e.g., for the identification of gene functions) and clusters of tissue samples (e.g., for discovering groups and subgroups of cancer) (Costa, Carvalho & Souto, 2004; D'haeseleer, 2005; Eisen, Spellman, Brown & Botstein, 1998; Golub et al., 1999; Hautaniemi et al., 2003; Nikkilä, Törönen, Kaski, Venna, Castrén & Wong, 2002; Pal, Aguan, Sharma & Amari, 2007; Spellman et al., 1998; Tamayo et al., 1999).

In spite of this success, traditional cluster analysis algorithms present several limitations that need to be addressed. For instance, there is no precise definition of cluster. As a consequence, the literature describes a large number of clustering algorithms, each one looking for clusters according to a given clustering criterion, or cluster definition (Law, Topchy, & Jain, 2004).

Another problem is the fact that there is no clustering algorithm, optimizing a single criterion and able to reveal all types of structures (homogeneous or heterogeneous) present in a dataset. In fact, it is unfeasible to establish, a priori, which clustering criterion is more appropriated to capture the true structure. It is also important to mention that clustering algorithms can find structures at various refinement levels, depending on their parameter settings (Jain & Dubes, 1988).

Moreover, the same data can have more than one relevant structure, each one in accordance with a different cluster definition and/or with a different refinement level (e.g., number of clusters). For example, in functional genomics and gene expression data analysis, data usually have multiple meaningful interpretations; genes can fit into more than one functional category, or a disease such as cancer may have different subtypes depending on the required level of investigation (Alizadeh et al., 2000; Golub et al., 1999; Yeoh et al., 2002).

By contrast, the usual application of cluster analysis for the exploration of a dataset focuses on the discovery of only a single structure that best fits the data (Handl & Knowles, 2005; Jain & Dubes, 1988; Xu & Wunsch, 2005). In these cases, several clustering algorithms are applied to the dataset, extracting different structures. Next, a validation method selects the structure that best fits the data. However, the search for only one best-fit structure limits the amount of knowledge that might be obtained. Besides, most validation measures are biased toward a given clustering criterion, which reduces the robustness of the analysis performed.

As an attempt to overcome the limitations of traditional clustering algorithms, finding a higher-quality solution and improving the robustness against different data conformations, recent works have proposed the use of multi-objective clustering algorithms (Handl & Knowles, 2007; Korkmaz, Du, Alhajj, & Barker, 2006; Liu, Ram, & Lusch, 2005; Navernioukv, 2005; Ripon, Tsang, & Kwong, 2006) and cluster ensemble methods (Ayad & Kamel, 2003; Boulis & Ostendorf, 2004; Fern & Brodley, 2004; Fred & Jain, 2005; Law et al., 2004; Monti, Tamayo, Mesirov, & Golub, 2003; Strehl & Ghosh, 2002; Topchy, Jain, & Punch, 2004; Topchy, Jain, & Punch, 2005). In fact, any area that benefits from using cluster analysis can take advantage of using cluster ensemble or multi-objective clustering methods. Applications of the first started in domains like network anomaly detection (Munson & Caruana, 2006), document clustering (Greene, Tsymbal, Bolshakov, & Cunningham, 2006), image analysis (Lourenco & Fred, 2005), medical diagnostics (Greene, Tsymbal, Bolshakov, & Cunningham, 2006), and bioinformatics (Asur, Parthasarathy, & Ucar, 2006; Hu, 2006; Hu, Yoo, Zhang, Nanavati, & Das, 2005; Monti et al., 2003; Souto, Araujo, & Silva, 2006). Multi-objective clustering applications started in the following applications: marketing (Liu et al., 2005), computer networks (Cheng, Cao, Wang, & Das, 2006), and bioinformatics (Handl, 2006; Mitra & Banka, 2006).

Despite their advantages when compared with traditional clustering methods, each previous ap-

proach has its own limitations (Faceli, Carvalho, & Souto, 2006; Handl & Knowles, 2007). Cluster ensemble techniques, which aim to result in a single consensus structure, require fine parameters tuning and could have a poor performance if a large number of the partitions to be combined are of low quality. The multi-objective approaches produce a set of partitions, instead of a single partition, as the final solution. From this set, the users can choose the solutions that seem to be more appropriate for their particular application. Nevertheless, as the size of the solution set increases, this user-based selection becomes harder (Handl & Knowles, 2007). Handl and Knowles (2007) and Navernioukv (2005) have proposed the use of an automatic selection method, which provides an indication of the best partitions in the solution set. However, in our experimental works, we have observed that these indications do not always correspond to the best solutions present in the solution set.

In an attempt to minimize these problems, we have proposed a multi-objective clustering ensemble (MOCLE) method that combines aspects from cluster ensemble and multi-objective clustering algorithms (Faceli, 2006; Faceli et al., 2006). In our approach, we first generate a set of individual partitions by running several traditional clustering algorithms with different parameters settings. These partitions are then used as the initial population for a Pareto-based multi-objective genetic algorithm. The optimization process is used to combine and select the best set of alternative partitions.

The initial population and the optimization of different objective functions (validation measures biased toward distinct clustering criteria) compose the multi-objective aspect of MOCLE. A special crossover operator that finds the consensus between two partitions guarantees the ensemble aspect. These characteristics give MOCLE the ability to (1) robustly deal with different types of clusters that occur at different refinement levels, (2) find a concise set of high-quality alternative structures, and (3) reduce the need for user expertise on either cluster analysis or the data domain in order to improve the performance obtained by a clustering technique. Moreover, MOCLE allows the integration of previous domain knowledge by means of the objective functions.

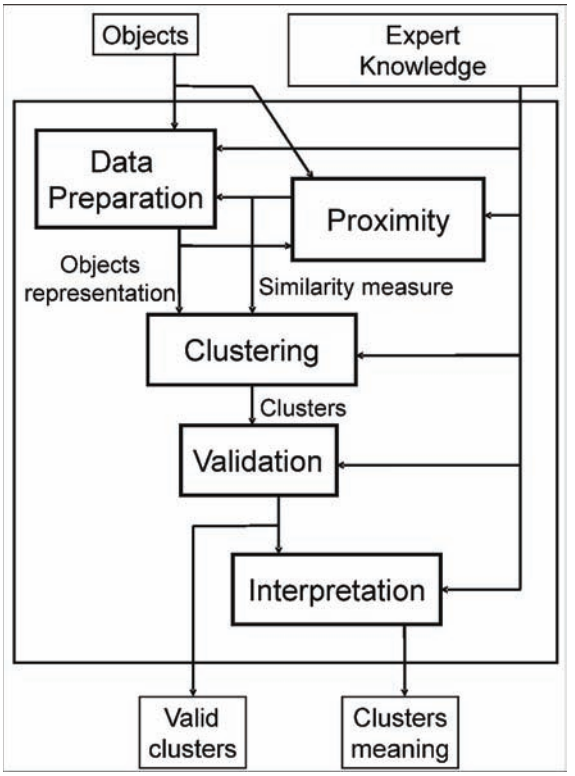
In this chapter, we describe the state of the art of the research in both cluster ensemble and multi-objective clustering algorithms. For such, first we will present the main problems faced by traditional clustering algorithms, which motivated the development of the cluster ensemble and multi-objective clustering approaches. Next, we will show the advantages provided by these methods to cluster analysis and their current limitations. To illustrate these approaches, we will briefly describe three cluster ensemble methods and one multi-objective clustering algorithm. We will also present the proposed multi-objective clustering ensemble (MOCLE) method and show how it relates to the other methods described. Then, to illustrate the techniques discussed throughout the chapter, we will present an application example from the area of bioinformatics. The limitations and future trends associated with the ensembles and multi-objective approaches will be discussed at the end of this chapter.

BACKGROUND

The main goal of a clustering technique is to find a structure of clusters in the dataset where the objects belonging to each cluster share some relevant properties regarding the data domain (Handl & Knowles, 2005; Jain & Dubes, 1988; Xu & Wunsch, 2005). Here we will focus on a particular type of structure: a hard partition (or partition, for short). In such a structure, each object should be assigned to only one cluster, and all objects must be assigned to one of the clusters.

Cluster analysis comprises several steps, as shown in Figure 1 (Barbara, 2000; Jain, Myrthy & Flynn, 1999). The data preparation includes

Figure 1. Clustering process



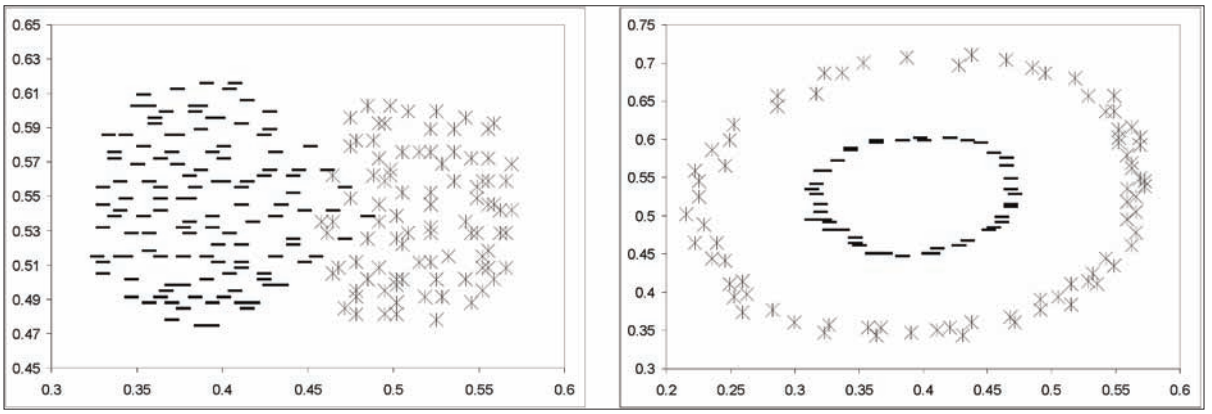
the preprocessing of the objects (normalization, type conversions, feature selection, or extraction) and their representation in a proper way for a particular clustering algorithm. Afterwards, an adequate proximity measure must be chosen to evaluate the similarity between objects. The main task of clustering consists of the application of a clustering algorithm to the dataset, given a proximity measure. Next, the results must be evaluated in an objective way to check if the clusters are valid (they did not occur by chance or as an artifact of the algorithm). An additional step of clusters interpretation can be included to establish their meaning.

One can see in Figure 1 that expert knowledge is important in all steps of the cluster analysis process. This includes knowledge of the clustering technique, the data gathering process, and on the domain (Jain et al., 1999). Even so, clustering can still be considered an important tool for the exploration of data when no or very little prior information is available, since the expert knowledge is applied only to guide some choices during the process (Handl & Knowles, 2005; Jain & Dubes, 1988; Xu & Wunsch, 2005). However, there are several difficult issues in cluster analysis. The first issue is that a precise definition of cluster does not exist (Xu & Wunsch, 2005). Some common definitions of cluster are (Barbara, 2000):

- **Well-separated cluster:** Each object in a cluster is closer to every other object in this cluster than to any object in another cluster.
- **Center-based cluster:** Each object in a cluster is closer to this cluster center than to the center of any other cluster.
- **Contiguous cluster:** Each object in a cluster is closer to one or more objects in this cluster than to any other object outside the cluster.
- **Density-based cluster:** A cluster is a region with a high density of objects separated from other high-density regions by regions of low density of objects.
- **Similarity-based cluster:** The objects in a cluster are similar, while objects in different clusters are dissimilar.

The mathematical formulation of the intuitive cluster definition is named clustering criterion or objective function. The clustering criterion consists of a way of selecting a structure (or model) to represent the clusters that best fit the dataset being analyzed (Estivill-Castro, 2002). As a consequence of such a diversity of cluster definitions, there is a large number of clustering algorithms, each one looking for clusters according to a different cluster definition (or clustering criterion) (Law et al., 2004). For example, algorithms that look for compact clusters (e.g., *k*-means) are biased toward spherically shaped clusters. On the other hand, algorithms optimizing a criterion based in

Figure 2. Datasets with different types of clusters



the concept of connectedness (e.g., single-link), which captures local densities, can detect arbitrarily shaped clusters but are not robust against overlapping or not well-separated clusters.

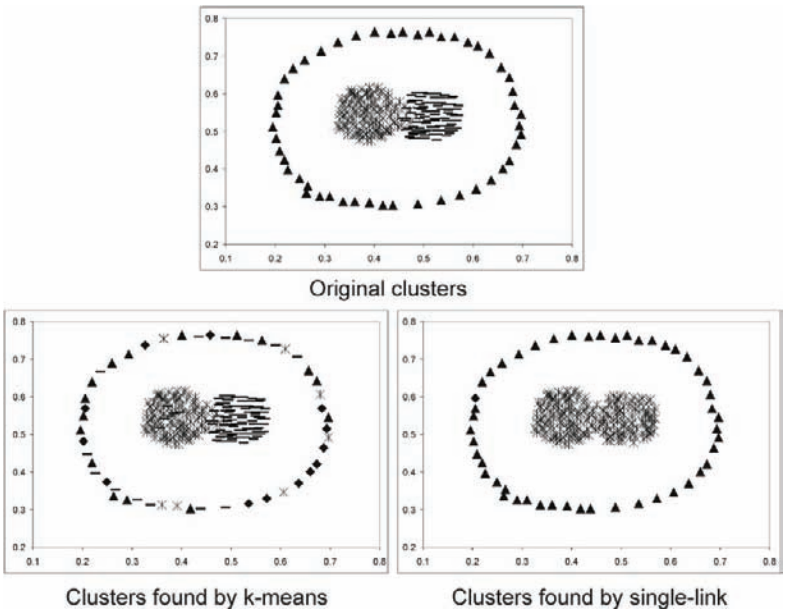
Consider the datasets shown in Figure 2. The dataset on the left presents two globular clusters not well separated. The dataset on the right contains two clusters with a ring shape. The *k*-means algorithm is able to find the globular clusters in the dataset on the left, but it fails for the dataset on the right with ring-shaped clusters. The single-link algorithm, in its turns, can find the ring clusters on the right, but it is inefficient for globular clusters. Therefore, the first difficulty in cluster analysis is to choose the most suitable algorithm to the dataset.

Another important aspect regarding clustering algorithms is the possibility of finding structures (partitions) at different refinement levels (different numbers of clusters or cluster densities), depending on their parameter settings (Jain & Dubes, 1988). This results in difficulties for the parameter adjustment. Clustering validation techniques can be used to guide the choice of the algorithm or its parameter adjustment (Handl & Knowles, 2005). However, most of these techniques are biased toward a clustering criterion. Therefore, several validation measures should be applied to select the most consistent results among those obtained with a variety of clustering algorithms using different parameter configurations (Handl & Knowles, 2005). That is, this whole process requires a deeper knowledge in cluster analysis than the experts in the data domain usually have.

Furthermore, each algorithm looks for a homogeneous structure (all clusters are in accordance with the same cluster definition), while data can present a heterogeneous structure (each cluster conforms to a different clustering criterion) (Law et al., 2004). For example, the dataset in Figure 3 contains three clusters: one ring-shaped cluster and two globular clusters. One can see that for this dataset, *k*-means distinguished the globular clusters but missed the ring-shaped one. On the other hand, single-link distinguished only the ring-shaped cluster. In fact, there is no single clustering algorithm able to find all kinds of clusters that can be present in a particular dataset (Estivill-Castro, 2002; Kleinberg 2002).

An additional issue that makes the cluster analysis difficult is that the same dataset can have more than one relevant structure, each one representing a different interpretation of the dataset (Handl & Knowles, 2007). These structures can be consistent with a different cluster definition and/or with a different refinement level. For example, consider the dataset shown in Figure 4. This dataset contains three structures: E1, E2,

Figure 3. Dataset with a heterogeneous structure

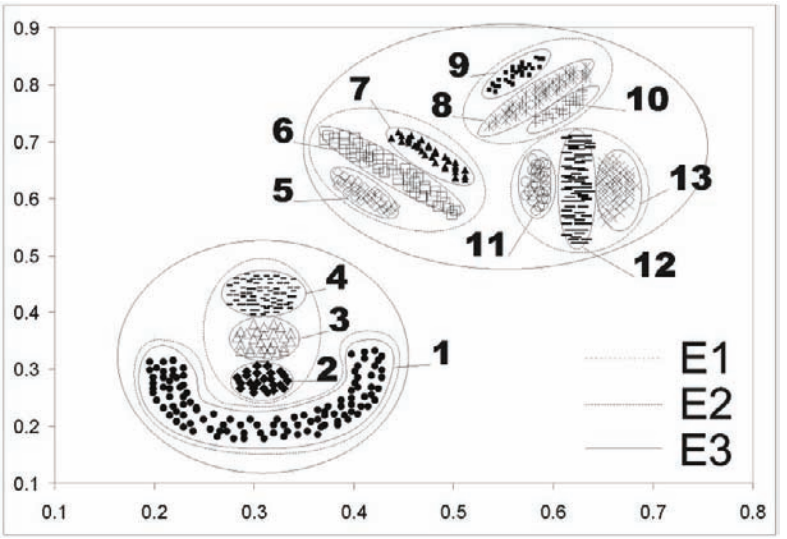


and E3. The simplest structure is E1, with two spherically shaped and well-separated clusters. In principle, any clustering algorithm and validation index would be able to identify it. The structure E2 is a refinement of E1, containing five clusters. E3 is also a refinement of E2, with 13 clusters. E2 and E3 are highly heterogeneous regarding the shape of the clusters. The traditional application of cluster analysis to explore a dataset focuses on the discovery of only a single structure that best fits the data. This limits the amount of knowledge that could be obtained. For this particular example, the structure E1 would be identified, and the structures E2 and E3 would be missed.

As previously mentioned, there exist several approaches in the literature trying to overcome some of the difficulties found in the traditional cluster analysis. These approaches are mainly based on cluster ensemble and the multi-objective clustering methods. There are several techniques related to each approach. In this chapter, we will describe three cluster ensembles techniques (Fern & Brodley, 2004; Law et al., 2004; Strehl & Ghosh, 2002) and one on multi-objective clustering algorithm (Handl & Knowles, 2007). These approaches are better suited to deal with different data conformations than the traditional clustering algorithms. However, as will be discussed later, each of them presents its own limitations.

A cluster ensemble is a way to obtain a consensus partition of high quality given a diverse set of individual (base) partitions. It comprises two steps: the generation of the base partitions and the combination of these partitions in order to produce a consensus partition. This last step is accomplished via a consensus function (Kuncheva, Hadjitodorov & Todorova, 2006). A few alternatives have been proposed for both tasks (Kuncheva et al., 2006; Topchy et al., 2004). Regarding the type of base partitions used as input for the consensus function, an ensemble can be homogeneous or heterogeneous. In a homogeneous ensemble, all base partitions are generated with the same clustering algorithm. In contrast, in a heterogeneous ensemble, the base partitions are generated with different clustering algorithms. Here, we will focus on the heterogeneous ensembles. Three

Figure 4. Dataset with several structures



good examples of such ensembles can be seen in Fern and Brodley (2004), Law, et al. (2004), and Strehl and Ghosh (2002**)**.

The work presented in Strehl and Ghosh (2002) is one of the most popular approaches for cluster ensemble. In this work, the authors formalize the cluster ensemble problem as a combinatorial optimization problem in terms of shared mutual information. In order to tackle the combinatorial complexity of the problem, they propose three algorithms (consensus functions): CSPA (cluster-based similarity partitioning algorithm), HGPA (hyper-graph partitioning algorithm), and MCLA (meta-clustering algorithm). A supra-consensus function based on the shared mutual information can be applied to select the best partition among the results produced by the three algorithms.

The CSPA algorithm starts with the construction of a new similarity matrix, according to the base partitions. The entries of this matrix denote the fraction of partitions in which two objects are assigned to the same cluster. The matrix is then employed to cluster the objects with any similarity-based clustering algorithm, producing the consensus partition. In the HGPA algorithm, the combination is treated as a problem of partitioning a hypergraph. In this hypergraph, the clusters of the base partitions are represented as hyperedges. The hypergraph is partitioned by cutting a minimal number of hyperedges. The MCLA algorithm considers the combination as a problem of finding the correspondence between the clusters of the base partitions. First, a metagraph is constructed where each cluster of the base partitions is a vertex. The edge weights are proportional to the similarity between the vertices. There are no edges connecting vertices from the same partition. Next, the metagraph is partitioned. The clusters assigned to the same group (metacluster) are considered correspondents. The objects are then assigned to the metaclusters to which they are more strongly associated, generating the consensus partition.

Another cluster ensemble method, based on graph partitioning, is proposed in Fern and Brodley (2004)—hybrid bipartite graph formulation (HBGF). In this method, first a bipartite graph is constructed using the set of base partitions, modeling their objects and clusters simultaneously as vertices. Next, the graph is partitioned by a traditional graph partitioning technique. The resulting division of the objects is the consensus partition.

In all the previously reviewed cluster ensemble methods, the goal is to find a consensus partition that agrees (resembles) as much as possible with all base partitions used as inputs. This goal can lead to two main problems. First, in this context, a large number of base partitions of poor quality can also result in a poor consensus partition, even if among the base partitions there are few of high quality. The second problem is related to the difficulty in the generation of a heterogeneous structure; that is, a partition with different types of clusters. For instance, even if each of the base partitions contains one cluster of excellent quality, according to one of the cluster definitions, this cluster will not appear in the final consensus partition because its information will be "overwritten" by the poor-quality clusters from other partitions. Thus, although considering multiple criteria in the individual partitions, the consensus function does not consider different clustering criteria for different regions of the feature space (Law et al., 2004).

Differently from the works of Fern and Brodley (2004) and Strehl and Ghosh (2002), the authors in Law, et al. (2004) proposed a cluster ensemble method that is not based on graph partitioning procedures. In fact, they call their approach a multi-objective data clustering algorithm. Even so, we decided to classify it as a cluster ensemble technique. This option was followed because this method differs significantly from the simultaneous optimization of multiple clustering criteria, which usually characterize multiobjective clustering. This method works as follows. Several clustering algorithms are employed to find different base partitions. Next, by using a given quality measure (stability of the clusters under the resampling of the dataset), some clusters of these base partitions are chosen to compose the consensus partition. By doing so, the algorithm chooses the best objective function for different parts of the feature space. One deficiency of this algorithm is its poor performance when the clusters formed according to different clustering criteria present a significant overlap.

One of the main disadvantages found in cluster ensemble methods is that, like the traditional clustering methods, they produce as a final result only a single partition. As previously discussed, this limits the amount of information that can be extracted from the data. Also, as in most of the traditional clustering algorithms, these methods rely on the fine adjustments of parameters to obtain a high-quality consensus partition. In several cases, the user must supply the number of clusters in advance. However, for real datasets, this number is usually not known a priori. Because of this complex parameter setting, the validation step for these methods requires a good deal of expertise, similar to what happens with traditional clustering algorithms.

The multi-objective clustering method described in Handl and Knowles (2007) overcomes several of the previous limitations. For example, different from the traditional clustering algorithms and the cluster ensemble methods, this approach can find solutions corresponding to different trade-offs between the clustering criteria being optimized. More precisely, Handl and Knowles (2007) describe a multi-objective evolutionary algorithm, MOCK (multi-objective clustering with automatic K-determination), able to simultaneously optimize two complementary clustering criteria: overall deviation and connectivity. MOCK returns a large number of different trade-off partitions over a range of different cluster numbers (solution set). However, as the number of alternatives increases, the analysis becomes harder (Handl & Knowles, 2007). To avoid this difficulty, MOCK includes a mechanism to automatically select the best partitions from the solution set. The selection is based on the shape of the Pareto front. This selection strategy relies on domain-specific considerations, which limits its application. Furthermore, we have found in our experimental work that the best partitions selected by MOCK did not always correspond

to the best partitions found by MOCK (Faceli, 2006; Faceli et al., 2006).

MULTI-OBJECTIVE CLUSTERING ENSEMBLE

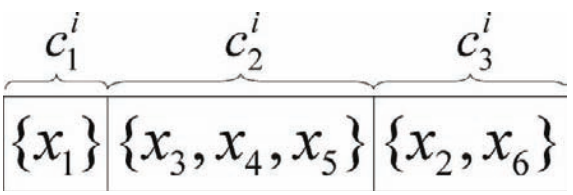
In an attempt to overcome the difficulties found in traditional cluster analysis algorithms, our multi-objective clustering ensemble algorithm (MOCLE) combines characteristics from both the cluster ensemble and multi-objective clustering methods (Faceli et al., 2006). As any cluster ensemble, MOCLE is composed of two main steps: (1) generation of a diverse set of base partitions and (2) determination of the consensus partition. Our approach differs from cluster ensemble methods in two ways. First, we look for a set of "consensus" partitions instead of only one. In fact, our set of solutions may contain partitions that are combinations of other partitions or high-quality partitions from the set of individual partitions. Second, we combine pairs of partitions iteratively in an optimization process instead of the usual combination of all partitions at the same time. This iterative combination/selection of partitions avoids the negative influence of low-quality base partitions, which can decrease the quality of the ensemble results.

More precisely, MOCLE works as follows. Initially, a set of base partitions is generated. Conceptually different clustering algorithms, optimizing different clustering criteria, are employed for this purpose. For example, algorithms looking for compact clusters may be used together with algorithms looking for connected clusters. The more diverse the algorithms are, the larger the number of types of cluster that can be discovered. Several parameter settings for the algorithms are also considered in the construction of the set of base partitions. These different settings generate partitions with clusters at different refinement levels (e.g., partitions with different numbers of clusters or partitions with clusters of several densities). It is important to have partitions with different types of clusters at several refinement levels so MOCLE can receive as much information as possible. As a result, the algorithm will be able to generate a large number of the structures present in the dataset. In fact, we assume that the relevant structures will be among the base partitions.

After generating the base partitions, the set of "consensus" partitions are found by the optimization of different objective functions using a Pareto-based multi-objective genetic algorithm. Any known algorithm can be employed. We have already investigated two of them: SPEA (strength Pareto evolutionary algorithm) (Zitzler & Thiele, 1999) and NSGA-II (non-dominated sorting genetic algorithm) (Deb, Pratap, Agarwal, & Meyrivan, 2002). Similar results were obtained with both algorithms (Faceli, 2006; Faceli et al., 2006). The use of this class of genetic algorithm results in a set of partitions, as previously mentioned, instead of a single partition produced by traditional and cluster ensemble methods. This is an important feature in domains like bioinformatics, where the data may have several interpretations.

The base partitions constitute the initial population to be used with the genetic algorithm. Each partition is an individual and is represented by an array of sets. Each set, in its turn, represents a cluster and contains the labels of its objects. Figure 5 illustrates the representation of an individual. In addition to the special initial population, two other adaptations are made in the traditional genetic algorithm: a special crossover operator and the use of diverse clustering validation measures

Figure 5. Individual



as objective functions. Together with the initial population, our special crossover operator is responsible for the ensemble aspect of MOCLE. This operator finds the consensus between two parent partitions. Any existing cluster ensemble method that can be applied to a pair of partitions can be used as crossover operator. We have investigated two methods for this step: HBGF and MCLA. Thus, our consensus crossover works as follows. Two parents are selected by binary tournament. Next, the cluster ensemble method is applied to combine the parent partitions. The number of clusters of the resulting consensus partition is randomly chosen in the interval of variation of the numbers of clusters of the parents.

With this operator, the partitions are combined in pairs, iteratively, during the evolutionary process. The consensus partitions generated at each iteration are also considered in the next combinations. This iterative combination avoids the negative influence of the low-quality partitions present in most of the traditional cluster ensemble methods. The low-quality partitions are gradually eliminated, while the best individual partitions and the good combinations are maintained for further combination.

Since we want to restrict the search space to the base partitions and their combinations, we do not apply a mutation operator. Therefore, the genetic algorithm tries to select the best partitions, not exploring the whole space of possible partitions. In the pure Pareto-based multi-objective clustering scenario, differences in the assignment of only one object to a different cluster in two partitions can result in a different trade-off of the measures optimized. This may result in a high number of very similar partitions in the approximation of the Pareto front obtained. Previous works on multi-objective clustering suggest the analysis of the Pareto front in order to select the best solutions (Handl & Knowles, 2007, Navernioukk, 2005). This is the approach followed by MOCK.

In contrast, we argue that in the context of clustering, the aim should not be the generation of the most complete Pareto front approximation possible. Indeed, having solutions representing each region of the Pareto front is enough to provide a relevant set of alternative partitions.

Considering this fact, MOCLE aims at the generation of a concise set of solutions that are representative of the Pareto front. As already mentioned, MOCLE relies on the ability of the clustering algorithms in finding high-quality partitions according to the employed criteria. Starting with a set of potentially good partitions, MOCLE uses the multiple objectives to select the best compromises. New partitions are created only by means of the crossover operator and represent the consensus among other existing partitions. As our crossover operator only produces combinations of existing partitions and no mutation is used, the search space will not be explored in detail. Thus, the large amount of similar partitions will not be produced by MOCLE, resulting in a concise set of solutions.

Finally, the objective functions should represent validation indexes able to measure the quality of partitions in different ways, each one related to a different clustering criterion. They should also complement each other. In our experiments, we have used the same measures employed in Handl and Knowles (2007): overall deviation and connectivity. The overall deviation of a partition measures the overall summed distances between objects and their corresponding cluster center. This measure is strongly biased toward spherically shaped clusters and improves with the increase in the number of clusters. The connectivity reflects how often neighboring objects have been placed in the same cluster. It improves with the decrease in the number of clusters. The connectivity is able to detect arbitrarily shaped clusters, but it is not robust to overlapping clusters. These two objectives, to be minimized, balance each other's tendency to increase or decrease the number of clusters, avoiding the convergence to trivial solutions. The objective functions are responsible for the selection of the high-quality partitions and the

robustness of MOCLE with respect to different data conformation.

Prior knowledge regarding one known structure of the data can also be integrated into MOCLE, helping the discovery of other structures. This can be accomplished by using an extra objective function that, for example, takes external information into account. We have investigated the information gain measure for this purpose (Raileanu & Stoffel, 2004). Such a characteristic is very useful, for instance, in an application whose aim is the discovery of disease subtypes by means of gene expression data analysis (Alizadeh et al., 2000; Azuaje, 2000; Bittner et al., 2000; Golub et al., 1999; Sorlie et al., 2001; Yeoh et al., 2002). We have developed some experiments in this context using bioinformatics data and found very interesting results (Faceli, 2006).

An important issue regarding our technique is that, like MOCK, it does not require a fine adjustment of parameters for its application to different datasets. The values of the parameters that we have employed depend only on the size of the dataset. Thus, the user easily adjusts the parameters without any additional knowledge on the algorithm or the dataset.

It should also be mentioned that, like other multi-objective clustering techniques, MOCLE is time-consuming. However, there are many applications (e.g., in bioinformatics) where the quality of the solutions obtained is more important than the computational time spent to find them. Besides, the computational time of the clustering method usually represents a small portion of the total time involved in these applications.

In summary, MOCLE automatically performs important steps of cluster analysis. It runs several conceptually different clustering algorithms with various parameter configurations, combines the partitions resulting from these algorithms, and selects the partitions with the best trade-offs for different validation measures. In this manner, MOCLE represents a useful approach to exploratory data analysis. It results in a concise and stable set of high-quality alternative structures without the need of previous knowledge about the data or deep knowledge on cluster analysis. Furthermore, MOCLE allows the automatic integration of previous knowledge. All these characteristics make MOCLE an attractive approach to experts from various domains that are making use of cluster analysis.

APPLICATION

To illustrate the ideas previously discussed, we show the results obtained with the application of different clustering techniques, including ensemble and multi-objective methods, to a dataset with several known structures. We selected a bioinformatics dataset frequently used for cluster analysis: the gene expression data from acute leukemia patients described in Golub, et al. (1999). This dataset has 72 objects (examples) with 3571 attributes (genes). For our experiments, we consider as known structures four distinct existing classifications for this dataset (E1, E2, E3, and E4). The two main classifications refer to types and subtypes of acute leukemia: E1 classifies the examples into Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). E2 includes a refinement of the ALL class, dividing the data into the classes AML, T-ALL (T-lineage ALL), and B-ALL (B-lineage ALL). The other structures correspond to different types of information. E3 classifies the examples according to the institution from which the examples came: DFCI (Dana-Farber Cancer Institute), CALGB (Cancer and Leukemia Group B), SJCRH (St. Jude Children's Research Hospital), and CCG (Children's Cancer Group). E4 defines if the examples are from bone marrow (BM) or from peripheral blood (PB). For the experiments, the dataset was preprocessed in the same way as in Golub, et al. (1999).

More specifically, in this section, we present the results obtained with the application of

four traditional clustering algorithms (*k*-means, average-link, single-link, and shared nearest neighbors [SNN]) (Ertöz, Steinbach & Kumar, 2002; Jain & Dubes, 1988), the ensemble method of Strehl and Ghosh (2002) (ES for short), the multi-objective clustering MOCK, and our multi-objective clustering ensemble MOCLE. The ES's code was obtained at http://www.lans.ece.utexas.edu/~strehl/soft.html, and MOCK's code was obtained at http://dbkgroup.org/handl/mock/. In order to evaluate the results, we used the corrected Rand index, *CR* (Jain & Dubes, 1988). This index measures the similarity between two partitions. A value of *CR* close to 0 means that the clustering method produced a random partition, and close to 1 indicates a perfect match between the partitions.

To run the traditional algorithms, we adjust their parameters to generate partitions with a number of clusters *k* varying from 2 to 8. These values are, respectively, the number of clusters in E1 and two times the number of clusters in E2. The *k*-means algorithm was run 30 times for each *k*, with a random choice of initial centers. For the average-link and single-link algorithms, we generated the hierarchies and cut them in order to produce one partition for each value of *k*. In the case of SNN, we ran it with several values for its parameters *NN* (2%, 5%, 10%, 20%, 30%, and 40% of the number of objects in the dataset), *topic* (0, 0.2, 0.4, 0.6, 0.8, and 1), and *merge* (0, 0.2, 0.4, 0.6, 0.8, and 1). Preliminary experiments showed that variations of the other parameters did not produce very different results. Thus, we used the default value for the parameter *strong* and the value 0 for the parameters *noise* and *label*. From the partitions created with these parameter values, we selected only the partitions having *k* in the interval of interest. To run the *k*-means, average-link, and single-link algorithms, we employed the software Cluster 3.0, which is available at http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/software.htm. For SNN, its authors sent us their implementation of the algorithm.

For the experiments with MOCLE, we employed the partitions generated with the traditional algorithms as the initial population. In the case of *k*-means, among all 30 partitions produced for a given *k*, we selected the partition with the lowest squared error for the initial population. This minimizes the occurrence of suboptimal solutions. The maximum number of generations used was defined as 50. In the first experiments, we observed that increasing the number of generations did not modify the Pareto front approximation obtained. The internal population size used was 46; that is, the number of partitions generated with the traditional algorithms (the base partitions for the ensemble). The number of nearest neighbors used to calculate the connectivity was set to 4 (5% of the number of objects in the dataset). In this section, we show only the results obtained in the experiments using NSGA-II and the crossover implemented with MCLA. We include two versions of MOCLE: one that does not consider prior knowledge, referred to as MOCLE; and a version that includes prior knowledge via the use of information gain, referred to as MOCLE-Gain. In the latter, the structure E1 is used as prior knowledge.

In the case of MOCK, we set the number of nearest neighbors to the same value used with MOCLE and the maximum number of clusters to 8. We employed five control fronts. We used the default values for all other parameters. For the ensemble, we considered the same initial population used by MOCLE. For the number of clusters, *k*, we adopted the same range used for the individual algorithms.

As previously mentioned, MOCLE and MOCK produce a set of n^S solutions, $\Pi_S = \{\pi^{S1}, \pi^{S2}, ..., \pi^{Sn^S}\}$. The individual algorithms and the ES do not generate a set of solutions. Thus, in order to include the results obtained by these techniques in the comparison, we form a set of solutions for each algorithm. This is accomplished by putting together the partitions generated for the different values of *k* used. For

k-means, we obtained 30 sets of solutions, one for each run.

For the deterministic techniques (average-link, single-link, and SNN), there is just one set of solutions, Π_S. In these cases, we calculated the *CR* between each solution partition, $\pi^{Si} \in \Pi^S$, and each known structure, *Ej*. Next, for each known structure, *Ej*, we selected the best partition in Π_S (the partition π^{Si} with the highest *CR*, when compared with *Ej*). These values are shown in Table 1.

As ES, MOCLE, and MOCK, like the *k*-means, are not deterministic, we run them 30 times with the same initial configurations. For these techniques, the experiments produced 30 sets of solutions, $\Pi_{S_1} ... \Pi_{S_{30}}$. For each set, Π_{S_l}, we calculated the *CR* between each solution partition, $\pi^{S_l i} \in \Pi_{S_l}$ and each known structure *Ej*. Next, for each known structure *Ej*, we selected the best partition in each Π_{S_l} (the partition $\pi^{S_l i}$ with the highest *CR* when compared with *Ej*).

Finally, for each known structure, we calculated the mean and standard deviation of the *CR* for the 30 selected partitions (one for each Π_{S_l}). For the MOCK algorithm, we considered two sets of solutions: the complete Pareto front approximation obtained (MOCK) and a reduced set of solutions recommended as the best solutions of the Pareto front approximation (MOCK-R). These values are shown in Table 1, with the highest value of *CR* for each known structure highlighted in boldface.

Looking at the traditional algorithms, the first aspect that we can observe is that for each known structure, a different algorithm achieved the best performance. This illustrates the previously discussed difficulty in the choice of an appropriate algorithm to be used for a particular dataset.

Now, turning the attention to MOCK and ES, for each known structure, we can observe that they did not outperform the best traditional algorithm. Nevertheless, they not only presented a relatively high performance, but also such a performance was uniform across the different structures when compared to the traditional algorithms. This shows that these strategies for the combination of different clustering criteria can be more robust with respect to different data conformations (observed in the different known structures) than the traditional algorithms. In contrast, MOCK-R showed a very poor performance for all known structures. That is, although MOCK was able to find relatively good solutions, the heuristic used to build MOCK-R was not able to identify them.

MOCLE and MOCLE-Gain, in their turn, showed a similar or superior performance for all known structures when compared to all other techniques. More specifically, in all cases, MOCLE presented the same performance obtained by the best traditional algorithm. This means that

Table 1. Performance of the clustering techniques

Technique	E1	E2	E3	E4
single-link	0.078	0.003	0.108	**0.315**
SNN	0.855	0.855	0.677	0.112
average-link	0.876	0.798	0.693	0.057
k-means	0.507 (0.211)	0.502 (0.127)	0.402 (0.147)	0.024 (0.015)
MOCK	0.684 (0.059)	0.795 (0.059)	0.622 (0.055)	0.057 (0.008)
MOCK-R	0.012 (0.138)	0.262 (0.141)	-0.015 (0.154)	-0.070 (0.043)
ES	0.743 (0.175)	0.637 (0.129)	0.589 (0.124)	0.110 (0.074)
MOCLE	0.876 (0)	0.855 (0)	0.693 (0)	**0.315** (0)
MOCLE-Gain	**0.942** (0.037)	**0.857** (0.009)	**0.723** (0.015)	**0.315** (0)

MOCLE was able to keep the high-quality solutions belonging to the initial population, which were found by the traditional algorithms, in the final set of solutions.

Indeed, MOCLE-Gain was even able to improve the quality of initial solutions found for the known structures E1, E2, and E3. This was, in fact, expected for the structure E1, since such a structure was provided to the algorithm as prior knowledge. The superiority of MOCLE-Gain over MOCLE for the structures E2 and E3 illustrates that the integration of prior knowledge from one structure can be useful for the analysis of the other structures.

For the structure E4, all techniques showed a very poor performance. This is clearly the case where the classification is not consistent with at least one of the clustering criteria optimized. More precisely, classes are entities related to categories previously defined in the real world to organize the objects. Clusters, on the other hand, are entities defined by the application of mathematical/statistical concepts to the data. The classes can be related to one or more of the mathematical/statistical concepts. Here, we assume that the known classifications are in accordance with some of the clustering criteria used. However, a classification could be unrelated to a clustering criterion. This would result, as in the case for E4, in a low performance for all clustering techniques.

It is important to mention that we used the *CR* only for evaluation purposes. Among all partitions from a set of solutions, we selected the partition closest to each known structure only to emphasize the ability of MOCLE in the recovery of more than one structure. MOCLE does not provide a method to select the best partitions from the solution set, as MOCK does. Nevertheless, the results of our experiments showed that the solution set generated with MOCLE is more concise than MOCK (the version without the selection heuristic), allowing the domain expert to individually analyze each of the solutions and identify the high-quality ones.

For instance, in the experiments reported, we obtained with MOCK a set of 79 solutions on average, with a standard deviation of 10.6. In contrast, for MOCLE, we obtained on average a set of 14 solutions (with a standard deviation of 0.3). In the context of MOCLE-Gain, we achieved a set of 20 solutions on average (with a standard deviation of 1.6). Finally, it is important to point out that although MOCK recommends only one or two best solutions (MOCK-R), the quality of these solutions is very poor, as previously mentioned.

Now if we observe the standard deviations, we can see that the ones calculated from the solutions found with MOCLE and MOCLE-Gain were smaller than those of the other techniques. A low standard deviation in our experiments means that one partition with a similar quality with respect to a given known structure was found in each run.

In summary, the experimental results presented in this section illustrate some of the problems of the traditional clustering techniques, the abilities of the ensembles, and the multi-objective clustering techniques, as well as how the combination of all these approaches could be useful for the experts in the data domain performing cluster analysis.

FUTURE TRENDS

Most of the directions for future work discussed in previous publications on cluster ensembles and multi-objective clustering methods are related to the adjustment of their components in order to improve the quality of their results. For the cluster ensembles, this involves the generation of the base partitions (trying different algorithms or different strategies) and the consensus function (investigating small changes of the existing functions or evaluating new functions). In the case of multi-objective clustering methods, much effort has been spent in the investigation of different objective functions, optimization strategies (e.g.,

genetic algorithms), representation of the solutions, operators, and strategies to construct the initial population.

MOCLE can benefit from several of these proposals. For example, a straightforward improvement for the MOCLE algorithm is the inclusion of new clustering algorithms in the generation of the base partitions. There are novel clustering algorithms that, by themselves, overcome some of the difficulties found in the more traditional algorithms. This could lead to an improvement in the quality of the solutions obtained. The use of other objective functions or genetic algorithms could also lead to improvements and are objects of future research.

These more basic research directions, together with the application of the proposed techniques to other domains, are important for the consolidation of the approaches discussed.

The approaches presented here are usually time-consuming, mainly the multi-objective alternatives. This restricts their applicability in problems with hard time constraints. Hence, an important concern for future efforts is the reduction in their time cost (Handl, 2006; Navernioukk, 2005). Furthermore, the study of strategies to reduce the number of solutions returned by the multi-objective approaches is also a challenging topic for future research. For some algorithms, such as MOCK, more general and sophisticated strategies to select the best solutions are required (Handl, 2006; Navernioukk, 2005).

For the cluster ensemble approaches, the development of methods to find the best number of clusters is an issue that has received increasing attention and also constitutes a subject of interest for further investigation (Kuncheva et al., 2006; Monti et al., 2003; Strehl & Ghosh, 2002).

Regarding the types of data and structures the clustering approaches discussed in this chapter can deal with, the main trends have been the extension of these approaches to fuzzy clustering, where the clusters can share objects (Ayad & Kamel, 2003; Law et al., 2004; Strehl & Ghosh, 2002); and to biclustering, which consists of grouping objects and attributes at the same time (Fern & Brodley, 2004; Handl, 2006, Mitra & Banka, 2006). The application of the discussed approaches to heterogeneous databases, where the data come from different sources, is an issue that started to be investigated in Strehl and Ghosh (2002) and showed to be another promising direction for future research (Jouve & Eric, 2003; Kasturi & Acharya, 2005; Tanay, Steinfeld, Kupiec, & Shamir, 2005).

Another issue that deserves attention is the representation of a set of partitions in such a way that the domain experts can easily compare them. For this, techniques to support the visualization of partitions are of great importance (Faceli, 2006; Faceli, Carvalho, & Souto, 2005; Handl, 2006; Monti et al., 2003).

CONCLUSION

Cluster analysis is a research area that has been active for many decades and is continually watching the proposal of new algorithms, methods, and evaluation criteria, addressing limitations found in previous techniques or providing solutions to new challenging problems.

In this chapter, we presented an overview of techniques based on the combination of different clustering criteria for the solution of common problems found in traditional cluster analysis approaches. First, we discussed these basic difficulties, which encompass the inexistence of a precise definition of cluster, the need of expert knowledge to select the best algorithms and to adjust their parameters, the heterogeneity of the clusters belonging to a certain dataset, and the possible existence of more than one structure in a dataset. Next, we presented more sophisticated approaches to deal with these difficulties, cluster ensembles and multi-objective clustering algorithms; discussed how they can overcome the previous difficulties; and pointed out their

limitations. In this context, as the central topic of the chapter, we presented our multi-objective clustering ensemble algorithm, which combines characteristics from the more sophisticated approaches described in a way that many of their own difficulties are minimized. We also presented an illustrative application example. The characteristics, advantages, and limitations of our approach were also described in detail.

Regarding relevant future research in the context of cluster ensemble and multi-objective clustering methods, we pointed out a few issues, such as the need to reduce the computational time spent by the existing techniques to find a suitable solution. The issues discussed in this chapter have also started to be applied to other types of clustering tasks, such as fuzzy clustering and biclustering.

REFERENCES

Alizadeh, A.A., et al. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, *403*(6769), 503–511.

Asur, S., Parthasarathy, S., & Ucar, D. (2006). An ensemble approach for clustering scalefree graphs. *Proceedings of the Workshop on Link Analysis: Dynamics and Static of Large Networks.* Philadelphia, Pennsylvania.

Ayad, H., & Kamel M.S. (2003). Refined shared nearest neighbors graph for combining multiple data clusterings. *Proceedings of the 5th International Symposium on Intelligent Data Analysis. Lecture Notes in Computer Science*, *2810*, 307–318.

Azuaje, F. (2000). *Gene expression patterns and cancer classification: A self adaptive and incremental neural approach* (Tech. Rep. TCD-CS-2000-26). Dublin: Trinity College, Computer Science Department.

Barbara, D. (2000). *An introduction to cluster analysis for data mining.* Retrieved November 12, 2003, from http://www-users.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf.

Bittner, M., et al. (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, *406*(6795), 536–540.

Boulis, C., & Ostendorf, M. (2004). Combining multiple clustering systems. *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Lecture Notes in Computer Science*, *3202*, 63-74.

Cheng, H., Cao, J., Wang, X., & Das, S.K. (2006). Stability-based multi-objective clustering in mobile ad hoc networks. *Proceedings of the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, Waterloo, Ontario, 191, 27.

Costa, I.G., Carvalho, F.A.T., & Souto, M.C.P. (2004). Comparative analysis of clustering methods for gene expression time course data. *Genetics and Molecular Biology*, *27*(4), 623–631.

Deb, K., Pratap, A., Agarwal, S., & Meyrivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

D'haeseleer, P. (2005). How does gene expression clustering work? *Nature Biotechnology*, *23*, 1499–1501.

Eisen, M.B., Spellman, P., Brown, P., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, *95*(25), 14863–14868.

Ertöz, L., Steinbach, M., & Kumar, V. (2002). A new shared nearest neighbor clustering algorithm and its applications. *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications, 2nd SIAM International Conference on Data Mining*, 105–115.

Estivill-Castro, V. (2002). Why so many clustering algorithms—a position paper. *SIGKDD Explorations, 4*(1), 65–75.

Faceli, K. (2006). *Um framework para análise de agrupamento baseado na combinação multi-objetivo de algoritmos de agrupamento (A framework for cluster analysis based in a multi-objective combination of clustering algorithms)* [doctoral thesis]. São Carlos: Institute of Mathematics and Computer Science, University of São Paulo.

Faceli, K., Carvalho, A., & Souto, M. (2005). Evaluation of the contents of partitions obtained with clustering gene expression data. *Proceedings of the Brazilian Symposium on Bioinformatics, Lecture Notes in Computer Science*, 3594, 65–76.

Faceli, K., Carvalho, A.C.P.L.F., & Souto, M.C.P. (2006). Multi-objective clustering ensemble. *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, Auckland, New Zealand, 51.

Fern, X.Z., & Brodley, C.E. (2004). Solving cluster ensemble problems by bipartite graph partitioning. *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta, 69, 36.

Fred, A.L.N., & Jain, A.K. (2005). Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 835–850.

Golub, T.R., et al. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science, 286*(5439), 531–537.

Greene, D., & Cunningham, P. (2006). *Efficient ensemble methods for document clustering* (Tech. Rep. TCD-CS-2006-48). Dublin: Trinity College, Department of Computer Science.

Greene, D., Tsymbal, A., Bolshakova, N., & Cunningham, P. (2004). Ensemble clustering in medical diagnostics. *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems*, 576–581).

Handl, J. (2006). *Multiobjective approaches to the data-driven analysis of biological systems* [doctoral thesis]. Manchester, UK: University of Manchester, School of Chemistry.

Handl, J., & Knowles, J. (2005). Computational cluster validation in postgenomic data analysis. *Bioinformatics, 21*(15), 3201–3212.

Handl, J., & Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation, 11*(1), 56–76.

Hautaniemi, S., et al. (2003). Analysis and visualization of gene expression microarray data in human cancer using self-organizing maps. *Machine Learning, 52*(1-2), 45–66.

Hu, X. (2006). Gene-miner: Integration of cluster ensemble and text mining for comprehensive gene expression analysis. *International Journal of Bioinformatics Research and Application, 2*(3), 325–338.

Hu, X., Yoo, I., Zhang, X., Nanavati, P., & Das, D. (2005). Wavelet transformation and cluster ensemble for gene expression analysis. *International Journal of Bioinformatics Research and Applications, 1*(4), 447–460.

Jain, A., & Dubes, R. (1988). *Algorithms for clustering dat*a. Prentice Hall.

Jain, A.K., Myrthy, M.N., & Flynn, P.J. (1999). Data clustering: A survey. *ACM Computing Survey, 31*(3), 264–323.

Jouve, P., & Eric, N.N.L. (2003). A new method for combining partitions, applications for cluster ensembles in KDD. *Proceedings of Parallel and Distributed Computing for Machine Learning,* in conjunction with *the 14th European Conference on Machine Learning* and *7th European Conference*

on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia.

Kasturi, J., & Acharya, R. (2005). Clustering of diverse genomic data using information fusion. *Bioinformatics*, *21*(4), 423–429.

Kleinberg, J. (2002). An impossibility theorem for clustering. *Advances in Neural Information Processing Systems, 15*, 446–453.

Korkmaz, E.E., Du, J., Alhajj, R., & Barker, K. (2006). Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis, 10*(2), 163–82.

Kuncheva, L.I., Hadjitodorov, S.T., & Todorova, L.P. (2006). Experimental comparison of cluster ensemble methods. *Proceedings of FUSION 2006*, 105–115.

Law, M., Topchy, A., & Jain, A.K. (2004). Multiobjective data clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2, 424–430.

Liu, Y., Ram, S., & Lusch, R. (2005). A unified market segmentation method for generating Pareto optimal sets. *Proceedings of the 15th Workshop on Information Technology and Systems*, Las Vegas, Nevada.

Lourenco, A., & Fred, A. (2005). Ensemble methods in the clustering of string patterns. *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 1, 143–148.

Mitra, S., & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition*, *39*(12), 2464–2477.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, *52*(1-2), 91–118.

Munson, A., & Caruana, R. (2006). *Cluster ensembles for network anomaly detection* (Tech. Rep. 2006-2047). Ithaca, NY: Cornell University, Department of Computer Science.

Narayanan, E.K.A. (2005). *Intelligent bioinformatics: The application of artificial Intelligence Techniques to bioinformatics problems*. John Wiley & Sons.

Navernioukk, I. (2005). *Multiobjective graph clustering with variable neighbourhood descent* [master's thesis]. University of British Columbia, Canada.

Nikkilä, J., Törönen, P., Kaski, S., Venna, J., Castrén, E., & Wong, G. (2002). Analysis and visualization of gene expression data using self-organizing maps. *Neural Networks, Special Issue on New Developments on Self-Organizing Maps, 15*(8-9), 953–966.

Pal, N.R., Aguan, K., Sharma, A., & Amari, S. (2007). Discovering biomarkers from gene expression data for predicting cancer subgroups using neural networks and relational fuzzy clustering. *BMC Bioinformatics, 8*(5). Retrieved January 19, 2007, from http://www.biomedcentral.com/1471-2105/8/5.

Raileanu, L.E., & Stoffel, K. (2004). Theoretical comparison between the Gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, *1*(41), 77–93.

Ripon, K.S.N., Tsang, C., & Kwong, S. (2006). Multi-objective data clustering using variable-length real jumping genes genetic algorithm and local search method. *Proceedings of the International Joint Conference on Neural Networks*, 3609–3616.

Sorlie, T., et al. (2001). Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98, 10869–10874.

Souto, M.C.P., Araujo, D.S.A., & Silva, B.L.C. (2006). Cluster ensemble for gene expression microarray data: Accuracy and diversity. *Proceedings of the IEEE International Joint Conference on Neural Networks*, 16, 2174–2180.

Spellman, P., et al. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, *9*, 3273–3297.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles—A knowledge reuse framework for combining partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Tamayo, P., et al. (1999). Interpreting patterns of gene expression with selforganizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences*, 96, 2907–2912.

Tanay, A., Steinfeld, I., Kupiec, M., & Shamir, R. (2005). Integrative analysis of genome-wide experiments in the context of a large high-throughput data compendium. *Molecular Systems Biology*, *1*.

Topchy, A., Jain, A., & Punch, W. (2004). A mixture model for clustering ensembles. *Proceedings of the SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, 331–338.

Topchy, A., Jain, A., & Punch, W. (2005). Clustering ensembles: Models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(12), 1866–1881.

Wang, J.T.L., Zaki, M.J., Toivonen, H.T.T., & Shasha, D.E. (Eds.). (2003). *Data mining in bioinformatics: Advanced information and knowledge processing.* Springer.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, *16*(3), 645–678.

Yeoh, E.J., et al. (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, *1*(2), 133–143.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

Chapter XVI
Implementing Negative Correlation Learning in Evolutionary Ensembles with Suitable Speciation Techniques[1]

Peter Duell
The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, UK

Xin Yao
The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), University of Birmingham, UK

ABSTRACT

Negative correlation learning (NCL) is a technique that attempts to create an ensemble of neural networks whose outputs are accurate but negatively correlated. The motivation for such a technique can be found in the bias-variance-covariance decomposition of an ensemble of learner's generalization error. NCL is also increasingly used in conjunction with an evolutionary process, which gives rise to the possibility of adapting the structures of the networks at the same time as learning the weights. This chapter examines the motivation and characteristics of the NCL algorithm. Some recent work relating to the implementation of NCL in a single objective evolutionary framework for classification tasks is presented, and we examine the impact of two speciation techniques: implicit fitness sharing and an island model population structure. The choice of such speciation techniques can have a detrimental effect on the ability of NCL to produce accurate and diverse ensembles and should therefore be chosen carefully. This chapter also provides an overview of other researchers' work with NCL and gives some promising future research directions.

INTRODUCTION

Ensembles of neural networks remain at the forefront of current pattern recognition research. Ensembles offer the possibility to improve on the performance of single learners if their members can meet the criteria of being both *accurate* and *diverse* (Hansen & Salamon, 1990; Krogh & Vedelsby, 1995; Opitz & Shavlik, 1996; Dietterich, 2000). The design of ensembles of neural networks that meet these criteria, however, is not an easy task, since there exists a trade-off between accuracy and diversity. This chapter presents a researcher's analysis of the issue of diversity in ensembles and discusses one method for managing the trade-off during learning: negative correlation learning (NCL) (Liu, 1998). NCL proposes a penalty term for correlated individuals during learning and therefore encourages networks to be both accurate and *negatively correlated*. NCL has been used in traditional ensemble learning (Brown, 2001, 2003; Islam, Yao & Murase, 2003; Liu, 1998; Liu & Yao, 1999) as well as in combination with evolutionary approaches (Chandra & Yao, 2004, 2006; Liu, Yao & Higuchi, 2000). NCL takes its inspiration from the analysis of regression ensembles, but it has also been applied with success to classification tasks (Liu & Yao, 1998).

Evolutionary ensemble learning is an interesting field of research since it offers the possibility to not only manage the accuracy-diversity trade-off, but also to explore other areas of ensemble design. We need not limit ourselves to adapting weights, but also individual network structures (Yao, 1999). One evolutionary approach that utilizes NCL and is also readily adaptable to evolving network structures is evolutionary ensembles with negative correlation learning (EENCL) (Liu et al., 2000). This chapter analyzes the use of NCL in EENCL and also the suitability of the speciation technique used. NCL is found to be ineffective in EENCL as it was presented by Liu, and similar results can be achieved by a simpler local search technique (Duell, Fermin & Yao, 2006a). EENCL uses implicit fitness sharing to promote diversity at the evolutionary level (Liu et al., 2000), and this renders NCL ineffectual (Duell et al., 2006a). We also present our experiments with an alternative speciation technique in Island-model with negative correlation learning (INCL) and find this method much better suited for use with NCL (Duell, Fermin & Yao, 2006b). We compare EENCL and INCL to some other popular ensemble techniques on a number of pattern recognition tasks and find that EENCL often performs poorly by comparison. INCL, however, provides comparable performance to the other methods tested, while still offering the same adaptability as EENCL to evolving structures as well as weights. Our analysis of the bias, variance, and covariance of INCL and EENCL on an artificial regression task suggest that INCL is more effective at reducing the covariance of the ensemble.

The rest of this chapter contains a discussion on the concept of diversity in ensembles of neural networks, and the analysis of a number of researchers is presented. NCL is explained, along with a brief overview of some of the work undertaken with this technique. We present our work by using NCL in a single-objective evolutionary framework, and finally, we draw conclusions about such an approach and indicate some promising future directions.

DIVERSITY AND MOTIVATION FOR ENSEMBLES

Importance of Diversity

The following sections detail a number of methods that have been used to justify and guide ensemble research; in particular, the need for diversity and what this term means. The literature mostly deals with regression tasks to separate these methods from classification (i.e., a zero-one loss function), which is much less understood theoretically. For

neural networks outputting real values, however, we can reformulate the classification task into a prediction of posterior probabilities, and then the following analysis continues to hold (Brown, Wyatt, Harris & Yao, 2005).

Ensembles of Independent Estimators

Perrone and Cooper (1993) provided the mathematical justification for the combination of neural networks into an ensemble. The outputs of each network first are combined by means of a simple average (basic ensemble method) and then by a weighted average (generalized ensemble method).

The errors of the individual networks are assumed to be mutually independent with zero mean for the basic ensemble method. Under these conditions, they show that the average mean square error for the population can be reduced by a factor equal to the number of networks in the ensemble. If these conditions hold, then simply increasing the number of networks will reduce the error to an arbitrarily level. In practice, of course, the networks to be ensembled will not meet these assumptions, especially as the number of networks increases.

To remove the need for these two unrealistic assumptions to hold, they propose a generalized method. In this method, the sample correlation matrix is used as an estimate of the actual correlation matrix of the networks. This correlation matrix is then used to find an optimal weight for a weighted average of the outputs of the networks. This is guaranteed to be better than the best individual and the simple average mentioned previously, since both methods are special cases of the generalized method. Two assumptions are necessary for this method to work. First, the correlation matrix is an estimate, not the true correlation matrix. As such, it is assumed that this estimate is accurate in order to find optimal weights. Second, inverting the correlation matrix requires the columns and rows to be linearly independent, or the inversion will be unstable. A linear dependency would arise when two networks are almost identical. This method, therefore, requires a degree of vetting of candidate ensembles to remove any that are too similar.

Although the generalized method seems to promise a guaranteed improvement in generalization performance, it still relies on assumptions that cannot be guaranteed to hold. Although problems with linear dependency can be eliminated by discarding some networks, similar networks on the training data may prove very different on unseen data. Discarding these networks may result in a loss to possible ensemble performance that is not immediately obvious from the training data alone. More significantly, the assumption that the training sample correlation matrix is an accurate estimate of the true correlation matrix may not hold, particularly for limited training data sets or for noisy data.

Turner and Ghosh (1996a, 1996b) arrived at the following equation for the *added error* of an ensemble of posterior probability estimators. The *added error* is the error in excess of the *Bayesian error*; that is, the error occurring when the decision boundary is at the intersection of two true posterior probability curves (Equation (1)), where E^{ens}_{add} is the expected added error for the ensemble, E_{add} is the expected added error of the individual members, and M is the number of learners and a measure of correlation.

Equation (1).

$$E^{ens}_{add} = E_{add}\left(\frac{1+\delta\,(M-1)}{M}\right) \tag{1}$$

In this analysis, the errors of the individual classifiers are assumed to be equal and have the same variance, and the probability curves are monotonic at the decision boundary. The individuals are combined by a simple average. Under these assumptions, we arrive at Equation (1). When it equals 1, the individual members are perfectly correlated (i.e., the same posterior probability curves), and the ensemble error is just that of the individual members. If it equals 0, the individuals have no correlation (are independent), and the ensemble error is reduced by a factor of M. These conclusions mirror those of Perrone and Cooper's (1993) but are based on similar assumptions. However, like Perrone and Cooper's (1993) work, it does illustrate the potential of the idealized ensemble, justifying further work on how we can create constituent members that approach these assumptions.

The Ambiguity Decomposition

Krogh and Vedelsby (1995) decomposed the generalization error of an ensemble of regressors on a single dataset into the errors and ambiguity of the networks. They show that the generalization error for the ensemble must be less than the weighted average of the network generalization errors, and that this difference is increased as the weighted average ambiguity of the networks increases. In other words, the greater the output of the individual networks vary from the average output, the greater the difference between the ensemble error and the (weighted) average errors of the networks. Our aim then should be to create accurate and diverse (uncorrelated) outputs.

It is significant that there is no assumption here that individual estimators should be independent, unlike Perrone and Cooper's (1993) model. The gains from the ensemble are unquantified in this case, but the ambiguity is naturally at its highest when estimators are completely independent, and Perrone and Cooper's equation can be seen as the upper limit achievable. In reality there is a trade-off here between increasing the ambiguity and decreasing the accuracy (Krogh & Vedelsby, 1995).

Bias-Variance Dilemma

While the ambiguity decomposition is carried out for a single training set, an alternative approach considers the generalization error over all possible training sets for an ensemble. First, we must consider a single network before extending the analysis to a group.

The generalisation error of an estimator can be decomposed into two components: bias and variance (Geman, Bienenstock & Doursat, 1992). The bias (squared) and variance terms of the error are as follows in Equations (2) and (3). E_D represents the expectation over all datasets, *f(x; D)* is the output of the classifier trained on dataset *D*, and *E[y|x]* is the expected (target) output given input ***x***.

Bias represents how far on average the model differs from the desired function over the entire space of possible training sets. The variance is how sensitive the model is to randomness involved in training, including the dataset used and, for neural networks, the initial weights.

These two concepts are illustrated graphically in Figure 1 for a training set randomly sampled from the true function plus an element of noise.

The high-bias estimate in Figure 1 shows an estimator that disregards the training data com-

Equations (2) and (3).

$$E_{\mathcal{D}}[(f(\mathbf{x}) - E[y|\mathbf{x}])^2] = (E_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - E[y|\mathbf{x}])^2 + E_{\mathcal{D}}[(f(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})])]^2 \quad (2)$$

$$= Bias^2 + Variance \quad (3)$$

pletely. One such estimator would be a network whose weights are arbitrarily initialized and no training takes place. The network is likely to display a high bias since the function it forms is unlikely to represent the desired function. However, the function will not change, regardless of what sample of the training data population is used (since the network simply ignores it and sticks to its initialization), and so it will have no variance.

The high-variance estimate in Figure 1 overfits to the training data. This could be a network with a sufficient number of weights, trained on the data until the training error is very small. The network's function, however, is overfitted to the particular data points in the training set and does not represent the true function. This network displays a high variance since a different set of training data would yield a different function. The bias of this network, however, is lower since it is closer on average to the true function.

A parametric model that assumes a linear function will display a high bias when trying to represent a quadratic function, since it cannot accurately approximate its true form. Neural networks with two layers of logistic functions are capable of approximating any function (Bishop, 1996). Such a network can be seen as a nonparametric model, since it makes no assumptions about the function it seeks to represent. Consequently, a neural network with two logistic layers is in principle capable of achieving arbitrarily low bias. However, variance can only be guaranteed to be minimized if the training set is infinitely large. The complexity necessary to model an arbitrary function and the availability of only limited data can result in overfitting to the training data and, therefore, to a high variance.

Generally, we can say that by training a network, we are reducing the bias. However, as the network continues to be trained, variance will increase as the network minimizes the error between its output and the training data, the network fits itself more and more to the training data. This is the problem of overfitting. The reduction in bias caused by training is eventually exceeded by the increase in variance, and so the overall generalization error begins to increase. In this way, there is a trade-off between bias and variance (Figure 2).

Bias, Variance, and Covariance

Extending the previous analysis for a single network to an ensemble, we decompose the variance term further into a variance and covariance (Ueda & Nakano, 1996). Here, f_i is the output of network i, $<d>$ is the target output, and *bias_avg*, *var_avg* and *covar_avg* are the average of the bias, variance and covariance across the networks. *E*

Figure 1. Illustration of high-bias and high-variance estimators in relation to the desired function

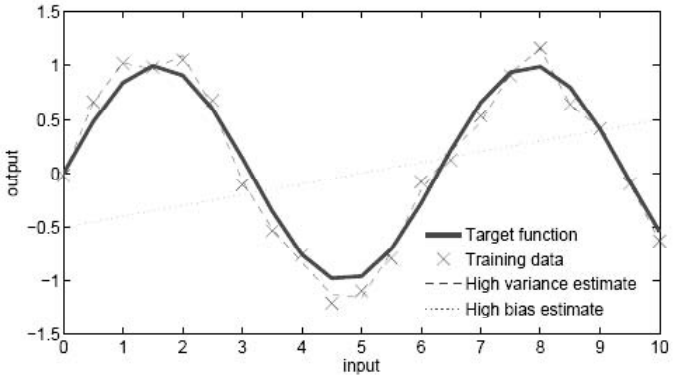
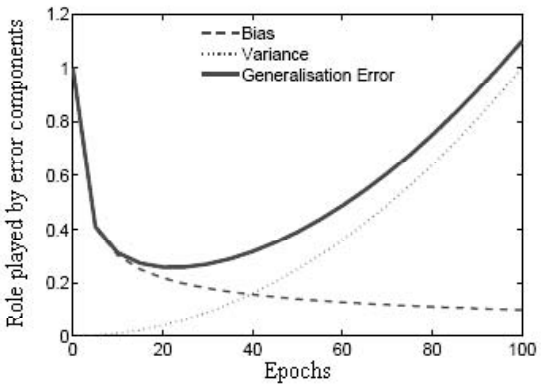


Figure 2. The bias/variance trade-off



stands for the expectation across all datasets (see Equation (4)).

Brown, et al. (2005) have recently shown that the previous decomposition is, in fact, equivalent to the ambiguity decomposition when taken over the set of possible datasets.

If the *covar* term is reduced to zero (uncorrelated network outputs), then the overall variance term is reduced by a factor of M, the number of networks in the ensemble. Both the *bias* and *var* terms are constrained to be positives, but interestingly, the *covar* can be either positive or negative. As such, an even greater reduction in the generalization error can be achieved, if the network outputs are *negatively* correlated.

Diversity Creation Taxonomy

One attempt to provide a taxonomy of diversity creation methods was proposed by Sharkey (1999). Sharkey defines all ensemble techniques as encouraging diversity by varying either the initial weights, training data, architectures, or training algorithms. However, recent methods do not fit well into these categories. Brown, et al. (2005) suggest an alternative taxonomy that differentiates between explicit and implicit methods. The former deterministically encourage diversity in the ensemble, whereas the latter rely on randomness. Current approaches also can be categorized

Equation (4).

$$E\left(\left(\frac{1}{M}\sum_i f_i - < d >\right)^2\right) = bias\_avg^2 + \frac{1}{M}var\_avg + \left(1+\frac{1}{M}\right)covar\_avg$$

$$bias\_avg = \frac{1}{M}\sum_i (E(f_i) - < d >)$$

$$var\_avg = \frac{1}{M}\sum_i E((f_i - E(f_i))^2) \qquad (4)$$

$$covar\_avg = \frac{1}{M(M-1)}\sum_i \sum_{j\neq i} E((f_i - E(f_i))(f_j - E(f_j)))$$

according to their manipulation of the hypothesis space into the following: starting point in hypothesis space, set of accessible hypotheses, and traversal of hypothesis space (Brown et al., 2005). A learner represents an hypothesis as to the true function or decision boundary we seek to represent. The hypothesis space is the space of all possible hypotheses and hence translates to the set of all possible learners. For an MLP with a single hidden layer of arbitrary length, there exists a representable hypothesis space. Hypotheses outside this space are not representable by the learner. Random initialization of weights corresponds to a single point in a set of feasible hypotheses, and the object of learning is to move toward the true hypothesis. We can create diversity in the ensemble either by varying the starting point, by using different training sets or network structures to vary the accessible regions, or finally, by directly influencing the traversal of the hypothesis space (e.g., by including penalty terms in the learning process).

NEGATIVE CORRELATION LEARNING

Rosen's Decorrelation Penalty Term

Rosen (1996) proposed the inclusion of a penalty term during training to encourage the formation of decorrelated networks. Starting from the bias, variance, and covariance decomposition, Rosen proposed a learning algorithm to minimize the covariance term and hence to decorrelate the ensemble members. Each network is trained in sequence to decorrelate its output from the previous network. A new error function for each learner is introduced in Equation (5), where E_j is the error of network j, y_p is the target output for pattern p, x_p is the input vector for pattern p, f is the output of a network, $\lambda(t)$ is a possibly time-dependent scaling function, d is an indicator function for decorrelation between networks i and j, and P is a correlation penalty function. The purpose of this error function is to minimize the (squared) difference between the target and network output (the first term), and also to apply a penalty to positively correlated networks (second term). Rosen suggests that a suitable correlation penalty is (see Equation (6)), and an indicator function d, shown in Equation (7).

Original NCL Penalty Function

NCL was first proposed by Liu and Yao as a means to generate an ensemble of neural networks whose outputs would be negatively correlated (Liu, 1998). It built upon Rosen's (1996) concept of a penalty term for correlated networks, but here the networks are trained in parallel rather than sequentially. The

Equations (5), (6), and (7).

$$E_j = \sum_{p=1}^{N} \left[\left(y_p - f_j\left(\vec{x}_p\right)\right)^2 + \sum_{i=1}^{j-1} \lambda\left(t\right) d\left(i,j\right) P\left(\vec{x}_p, y_p, f_i, f_j\right) \right], \tag{5}$$

$$P\left(\vec{x}_p, y_p, f_i, f_j\right) = \left(y - f_i\left(\vec{x}_p\right) f_j\left(\vec{x}_p\right)\right), \tag{6}$$

$$d\left(i,j\right) = \begin{cases} 1 \text{ if } i = j - 1 \\ \\ 0 \text{ otherwise} \end{cases}. \tag{7}$$

justification for desiring such networks has been derived earlier in relation to the bias, variance, and covariance decomposition.

Standard back-propagation is used to train the network, but the error to be minimized is now shown in Equation (8) (Rumelhart, Hinton & Williams, 1986; Liu & Yao, 1999), where N is number of training patterns, $E_i(n)$ is the error of network i on training pattern n, $F_i(n)$ is the output of network i on pattern n, λ is the strength of penalty parameter, and $p_i(n)$ is the penalty term, defined for noiseless data as Equation (9).

NCL is an explicit diversity creation method that directly influences the hypothesis space traversal for all ensemble members simultaneously. This is an advantage over methods such as Boosting, where the members are fixed once created, because the algorithm can adapt the composition of its members to take into account the learning of all members, not just previously created ones (Schapire, 1990).

The danger of NCL is that a highly negative correlation among the networks is likely to also have an impact on the bias of the individual members. In fact, a large negative correlation may disguise a very inaccurate learner with a high bias. As such, it is important that the strength parameter be well understood, a task undertaken by Brown (2003).

Amended Derivation of NCL Penalty Function

Islam, et al. (2003) and later Brown (2003) showed that Liu's derivation of the error from (8) to give the derivative shown in Equation (10) is flawed, and that the correct derivation should be as seen in Equation (11), where γ is now used in place of λ as the penalty strength to distinguish it from Liu's derivation.

Conversion from one value to another is possible by means of the equality shown in Equation (12).

Additional NCL Developments

Liu found that NCL was capable of producing biased individual networks whose errors tend to be negatively correlated on both classification and regression tasks (Liu & Yao, 1999). McKay and Abbass (2001) proposed an alternative penalty function in the form of root quadratic negative correlation learning (RTQRT-NCL), which they applied to a genetic programming system and

Equations (8)-(11).

$$E_i = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{1}{2}\left(d\left(n\right) - F_i\left(n\right)\right)^2 + \lambda p\left(n\right)\right] \tag{8}$$

$$p_i\left(n\right) = \left(F_i\left(n\right) - d\left(n\right)\right)\sum_{j\neq i}\left(F_j\left(n\right) - F\left(n\right) - d\left(n\right)\right) \tag{9}$$

$$\frac{\partial E_i\left(n\right)}{\partial F_i\left(n\right)} = \left(1-\lambda\right)\left(F_i\left(n\right) - d\left(n\right)\right) + \lambda\left(F\left(n\right) - d\left(n\right)\right), \tag{10}$$

$$\frac{\partial E_i\left(n\right)}{\partial F_i\left(n\right)} = \left(F_i\left(n\right) - d\left(n\right)\right) - 2\gamma\left(1 - \frac{1}{M}\right)\left(F_i\left(n\right) - F\left(n\right)\right), \tag{11}$$

found an improvement in performance over standard negative correlation learning for large ensembles. RTQRT may be more successful than NCL at producing small widely separated clusters of networks. Brown examined NCL in the wider context of diversity creation techniques, explored bounds on the penalty strength parameter (Brown, 2003; Brown et al., 2005), and also found that NCL is most effective with less complex networks (Brown, 2001). Islam, et al. (2003) used NCL as part of a constructive approach to ensemble design, as well as providing a new version of the penalty strength parameter. The communication costs of NCL have also been reduced through the use of correlation-corrected data (Chan & Kasabov, 2005). NCL has also been applied to the creation of fuzzy rules (Cai, Sun & Jia, 2004).

Multi-objective evolution is also a promising area of current ensemble research increasingly explored (Abbass, 2003; Kottathra & Attikiouzel, 1996; Kupinski & Anastasio, 1999). In multi-objective evolution, we seek to find the set of non-dominated solutions: the pareto optimal front. The concept of a population of good solutions fits well with the concept of an ensemble of good learners. Chandra and Yao (2004, 2006) use the two terms of the NCL error function: accuracy and penalty term (see Equation (8)) as two objectives in a multi-objective approach named DIVACE. The next section of this chapter examines attempts to combine NCL with a single-objective evolutionary approach.

SINGLE-OBJECTIVE EVOLUTION AND NCL

This chapter examines two attempts to combine NCL with a single-objective evolutionary approach.

EENCL Algorithm

Algorithm Description

EENCL uses partial training with the NCL algorithm alongside an evolutionary process to form a population of neural networks suitable for combination into an ensemble (Liu et al., 2000). EENCL exploits two mechanisms to ensure that the final networks are both accurate and diverse. First, NCL encourages the negative correlation of the outputs of the networks in the population. Second, the fitness of individuals in the population is evaluated with implicit fitness sharing (Horn, Goldberg & Deb, 1994) based on the coverage of patterns in the training set. The entire final population is used to form either the ensemble or some subset. In all the experiments in this chapter, the entire final population is used. The outputs of the individual networks are combined by either a simple average, a majority vote, or a winner-takes-all procedure.

The algorithm proceeds according to the following steps, shown in Box 1 (Liu et al., 2000).

Our implementation of the EENCL algorithm differs from Liu's in one respect: we use the amended derivative of the penalty term (Equation (11)). The learning rate is set to 0.1, and the NCL penalty term γ is set at 0.390625 in EENCL in order to provide a comparison to the lambda

Equation (12).

$$\lambda = \gamma\left(2\left(1-\frac{1}{M}\right)\right) = \gamma\left(2\frac{M-1}{M}\right). \quad (12)$$

Box 1.

1. An initial population (M) is trained for a small number of epochs (n_e) according to the NCL algorithm.
2. n_b parents are randomly selected from M, according to a uniform distribution.
3. n_b parents' weights are mutated, by Gaussian mutation N(0,1) to form n_b offspring.
4. n_b offspring are added to population and trained for e epochs, holding the weights of M fixed.
5. Fitness of all $M + n_b$ individuals evaluated using a fitness sharing scheme based on the coverage of training patterns.
6. Fittest M individuals from current population $M + n_b$ selected for next generation.
7. If the total number of generations reached, go to step 8, otherwise go to 2.
8. Combine the population into an ensemble.

value of 0.75 in Liu's original experiments (Liu et al., 2000), according to the equivalence provided in Equation (12).

The raw fitness of an individual is converted to an implicit shared fitness according to:

$$F_{share} = \sum_{n} \frac{1}{p_n}$$

where F_{share} is the shared fitness, n is the training pattern, and p_n is the number of individuals in the population that correctly classifies pattern n. Implicit fitness sharing is an extension to the original fitness sharing. Like the original, it also reduces the realized fitness at heavily populated regions of the search space (Darwen & Yao, 1997). However, it differs from the original, where a sharing radius must be defined to determine if two individuals will share their fitness.

Experiments with Local Search in EENCL

Liu, et al. (2000) showed that the combination of evolution with fitness-sharing and NCL could produce competitive results in comparison to a number of other classification techniques. It is less clear to what extent these learning mechanisms are responsible for this performance. We sought to establish whether NCL was a necessary component of the EENCL algorithm or if similar results could be obtained by means of an alternative (and less complex) local search. These results and analysis are also presented in Duell, et al. (2006a).

We applied the EENCL algorithm to four datasets—Australian Credit Card, Wisconsin Breast Cancer, Pima Indian Diabetes, and Heart Disease—over 30 independent runs. The number of instances, input attributes, classes, and class distributions of the datasets are presented in Table 1. These datasets are all available by anonymous ftp from the UCI Machine Learning Repository at ics.uci.edu (128.195.1.1) in /pub/machine-learning-databases. For each run, the datasets were shuffled and divided into three equal size sets to compose a training set, a validation set, and a testing set. The validation set is not used in these experiments. Each network learns the same training set. The results on the test sets are averaged to approximate the generalization error of the resulting ensembles. Three combination schemes are tested (simple average, majority vote, and winner-takes-all), and the ensemble is composed of the entire final population. In each case, the classification accuracy is shown in Table 2.

Table 1. Description of the datasets used in the experiments

Dataset	Instances	Input attributes	Classes (distributions)
Australian	690	14	2 (44.5%, 55.5%)
Breat Cancer	699	10	2 (65.5%, 34.5%)
Diabetes	768	8	2 (65.1%, 34.9%)
Heart Disease	920	13	4 (44.6%, 21.3%, 14.7%, 14.7%, 4.7%)

Table 2. Testing classification accuracies for the EENCL and EE-Backprop algorithms over four datasets, with three combination schemes. No statistically significant difference between the algorithms is found using a student t-test with a confidence of 1%.

Dataset	Algorithm	Average	Majority	Winner
Australian	EENCL	0.876	0.873	0.867
	EE-Backprop	0.873	0.870	0.868
Breast Cancer	EENCL	0.972	0.972	0.973
	EE-Backprop	0.972	0.972	0.972
Diabetes	EENCL	0.766	0.764	0.762
	EE-Backprop	0.763	0.764	0.757
Heart Disease	EENCL	0.789	0.785	0.772
	EE-Backprop	0.793	0.793	0.772

The EENCL algorithm was implemented as described by Liu, et al. (2000). The initial population is a set of randomly initialized MLPs with full connection and a single hidden layer of five nodes. Output nodes are encoded using a 1-of-c scheme, and all nodes are a sigmoidal logistic of the form:

$$f(x) = \frac{1}{1+exp(-x)}$$

where x is the input to the node and $f(x)$ is the output of the node. The node with the highest output is considered to be the classification of the network. The initial population M is set to 25, and the number of offspring per generation, n_b, is 2. The population is allowed to evolve for 200 generations. Both the initial population and offspring are trained for five epochs, n_e. The learning rate is set to 0.1, and the networks are trained using mean-square-error. The NCL penalty term γ is set at 0.390625. This is identical to the experimental setup described by Liu and Yao, arrived at in the original study after limited experimentation (Liu et al., 2000).

Here we also define a new algorithm, EE-backprop. This algorithm is identical to EENCL, except that NCL is no longer used as the local search technique in steps 1 and 4 of the algorithm. In EE-backprop, the penalty strength, λ is set to 0. As can be seen in Equation (8), the right-hand side now disappears, and we are left with a conventional mean-square-error. Hence, EENCL with a penalty of 0 is equivalent to using conventional backpropagation for the local search. In all other ways, EE-backprop is identical to EENCL and is also used with the same set of parameters.

Results and Discussion

The classification accuracies for EENCL and EE-backprop are shown in Table 2. Interestingly, no statistically significant difference between the techniques could be found using a student t-test with a 1% confidence interval. EENCL was found to be most effective with the winner-

takes-all combination scheme over the Australian and diabetes datasets in Liu's experiments (Liu et al., 2000). Unlike Liu, we do not observe that EENCL is more suited to the winner-takes-all combination scheme for all problems (only in the diabetes problem is the accuracy highest). This may be because winner-takes-all works well with specialized networks, and our ensembles have been trained on a smaller proportion of the datasets than Liu's and, therefore, had less opportunity to specialize.

EE-backprop is a simpler algorithm; each network learns independently and does not require the setting of a penalty strength parameter. However, it is just as effective as EENCL in terms of classification accuracy, so it is difficult to justify the added complexity of NCL. NCL alone is able to significantly improve on backpropagation and many other algorithms (Liu & Yao, 1999), but it appears that when used in conjunction with fitness sharing in an evolved ensemble, NCL's effectiveness is no longer apparent.

One method to analyze diversity in the final ensemble is to compare their correct response sets (Liu & Yao, 1999). Here we define the correct response set of a network i, Ω_i, as the set of examples it correctly classifies. We also define the *joint* correct response set between networks i and j, $\Omega_{i,j}$, as the set of examples that both networks classify correctly. Table 3 shows the mean individual correct response sets, Ω_i, and the mean joint correct response sets for all networks in the final population, $\Omega_{\forall i}$, over 30 runs for both EENCL and EE-backprop. Liu found that NCL produced significantly lower joint correct response sets to independent training with backpropagation (Liu & Yao, 1999). Our results show that this is not the case for EENCL and EE-backprop, where no significant difference could be found in $\Omega_{\forall i}$ except for the heart disease set, where $\Omega_{\forall i}$ is significantly *lower* for EE-backprop.

We also measured the average pairwise correlation between the networks in each ensemble. For the Australian credit card and breast cancer datasets, EENCL reduced correlation to a greater degree than EE-backprop, as expected. This was expected because EENCL explicitly seeks to minimize correlation, whereas EE-backprop seeks only to minimize error during local search. The other datasets, however, provided counterintuitive results, with EENCL creating more highly correlated ensembles. One possible explanation for these unexpected results is that they are a consequence of using two ways of encouraging diversity. EENCL can only effectively decorrelate the networks if the offspring that are trained are fit enough to survive; otherwise, the offspring will be discarded. In EENCL, fitness is awarded according to accuracy and also coverage of training patterns, which is not necessarily the same

Table 3. Correct response sets for EENCL and EE-Backprop across four datasets. Ω_i is the mean individual correct response set over 30 runs. $\Omega_{\forall i}$ is the mean joint correct response. σ is the standard deviation. No significant difference was found using student t-tests with a 1% confidence interval, except for $\Omega_{\forall i}$ for the heart disease problem, where EE-Backprop is significantly lower.

Dataset	Algorithm	Ω_i	σ	$\Omega_{\forall i}$	σ
Australian	EENCL	195.16	5.90	138.80	15.52
	EE-Backprop	195.11	5.77	139.20	11.16
Breast Cancer	EENCL	225.98	1.40	219.07	3.69
	EE-Backprop	225.98	1.59	218.10	4.77
Diabetes	EENCL	183.07	6.91	86.33	14.49
	EE-Backprop	182.32	8.31	81.57	25.12
Heart Disease	EENCL	66.75	11.01	29.67	3.75
	EE-Backprop	66.73	4.40	26.10	4.36

as correlation of outputs. NCL warps the mean-square-error landscape and then descends this new landscape (Brown, 2003). Fitness sharing, however, operates on a different landscape—a warping of the classification accuracy landscape according to the coverage of training patterns among the ensemble.

Our analysis shows that NCL is not integral to the success of the EENCL algorithm for these datasets. We have demonstrated that EE-backprop produces comparable classification accuracies. Likewise, we find that both techniques produce similar joint response sets, showing that EENCL is no more effective in producing specialization within the ensemble. We obtained surprising results that show that on some problems, EE-backprop was able to produce lower correlation among the ensemble than EENCL, but this did not necessarily translate into improved classification accuracy. We hypothesize that the explanation for how a method that explicitly seeks to reduce correlation such as EENCL can produce higher correlated networks than EE-backprop, which only implicitly reduces correlation, is to be found in the different and not necessarily complementary representations of diversity in EENCL. EENCL also requires the setting of a penalty strength parameter, which does not significantly improve performance over EE-backprop. We hypothesised that better results could be achieved if both local search and global evolution had complementary implementations of diversity, (e.g., both based on the coverage of training patterns or on correlation of outputs).

INCL Algorithm

Algorithm Description

Here we introduce a new algorithm (Duell et al., 2006b) based on EENCL without implicit fitness sharing. Instead, it follows the island model (Back, Fogel & Michalewicz, 1997). According to this model, the population is divided into subpopulations (islands), which have periods of isolated evolution. In the end of these periods, communication between neighboring subpopulations can occur through migration of individuals. Such parallel evolutionary algorithms have been used to produce more efficient evolution and also to maintain a diverse set of solutions in numerous studies (Cantu-Paz, 1997). We term the new algorithm based on the island model Island model with negative correlation learning (INCL).

INCL differs from EENCL in the following ways. In our implementation, the overall population is divided into equal-sized subpopulations. Offspring are trained using NCL to decorrelate their outputs from the other members of the same subpopulation. No migration takes place between subpopulations. Fitness is evaluated on raw classification accuracies, with no implicit fitness sharing. At the end of the evolutionary process, the subpopulations are merged into a single ensemble.

Experiments with Speciation Techniques in Evolutionary Ensembles

Here we examine whether it is possible to improve on the performance of EENCL by replacing fitness sharing with a subpopulation model (INCL) as an alternative method for encouraging speciation, to test whether fitness sharing is integral to EENCL's performance.

We applied the EENCL and INCL algorithms to the same four datasets described in Table 1 (Australian Credit Card, Wisconsin Breast Cancer, Pima Indian Diabetes, and Heart Disease) over 30 independent runs. For each run, the datasets were shuffled and divided into three equal-sized sets to compose a training set, a validation set, and a testing set. The validation set is not used in these experiments. Each network learns the same training set. The results on the test sets are averaged to approximate the generalization error of the resulting ensembles. Three combination

schemes are tested: simple average, majority vote, and winner-takes-all. In each case, the classification accuracy is shown in Table 4.

In the following experiments, the experimental setup for EENCL is as previously shown. The parameters for INCL are identical to EENCL except that the overall population of 25 is now arbitrarily divided into five subpopulations, each with five individuals. In this implementation, no migration occurs between the subpopulations. It is also worth noting that no attempt has been made to tune the parameters of INCL in order to achieve optimal performance. These results and analysis were previously presented in Duell et al., 2006b).

Results and Discussion

Classification accuracies for EENCL and INCL can be found in Table 4. Ensemble outputs are obtained using either simple-average (AVG), majority vote (MAJ), or winner-takes-all (WTA) combination scheme. Using a student t-test with a 1% confidence interval, INCL has significantly better performance for AVG and MAJ combinations on the breast cancer and heart disease sets, and WTA on the Australian credit card datasets (Table 5). For all other comparisons, no significant difference was found, although in each case, INCL had a higher accuracy, and in no comparison was EENCL significantly better than INCL.

By replacing fitness sharing in EENCL with an alternative diversity creation technique during global evolution, classification accuracy can be significantly improved for some problems without significant reduction in others. Our earlier work (Duell et al., 2006a) showed that fitness sharing negates the effects of NCL in EENCL on the problems we tested and was clearly the most sig-

Table 4. Classification accuracies for EENCL and INCL using simple-average (AVG), majority vote (MAJ) and winner-takes-all (WTA) combination schemes

Dataset	Algorithm	AVG	MAJ	WTA
Australian Credit	EENCL	0.876	0.873	0.867
	INCL	0.878	0.878	0.876
Breast Cancer	EENCL	0.972	0.972	0.973
	INCL	0.976	0.975	0.975
Diabetes	EENCL	0.766	0.764	0.762
	INCL	0.767	0.767	0.763
Heart Disease	EENCL	0.789	0.785	0.774
	INCL	0.808	0.810	0.789

Table 5. Probability that classification results are from the same distribution using student t-test

Dataset	Combination	p
Australian Credit	AVG	0.5155
	MAJ	0.1007
	WTA	0.0012
Breast Cancer	AVG	0.0001
	MAJ	0.0000
	WTA	0.0131
Diabetes	AVG	0.7372
	MAJ	0.1451
	WTA	0.7076
Heart Disease	AVG	0.0032
	MAJ	0.0002
	WTA	0.1065

nificant aspect of EENCL. This work suggests that the algorithm can be further improved if fitness sharing is replaced by a more suitable speciation technique, such as the island model in INCL.

Table 6 gives the average pairwise correlation between the networks in the ensembles, averaged over the output nodes and 30 runs. The results for the breast cancer and diabetes datasets are significantly different, based on a student t-test with a 1% confidence interval (Table 7). For the breast cancer dataset, a lower correlation was found with the INCL ensemble over the EENCL, but for the diabetes dataset, the opposite is true. This is not surprising since both EENCL and INCL attempt to decorrelate the outputs of the networks. However, INCL only decorrelates within a subpopulation, and so there is no guarantee that two networks from different subpopulations will be decorrelated. Further analysis would be interesting to see how correlation differs as the number of subpopulations changes, and also how the correlation in a particular subpopulation is affected. It is expected that smaller subpopulations will lead to a lower subpopulation correlation, but that this may lead to a higher population correlation, since each network is decorrelated from fewer other networks.

Correlation among the final population is significant, since the concept that decorrelated (and accurate) networks will reduce generalization error is the theoretical underpinning for NCL. NCL attempts to manage the trade-off between reducing the bias and covariance terms of the bias-variance-covariance trade-off (Ueda & Nakano, 1996). Our results show that for different problems, EENCL and INCL have differing success in reducing covariance, but that overall, generalization error is reduced most by INCL. Consequently, it appears that INCL is better suited to successfully exploit the trade-off between reducing covariance while also maintaining a low bias. In order to analyze diversity in the final ensemble, we will compare their correct response sets (Liu & Yao, 1999), as in the previous Results and Discussion section. Table 8 shows the mean individual correct response sets, Ω_i, and the mean joint correct response sets for all networks in the final population, $\Omega_{\forall i}$, over 30 runs for both EENCL and INCL. Interestingly, while our previous work (Duell et al., 2006a) found no significant difference between the size of correct response sets when changing the local

Table 6. Pairwise correlations averaged over each network and output node

Dataset	Algorithm	AVG	STDEV
Australian Credit	EENCL	0.917	0.021
	INCL	0.929	0.008
Breast Cancer	EENCL	0.992	0.003
	INCL	0.984	0.003
Diabetes	EENCL	0.842	0.042
	INCL	0.883	0.016
Heart Disease	EENCL	0.785	0.060
	INCL	0.773	0.030

Table 7. Probability that correlation results are from the same distribution using student t-test

Dataset	p
Australian Credit Card	0.0001
Breast Cancer	0.0000
Diabetes	0.0000
Heart Disease	0.1828

Table 8. Average individual ($\bar{\Omega}_i$) and joint ($\bar{\Omega}_{\forall i}$) correct response sets

Dataset	Algorithm	$\bar{\Omega}_{\forall i}$	$\bar{\Omega}_i$
Australian Credit	EENCL	138.800	195.157
	INCL	160.600	197.365
Breast Cancer	EENCL	219.067	225.980
	INCL	219.733	226.548
Diabetes	EENCL	86.333	183.069
	INCL	144.067	192.540
Heart Disease	EENCL	29.667	66.747
	INCL	37.767	68.463

Table 9. Probability that correct set results are from the same distribution using student t-test

Dataset	Type	p
Australian Credit	$\bar{\Omega}_i$	0.0000
	$\bar{\Omega}_{\forall i}$	0.0000
Breast Cancer	$\bar{\Omega}_i$	0.0000
	$\bar{\Omega}_{\forall i}$	0.3827
Diabetes	$\bar{\Omega}_i$	0.0000
	$\bar{\Omega}_{\forall i}$	0.0000
Heart Disease	$\bar{\Omega}_i$	0.0000
	$\bar{\Omega}_{\forall i}$	0.0000

search in EENCL, here the results are all found to be significantly different, except the joint set for the breast cancer dataset, using a student t-test with a 1% confidence interval (Table 9).

The joint sets for INCL are higher for all the tested datasets, indicating that by this measure, EENCL produces more diverse ensembles. This is as expected, since implicit fitness sharing rewards diversity based on the overlap of correct response sets. This does not, however, necessarily translate into a higher accuracy, as seen in Table 4. Therefore, in the presented experiments, EENCL is enforcing a larger degree of diversity than is desirable. INCL, on the other hand, does not enforce diversity (during evolution) but merely provides a framework where diversity can prosper. These results provide evidence that the latter approach may be preferable, especially when it is unclear how well a particular measure of diversity is suited to a problem.

Our analysis shows that implicit fitness sharing may not be the best way to encourage diverse and accurate evolved ensembles. We find that more accurate ensembles can be produced by providing the conditions for diversity to survive at the global evolutionary level through subpopulations rather than attempting to enforce a particular definition of diversity within the fitness evaluation. Our results show that in terms of correlation, fitness sharing does not produce more diverse ensembles than the island model in INCL across the datasets we tested. In terms of joint correct sets, however, EENCL does produce significantly more diverse ensembles on three out of four datasets, as this is the type of diversity that is being encouraged. That this does not translate into improved performance over INCL is evidence that this type of diversity is given too much weight in the EENCL algorithm. We suggest that better results can be obtained by methods that allow diversity to evolve, since they do not attempt to impose a potentially unsuitable predefined concept of "useful" diversity to an arbitrary problem.

Comparison of INCL and EENCL to Alternative Ensemble Techniques

Here we compare the classification performance of both EENCL and INCL to other popular ensemble learning techniques. The comparison is made to Bagging (Breiman, 1996), NCL (Liu, 1998), and an ensemble where each network is trained independently on the dataset from a different initialization. The purpose of this work is to gain a better understanding of the EENCL and INCL algorithms.

We applied the algorithms to the same four datasets as in the previous section (Australian Credit Card, Pima Indian Diabetes, Heart Disease, and Wisconsin Breast Cancer) over 30 independent runs. Each set was again equally divided into training, validation, and testing sets. A validation set was used to determine a suitable number of epochs/generations for the each algorithm on each problem. Each network learns the same training sets. The results of all 30 runs on the test set are averaged to approximate the generalization error of the resulting ensembles.

The EENCL and INCL algorithms were implemented as described previously. This time, however, the number of generations is determined from limited experimentation with a validation set. It is also worth noting that no attempt has been made to tune the parameters of INCL in order to achieve optimal performance except for the number of generations to run the algorithm.

In Bagging (Breiman, 1996), each network is presented with a different training set, sampled with replacement from the original set. In NCL (Liu, 1998), the networks are trained as in the local search of EENCL, except that gamma is now set at 0.390625, again to replicate a lambda value of 0.75. For the independently trained ensemble, each network learns from different initializations on the same data. No other diversity creation techniques are employed. In each case, the number of epochs is determined by the performance on the validation set as before.

Error Rates

Tables 10, 11, 12, and 13 show the classification accuracy of each tested algorithm on the Australian credit card, breast cancer, diabetes, and heart disease datasets, respectively, as well as the number of generations or epochs each ran for. The probabilities that the difference between the algorithms are insignificant using a student T-test are given in Tables 14, 15, 16, and 17, again for the Australian credit card, breast cancer, diabetes, and heart disease datasets, respectively. In the following discussion, a probability of less than 1% is taken to indicate that a significant difference exists between the results.

For the Australian credit card dataset, INCL, NCL, and bagging are all significantly better than independent training. Although no significant difference could be found between INCL, NCL,

Table 10. Classification accuracies on Australian credit card dataset for ensembles trained by EENCL, INCL, bagging, independent training, and NCL

Algorithm	Generations / Epochs	Train	Test
EENCL	50	0.888	0.862
INCL	60	0.878	0.866
Bagging	40	0.881	0.866
Ind.	25	0.882	0.860
NCL	30	0.874	0.863

Table 11. Classification accuracies on breast cancer dataset for ensembles trained by EENCL, INCL, bagging, independent training, and NCL

Algorithm	Generations / Epochs	Train	Test
EENCL	30	0.989	0.976
INCL	50	0.983	0.982
Bagging	60	0.976	0.983
Ind.	35	0.975	0.983
NCL	35	0.976	0.982

Table 12. Classification accuracies on diabetes dataset for ensembles trained by EENCL, INCL, bagging, independent training, and NCL

Algorithm	Generations / Epochs	Train	Test
EENCL	130	0.800	0.765
INCL	180	0.789	0.764
Bagging	105	0.781	0.765
Ind.	85	0.778	0.761
NCL	70	0.778	0.771

Table 13. Classification accuracies on heart disease dataset for ensembles trained by EENCL, INCL, bagging, independent training, and NCL

Algorithm	Generations / Epochs	Train	Test
EENCL	220	0.874	0.767
INCL	285	0.849	0.804
Bagging	95	0.840	0.817
Ind.	60	0.833	0.826
NCL	55	0.837	0.817

Table 14. Probability that classification results are from the same distribution using student t-test: Australian credit card problem

Algorithm	EENCL	INCL	Bagging	Ind.	NCL
EENCL	1.000	0.118	0.125	0.621	0.556
INCL	0.118	1.000	0.912	0.000	0.036
Bagging	0.125	0.912	1.000	0.000	0.032
Ind.	0.621	0.000	0.000	1.000	0.006
NCL	0.556	0.036	0.032	0.006	1.000

Table 15. Probability that classification results are from the same distribution using student t-test: breast cancer problem

Algorithm	EENCL	INCL	Bagging	Ind.	NCL
EENCL	1.000	0.000	0.000	0.000	0.000
INCL	0.000	1.000	0.052	0.052	0.753
Bagging	0.000	0.052	1.000	1.000	0.039
Ind.	0.000	1.000	1.000	1.000	1.000
NCL	0.000	0.753	0.039	0.039	1.000

Table 16. Probability that classification results are from the same distribution using student t-test: diabetes problem

Algorithm	EENCL	INCL	Bagging	Ind.	NCL
EENCL	1.000	0.680	0.952	0.069	0.020
INCL	0.680	1.000	0.492	0.027	0.001
Bagging	0.952	0.492	1.000	0.002	0.002
Ind.	0.069	0.027	0.002	1.000	0.000
NCL	0.020	0.001	0.002	0.000	1.000

Table 17. Probability that classification results are from the same distribution using student t-test: heart disease problem

Algorithm	EENCL	INCL	Bagging	Ind.	NCL
EENCL	1.000	0.000	0.000	0.000	0.000
INCL	0.000	1.000	0.008	0.000	0.002
Bagging	0.000	0.008	1.000	0.004	0.816
Ind.	0.000	0.000	0.004	1.000	0.001
NCL	0.000	0.002	0.816	0.001	1.000

bagging, and EENCL, it is important to note that in this case, EENCL's classification accuracy is not significantly better than independent training. All of the other algorithms have a significantly higher classification accuracy than EENCL on the diabetes dataset. NCL has a significantly higher classification accuracy than INCL, bagging, and independent training on the diabetes dataset; and bagging is also significantly better than independent training. Here, neither EENCL nor INCL are significantly better than independent training, whereas both NCL and bagging are. For the heart disease problem, all the algorithms have a significantly higher classification rate than EENCL. NCL, bagging, and independent training are significantly better than INCL, and independent training is significantly better than all the other algorithms, perhaps indicating that this problem contains too few examples to effectively produce a diverse ensemble by any of these methods.

For the problems tested here, EENCL has a significantly lower classification rate than all the other algorithms on half of the datasets (breast cancer and heart disease) and is not significantly better than any of the other algorithms on the other two problems (Australian credit card and diabetes). So EENCL's strength lies not in its increase in classification performance, but in the use of an evolutionary approach that is easily adaptable to evolve more than just the weights of the networks. However, INCL also utilizes an evolutionary approach but has a comparable performance to the other algorithms tested except for NCL on the diabetes problem (which also outperforms bagging here) and on the heart disease problem. Even on the heart disease problem, an improvement on EENCL's performance is made (although not a significant one). It therefore appears that INCL is able to match or improve upon the performance of EENCL on the tested problems without losing the evolutionary approach and redresses some of the shortfall in performance of EENCL compared to NCL, bagging, and independent training.

Our previous work suggested that one possible explanation for why EENCL produced less accurate ensembles was that fitness sharing promotes a different kind of diversity than NCL, and that the two are not necessarily complementary (Duell et al., 2006b). Our results showed that EENCL was more effective than INCL at reducing the overlap in the correct sets of the ensembles (encouraged by fitness sharing), but that this did not necessarily translate into a reduced correlation of outputs (encouraged by NCL). Our hypothesis was that by attempting to manage two kinds of diversity, EENCL was less effective at reducing covariance in the ensemble. The next section analyzes EENCL and INCL in terms of the bias-variance-covariance decomposition (Ueda & Nakano, 1996) in order to test this hypothesis.

Bias, Variance, and Covariance Analysis

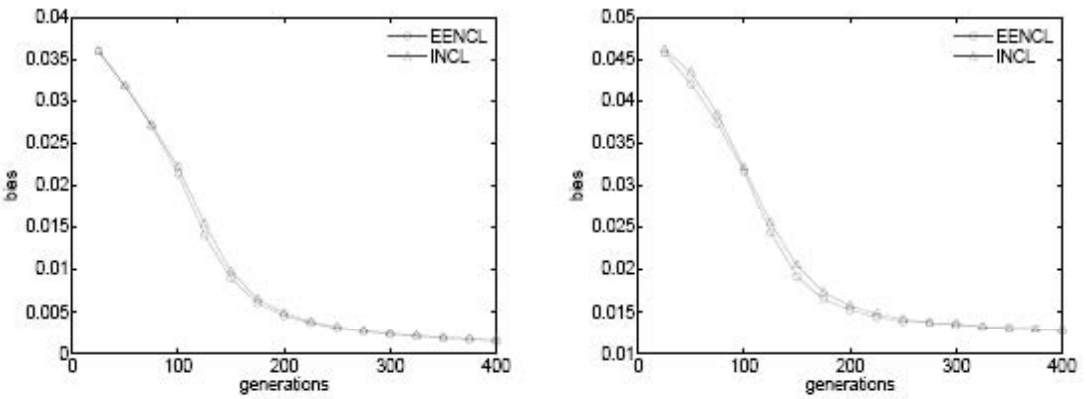
In order to understand why EENCL performed poorly in comparison to the other ensemble techniques, we conducted further experiments to determine the bias, variance, and covariance decomposition (Ueda & Nakano, 1996) of the generalization error. This decomposition is well defined and understood for regression problems, but not for classification problems. Implicit fitness sharing as implemented in EENCL is, however, only applicable to classification problems, since it is dependent on coverage of training patterns and is therefore not easily translated into a regression context. Implicit fitness sharing is, however, a parameterless approximation to the classical fitness-sharing technique (Goldberg, 1989), requiring the setting of a sharing radius. Now an individual's fitness is reduced, depending upon the number of other individuals whose outputs are similar within the sharing radius for each pattern. This explicit fitness sharing will allow us to investigate EENCL in the better-understood regression context in order to explain its behavior on classification problems. INCL is applicable

to both classification and regression problems without alteration.

Here, we analyze EENCL and INCL on an artificial regression problem in order to estimate the bias, variance, and covariance of the algorithms. We examine each in the context of a zero-, small-, and large-noise situation. For ease of comparison, the experimental setup follows as closely as possible Liu's work on NCL (Liu, 1998). The function to be learned is shown in Equation (13), where x_1, x_2, x_3, x_4, and x_5 are inputs in the range of [0,1]. The target value *f(x)* lies in *Equation (13).*

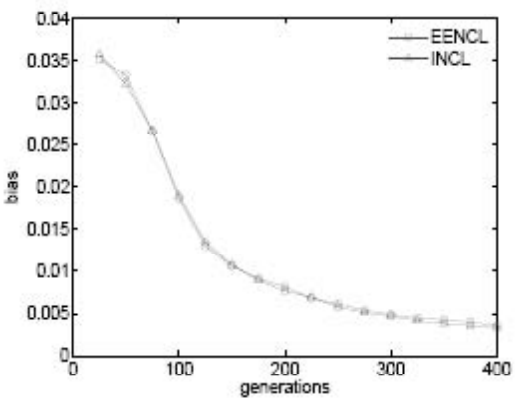
$$f(x) = \frac{1}{13}\Big(10\sin(\pi x_1 x_2) + 20\left(x_3 - \frac{1}{2}\right)^2 + 10x_4 + 5x_5\Big) - 1, \quad (13)$$

Figure 3. Bias estimation of EENCL and INCL on an artificial regression problem in the zero-, small-, and large-noise condition



(a) Zero Noise

(b) Small Noise

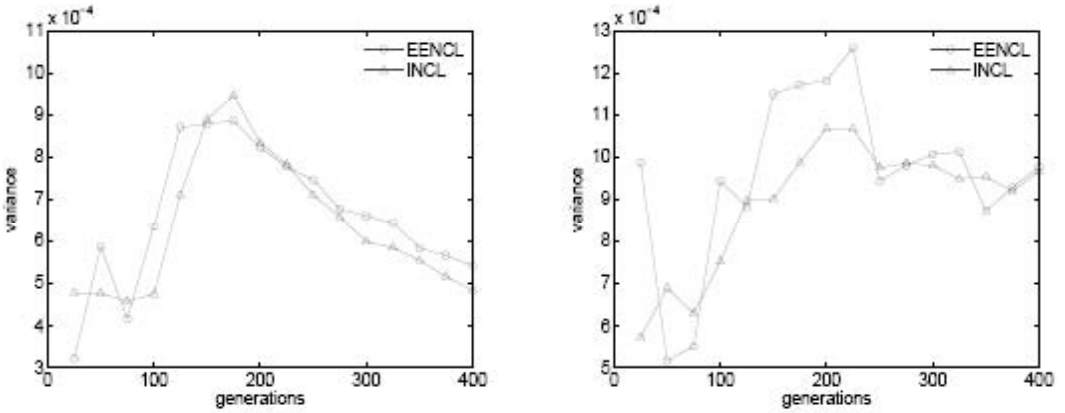


(c) Large Noise

the range [-1,1]. We created 25 training sets of 500 examples randomly, with the inputs sampled uniformly from the range [0,1]. In the zero-noise problem, the target outputs were not corrupted by noise, but in the small-noise and large-noise cases, Gaussian noise was added from $\sigma^2 = 0.1$ and $\sigma^2 = 0.2$, respectively. A testing set of 1,024 examples was created in the same manner, except the target outputs were not corrupted by noise.

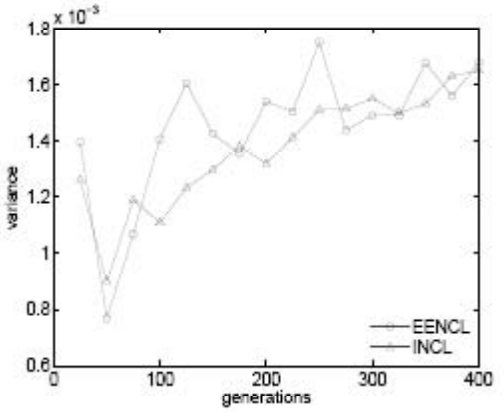
Network structures are as described in the previous experiments, except that the output layer was changed to linear nodes rather than logistic. For EENCL, each ensemble consisted of nine individual networks, and for INCL, there were three subpopulations, each with three individuals. A λ value of 0.75 for both algorithms is again used, giving γ values of 0.42188 for EENCL and 0.56250 for INCL. For both EENCL and INCL, each ensemble was trained on each of the 25 training sets from the same randomly initialized weights for 400 generations. For EENCL, a

Figure 4. Variance estimation of EENCL and INCL on an artificial regression problem in the zero-, small-, and large-noise conditions



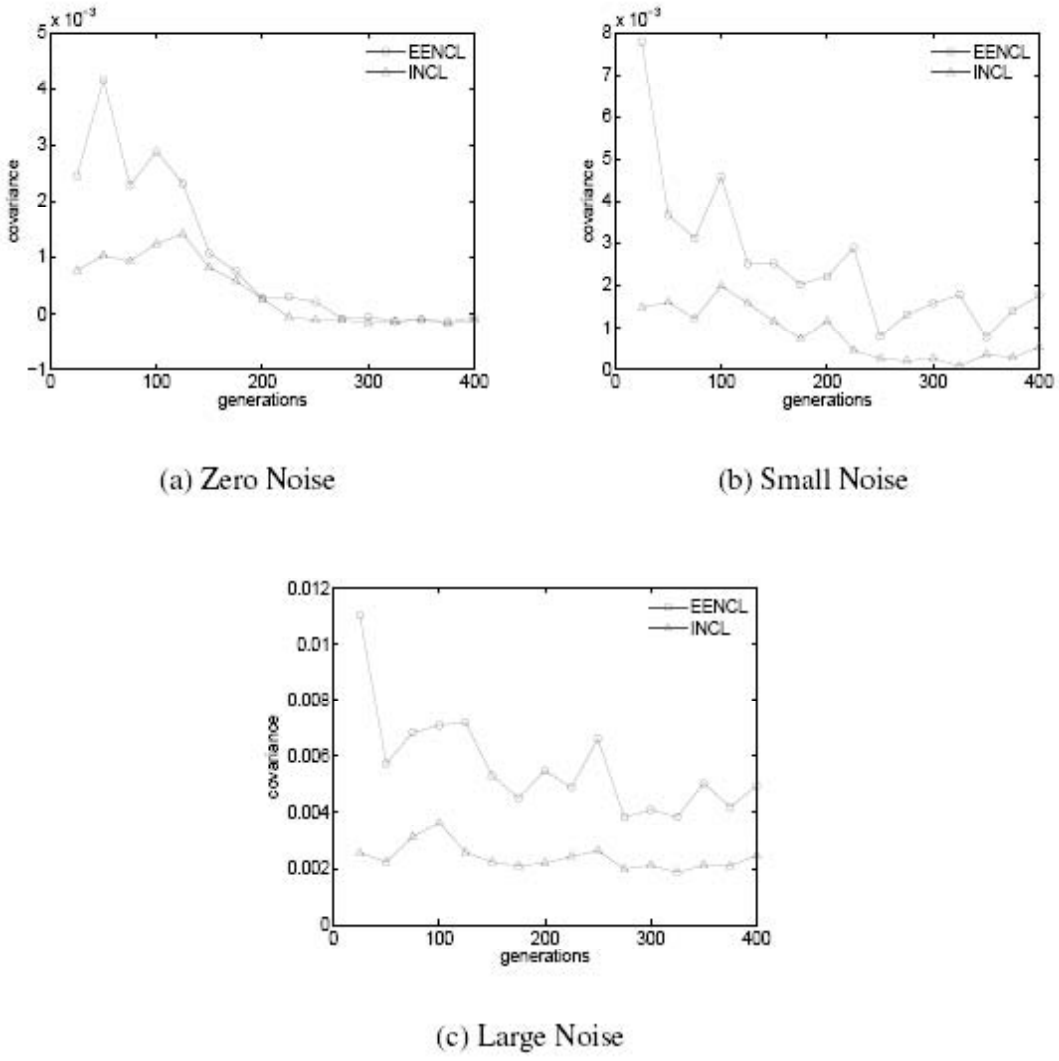
(a) Zero Noise

(b) Small Noise



(c) Large Noise

Figure 5. Covariance estimation of EENCL and INCL on an artificial regression problem in the zero-, small-, and large-noise conditions

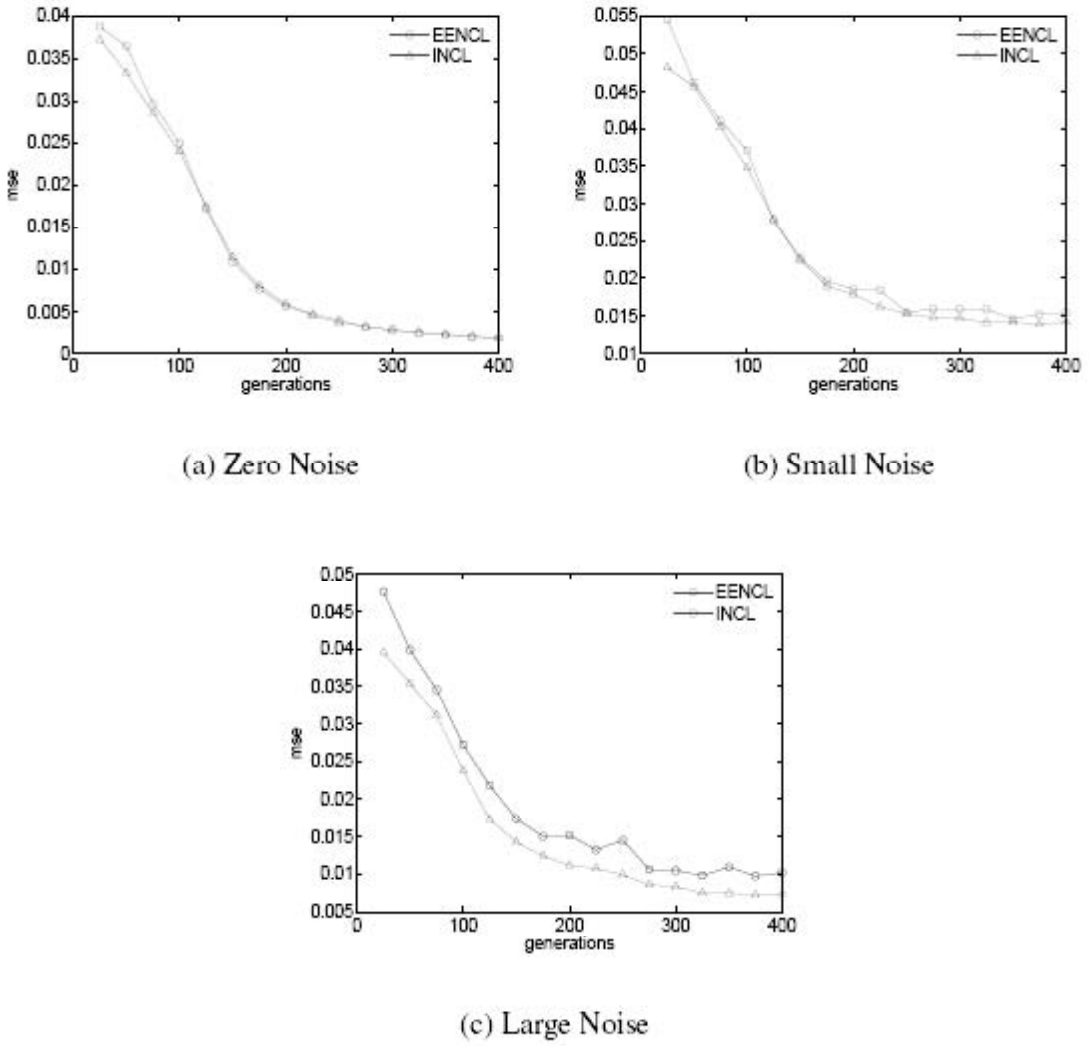


sharing radius of 0.2 was selected after limited experimentation.

Figures 3, 4, 5, and 6 show the bias, variance, covariance, and generalization error of the ensembles during the simulation. Each figure is divided into a zero-, small-, and large-noise condition. For both algorithms, bias is reduced at a similar rate during learning in the noise and noise-free conditions. For both algorithms, in the zero-noise condition, variance rises at first but then declines. In the small-noise condition, variance rises and stabilizes, and in the large-noise variance, it rises.

Both algorithms exhibit similar trends, but at most points in the learning process, INCL displays a slightly lower variance. Covariance is more interesting, falling at first before stabilizing. However, as the noise in the dataset increases, INCL becomes more efficient at reducing covariance (and from the previous Figures 3 and 4, bias and variance remain comparable). This lends support to our hypothesis that implicit fitness sharing in

Figure 6. Generalization error (MSE) estimation of EENCL and INCL on an artificial regression problem in the zero-, small-, and large-noise conditions



EENCL is preventing NCL from effectively reducing the covariance of the ensemble, and shows that INCL is able to improve on the performance of EENCL because it more effectively reduces covariance.

Discussion

The purpose of these experiments was to gain a better understanding of the EENCL and INCL algorithms. Our empirical studies show that EENCL is not as effective as some other ensemble learning algorithms in some cases on the problems we tested. Our hypothesis for these results and those reported in our paper (Duell et al., 2006b) was that fitness sharing was not appropriate for use with NCL, since they both encourage diversity but measure their success in differing ways. However, such evolutionary approaches to ensemble design remain desirable since they have the potential to adapt more than just network weights. As such, an alternative method

for speciation during evolution is necessary. One approach is to divide the population according to an island model, which is the approach followed in INCL. We have shown that INCL improves on the classification performance of EENCL, making it comparable with the other algorithms on all but one of the tested datasets. Our analysis of the bias-variance-covariance decomposition supports our earlier hypothesis: replacing fitness sharing with an alternative speciation method led to lower covariance in the ensemble, maintaining comparable bias and variance values. As such, INCL improves on EENCL because it better exploits the bias-variance-covariance trade-off.

CONCLUSION

The concept of diversity for creating effective ensembles remains an interesting field of study. This chapter examines one mechanism for producing diverse ensembles: negative correlation learning (NCL). We have motivated the desire for diversity in ensembles and reviewed how NCL is intended to exploit the bias, variance, and covariance decomposition. Several researchers have begun to include NCL in a wider evolutionary framework, giving rise to the possibility of determining not just the weights of neural networks, but network architectures as well. In such a framework, it is important that we maintain diversity during the selection process. The work presented here shows that the choice of such a diversity creation mechanism to work alongside NCL is an important one. Implicit fitness sharing tends to dominate the effect of NCL when used in combination (as in EENCL), and as such is not a suitable choice. We suggest that an island population structure (as in INCL) is a better alternative since it does not enforce a predefined concept of "good" diversity onto an arbitrary problem, but simply provides a structure in which diversity is able to survive. In a comparison to some other nonevolutionary ensemble methods, we find that EENCL's performance is significantly worse on some problems and never significantly better. INCL, however, improves on the performance of EENCL and proves comparable to the other methods tested. Therefore, INCL provides the advantages of an evolutionary approach (i.e., the ability to evolve more than just weights) without the cost to classification performance exhibited by EENCL. Our analysis of the bias, variance, and covariance for both INCL and EENCL on an artificial problem suggests that the lower generalization error is the result of a more effective reduction of Covariance, the better the diversity creation mechanisms of INCL are at producing desired diversity.

REFERENCES

Abbass, H.A. (2003). Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. *Proceedings of the IEEE 2003 Conference on Evolutionary Computation*, *3*, 2074–2080.

Back, T., Fogel, D., & Michalewicz, Z. (1997). *Handbook on evolutionary computation*. Oxford University Press.

Bishop, C.M. (1996). *Neural networks for pattern recognition*. Oxford University Press.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.

Brown, G. (2001). On the effectiveness of negative correlation learning. *Proceedings of the First UK Workshop on Computational Intelligence*, 57–62.

Brown, G. (2003). *Diversity in neural network ensembles* [doctoral thesis]. University of Birmingham, School of Computer Science.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion, 6*(1), 5–20.

Cai, Y., Sun, X., & Jia, P. (2004). Negative correlation learning approach for t-s fuzzy models. *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, 3*, 2254–2259.

Cantu-Paz, E. (1997). *A survey of parallel genetic algorithms* [Technical Report 91003]. Urbana-Champaign, IL: University of Illinois, Department of Computer Science and Illinois Genetic Algorithms Laboratory.

Chan, Z.S.H., & Kasabov, N. (2005). A preliminary study on negative correlation learning via correlation-corrected data (NCCD). *Neural Processing Letters, 21*, 207–214.

Chandra, A., & Yao, X. (2004). Divace: Diverse and accurate ensemble learning algorithm. *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning (LNCS 3177)*, Exeter, UK, 619–625.

Chandra, A., & Yao, X. (2006). Ensemble learning using multi- objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms, 5*, 417–445.

Darwen, P.J., & Yao, X. (1997). Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation, 1*(2), 101–108.

Dietterich, T. (2000). Ensemble methods in machine learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, 1–15.

Duell, P., Fermin, I., & Yao, X. (2006a). Diversity creation in local search for the evolution of neural network ensembles. *Proceedings of the 14th European Symposium on Artificial Neural Networks (ESANN06).*

Duell, P., Fermin, I., & Yao, X. (2006b). Speciation techniques in evolved ensembles with negative correlation learning. *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, Vancouver, British Columbia, 3317–3321.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation, 4*(1), 1–58.

Goldberg, D.E. (1989). *Genetic algorithms in search optimization, and machine learning.* Reading, MA: Addison-Wesley.

Hansen, L.K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*(10), 993–1001.

Horn, J., Goldberg, D.E., & Deb, K. (1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation, 2*(1), 37–66.

Islam, M.M., Yao, X., & Murase, K. (2003). A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks, 14*(4), 820–834.

Kottathra, K., & Attikiouzel, Y. (1996). A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks. *Journal of Network and Computer Applications, 19,* 135–147.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems*, Vol. 7 (pp. 231–238). Cambridge, MA: MIT Press.

Kupinski, M.A., & Anastasio, M.A. (1999). Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging, 18*(8), 675–685.

Liu, Y. (1998). *Negative correlation learning and evolutionary neural network ensembles* [doctoral thesis]. Canberra, Australia: University College,

The University of New South Wales, Australian Defence Force Academy.

Liu, Y., & Yao, X. (1998). Negatively correlated neural networks for classification. *Proceedings of the Third International Symposium on Artificial Life and Robotics (AROBIII'98)*, Beppu, Japan, 2, 736–739.

Liu, Y., & Yao, X. (1999). Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, *29*(6), 716–725.

Liu, Y., Yao, X., & Higuchi, T. (2000). Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, *4*(4), 380–387.

McKay, R.I.B., & Abbass, H.A. (2001). Anti-correlation: A diversity promoting mechanisms in ensemble learning. *The Australian Journal of Intelligent Information Processing Systems (AJIIPS)*, *7*(3/4), 139–149.

Opitz, D.W., & Shavlik, J.W. (1996). Generating accurate and diverse members of a neural-network ensemble. In D.S. Touretzky, M.C. Mozer, & M.E. Hasselmo (Eds.), *Advances in neural information processing systems*, Vol. 8 (pp. 535–541). Cambridge, MA: MIT Press.

Perrone, M., & Cooper, L. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R.J. Mammone (Ed.), *Artificial neural networks for speech and vision* (pp. 126–142). London: Chapman & Hall.

Rosen, B.E. (1996). Ensemble learning using decorrelated neural networks. *Connection Science—Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, *8*(3/4), 373–384.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representation by error propagation. In D.E. Rumelhart, & J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1 (pp. 318–362). Cambridge, MA: MIT Press.

Schapire, R.E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Sharkey, A. (1999). Combining artificial neural nets: Ensemble and modular multi-net systems. In *Multi-net systems* (pp. 1–30). Springer-Verlag.

Turner, K., & Gosh, J. (1996a). Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, *29*(2), 341–348.

Turner, K., & Gosh, J. (1996b). Error correlation and error reduction in ensemble classifiers. *Connection Science*, *8*(3-4), 385–403.

Ueda, N., & Nakano, R. (1996). Generalization error of ensemble estimators. *Proceedings of International Conference on Neural Networks*, 90–95.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), 1423–1447.

ENDNOTE

[1] This chapter is partially based on Duell, Fermin, and Yao (2006a, 2006b). This work is partially funded by EPSRC and Thales Research & Technology (UK) Ltd.

Chapter XVII
A Recurrent Probabilistic Neural Network for EMG Pattern Recognition

Toshio Tsuji
Hiroshima University, Japan

Nan Bu
Hiroshima University, Japan

Osamu Fukuda
National Institute of Advanced Industrial Science and Technology, Japan

ABSTRACT

In the field of pattern recognition, probabilistic neural networks (PNNs) have been proven as an important classifier. For pattern recognition of EMG signals, the characteristics usually used are: (1) amplitude, (2) frequency, and (3) space. However, significant temporal characteristic exists in the transient and non-stationary EMG signals, which cannot be considered by traditional PNNs. In this article, a recurrent PNN, called recurrent log-linearized Gaussian mixture network (R-LLGMN), is introduced for EMG pattern recognition, with the emphasis on utilizing temporal characteristics. The structure of R-LLGMN is based on the algorithm of a hidden Markov model (HMM), which is a routinely used technique for modeling stochastic time series. Since R-LLGMN inherits advantages from both HMM and neural computation, it is expected to have higher representation ability and show better performance when dealing with time series like EMG signals. Experimental results show that R-LLGMN can achieve high discriminant accuracy in EMG pattern recognition.

INTRODUCTION

Electromyographic (EMG) signals provide information about neuromuscular activities and have been recognized as efficient and promising resources for human-machine interface (HMI) used for the rehabilitation of people with mobility limitations and those with severe neuromuscular impairment. Typically, a pattern recognition process is applied to *translate* EMG signals into control commands for the HMIs, such as powered prostheses and functional electrical stimulation devices (Englehart et al., 2001; Fukuda et al., 2003; Hudgins et al., 1993; Lusted & Knapp, 1996). Generally speaking, a successful EMG pattern recognition technique relies on two principle elements: a pattern classifier with reliable discrimination accuracy and efficient representation of EMG feature characteristics.

Probabilistic neural networks (PNNs) developed in the field of pattern recognition make a decision according to the probability density distribution of patterns in the feature space (Specht, 1990; Tsuji et al., 1999). Since PNNs integrate statistical models into the neural networks' architecture as prior knowledge, outstanding performance has been reported. Recently, PNNs have become widely accepted as important classifiers and have been proven to be efficient, especially for complicated problems such as pattern recognition of bioelectric signals.

For EMG pattern recognition using PNNs, the feature characteristics usually used include: (1) amplitude, (2) frequency, and (3) spatial information from multiple channels of EMG signals. However, significant temporal characteristics exist in the transient and non-stationary EMG signals, which cannot be considered by the traditional PNNs based on static stochastic models, and, in some cases, temporal characteristics could be the only clues for reliable recognition.

This chapter introduces a recurrent PNN called recurrent log-linearized Gaussian mixture network (R-LLGMN) (Tsuji et al., 2003) into EMG pattern recognition, with emphasis on utilizing temporal characteristics. The structure of R-LLGMN is based on the hidden Markov model (HMM) algorithm, which is a routinely used technique for modeling stochastic time series. Since R-LLGMN inherits the advantages from both HMM and neural computation, it is expected to have higher representation ability and exhibit better classification performance when dealing with time series like EMG signals.

After a review of the literature, the structure and algorithm of R-LLGMN are explained. The proposed EMG pattern recognition method using R-LLGMN is then described, and experiments on filtered EMG and raw EMG signals are presented. Based on the experimental results, the possibility of applying the proposed method to practical human interface control is discussed. The final section offers some concluding remarks.

BACKGROUND

Up to now, many techniques have been developed for EMG pattern recognition using statistical methods and neural networks (NNs). Kang et al. (1995) proposed a maximum likelihood method (MLM) based on Mahalanobis distances between input pattern and the prototypes, and the Bayes decision rule is applied in this method. A traditional linear discriminant analysis (LDA) classifier is used in an EMG classification scheme for multifunction myoelectric control (Englehart et al., 2001).

Due to NNs' learning capability of finding near-optimum functional relationships between the class memberships and the EMG patterns, several NN-based EMG pattern recognition methods have been presented. For example, Hiraiwa et al. (1989) used a multilayer perceptron (MLP) NN to perform pattern discrimination of five finger motions. Kelly et al. (1990) applied an MLP to classify four arm functions. Hudgins et al. (1993) devised a control system for powered

upper-limb prostheses using a set of time-domain features extracted from EMG signals and a simple MLP as a classifier. Also, similar studies have been developed using MLPs to classify EMG features, such as autoregressive (AR) parameters (Lamounier et al., 2002) and features of filtered EMG signals (Tsuji et al., 1993). However, several factors have hindered the extension of MLP classifiers for other applications, such as the choice of network structure, slow learning convergence, the need for a large amount of training data, and local minima.

To tackle these problems, numerous attempts have been made by the pattern recognition community to integrate statistical models, as prior knowledge, into the classifier's architecture, to take advantage of both statistical classification methods and neural computation. Consequently, probabilistic neural networks (PNNs) have been developed for pattern recognition (Specht, 1990; Zhang, 2000). In particular, Tsuji et al. (1999) proposed a feedforward PNN, a log-linearized Gaussian mixture network (LLGMN), which is based on the Gaussian mixture model (GMM) and a log-linear model. Although weights of the LLGMN correspond to a non-linear combination of the GMM parameters, such as mixture coefficients, mean vectors, and covariance matrices, constraints on the parameters in the statistical model are relieved in the LLGMN. Therefore, a simple backpropagation-like learning algorithm can be derived, and the parameters of LLGMN are trained according to a criterion of maximum likelihood (ML). The LLGMN has been successfully applied to EMG pattern recognition, where eight motions of the forearm have been classified using EMG signals measured by several pairs of electrodes (Fukuda et al., 2003). Also, the LLGMN has been further used to develop interface applications like prosthetic devices and EMG-based pointing devices (Fukuda et al., 1997, 1999; Fukuda et al., 2003).

However, since the GMM is a *static* stochastic model, it cannot make efficient use of temporal (time-varying) characteristics in EMG signals. Generally, pattern recognition using LLGMN is made under the assumption that feature patterns are stationary or change very slowly. EMG signals, in fact, are non-stationary and vary significantly in amplitude and frequency, even in the space domain. Due to the complicated nature of EMG signals, it is widely accepted that the temporal characteristic contains information important for pattern recognition (Englehart et al., 1999).

In order to cope with the time-varying characteristics of EMG signals, a pattern recognition method using an MLP classifier and a neural filter (NF) was applied (Tsuji et al., 2000). Continuous motions by the operators can be discriminated with sufficient accuracy even using the non-stationary time series of EMG signals. In addition to improving the classifiers, time-frequency representations of EMG signals have been adopted to gain a high level of discrimination accuracy (Englehart et al., 1999, 2001; Hussein & Granat, 2002). Although these methods can generate sufficient discrimination accuracy, there may be some criticism due to more complicated signal processing required or more intricate structure of classifiers. Also more parameters in the algorithm(s) of the signal processing and/or the classifier need to be determined by the user. Optimization of the whole pattern recognition method is almost impossible, and it is hard to gain a high performance of discrimination, especially in practical applications.

The present study focuses on the classifier aspect of EMG pattern recognition and introduces a recurrent PNN to improve discrimination accuracy when dealing with non-stationary EMG signals.

A RECURRENT PROBABILISTIC NEURAL NETWORK

The recurrent PNN, R-LLGMN (Tsuji et al., 2003), is based on the algorithm of continuous density hidden Markov model (CDHMM), which is a

combination of the GMM and the HMM (Rabiner, 1989). The probability density function (pdf) of input patterns is estimated using GMM; HMM is used simultaneously to model the time-varying characteristics in stochastic time series. In the R-LLGMN, recurrent connections are incorporated into the network structure to make efficient use of the time-varying characteristics of EMG signals. With the weight coefficients well trained using a learning scheme of the backpropagation through time (BPTT) algorithm, R-LLGMN can calculate posterior probabilities of the discriminating classes.

HMM-Based Dynamic Probabilistic Model

First, let us consider a dynamic probabilistic model, as shown in Figure 1. There are C classes in this model, and each class c ($c \in \{1,\cdots,C\}$) is composed of K_c states. Suppose that, for the given time series $\tilde{\mathbf{x}} = \mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(T)$ $(\mathbf{x}(t) \in \Re^d)$, at any time $\mathbf{x}(t)$ must occur from one state k of class c in the model. With this model, the posterior probability for class c, $P(c \mid \tilde{\mathbf{x}})$, is calculated as

$$P(c \mid \tilde{\mathbf{x}}) = \sum_{k=1}^{K_c} P(c,k \mid \tilde{\mathbf{x}}) = \sum_{k=1}^{K_c} \frac{\alpha_k^c(T)}{\sum_{c'=1}^{C} \sum_{k'=1}^{K_{c'}} \alpha_{k'}^{c'}(T)}. \quad (1)$$

Here, $\alpha_k^c(T)$ is the forward variable, which is defined as the probability for time series $(\mathbf{x}(1),\mathbf{x}(2),..., \mathbf{x}(T))$ to be generated from class c, and vector $\mathbf{x}(T)$ occurs from state k in class c. According to the forward algorithm (Rabiner, 1989), it can be derived as:

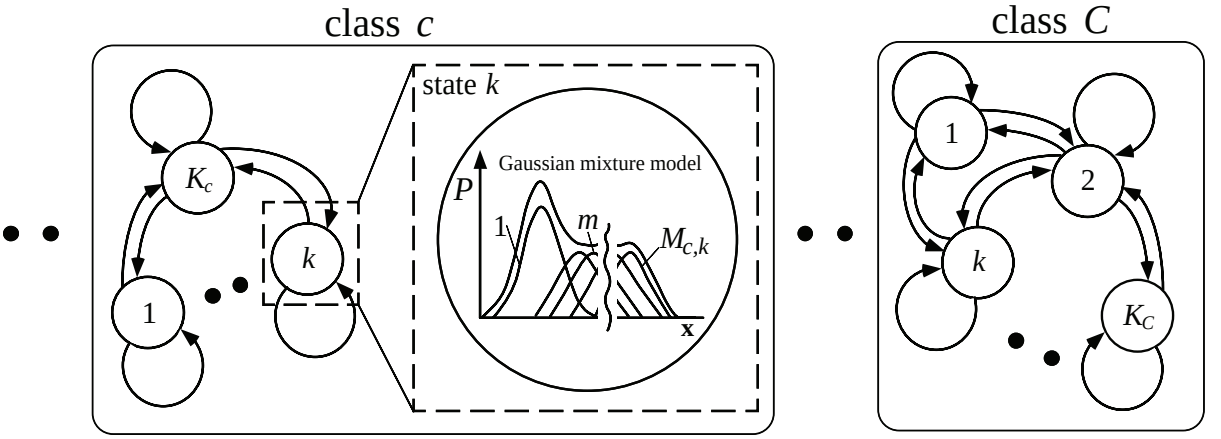
$$\alpha_k^c(1) = \pi_k^c b_k^c(\mathbf{x}(1)), \quad (2)$$

$$\alpha_k^c(t) = \sum_{k'=1}^{K_c} \alpha_{k'}^c(t-1)\, \gamma_{k',k}^c\, b_k^c(\mathbf{x}(t)) \quad (1 < t \le T), \quad (3)$$

where $\gamma_{k',k}^c$ is the probability of the state changing from k' to k in class c, and $b_k^c(\mathbf{x}(t))$ is defined as the posterior probability for state k in class c corresponding to $\mathbf{x}(t)$. Also, the prior probability π_k^c is equal to $P(c,k)|_{t=0}$.

In this model, the posterior probability $b_k^c(\mathbf{x}(t))$ is approximated by summing up $M_{c,k}$ components of a Gaussian mixture distribution, and $\gamma_{k',k}^c b_k^c(\mathbf{x}(t))$ on the right side of (3) is derived in the form shown in Box 1, where $r^{(c,k,m)}$, $\boldsymbol{\mu}^{(c,k,m)} = (\mu_1^{(c,k,m)}, \cdots, \mu_d^{(c,k,m)})^{\mathrm{T}}$, $\Sigma^{(c,k,m)} \in \Re^{d \times d}$, $s_{jl}^{(c,k,m)}$ and $x_j(t)$ stands for the mixing proportion, the mean vector, the covariance matrix of each component $\{c,k,m\}$, the element of the inverse of covariance matrix $\Sigma^{(c,m,k)-1}$, and the element of $\mathbf{x}(t)$.

Figure 1. HMM-based dynamic probabilistic model with C classes and Kc states in class c



The R-LLGMN is developed from the model defined above. For an input time series $\tilde{\mathbf{x}}$, the posterior probability for each class can be estimated with a well-trained R-LLGMN. The R-LLGMN network structure and learning algorithm are explained in the following.

Network Architecture

R-LLGMN is a five-layer recurrent NN with feedback connections between the fourth and the third layers, the structure of which is shown in Figure 2. First, the input vector series $\mathbf{x}(t) \in \Re^d$ (t=1,...,T) is preprocessed into the modified input series $\mathbf{X}(t) \in \Re^H$ as follows:

$$\mathbf{X}(t) = (1, \mathbf{x}(t)^{\mathrm{T}}, x_1(t)^2, x_1(t)x_2(t), \cdots, x_1(t)x_d(t), x_2(t)^2, x_2(t)x_3(t), \cdots, x_2(t)x_d(t), \cdots x_d(t)^2)^{\mathrm{T}}, \tag{5}$$

where the dimension H is determined as $H = 1 + d\,(d + 3)/2$. The vector $\mathbf{X}(t)$ acts as the input of the first layer, and the identity function is used to activate each unit. The output of the hth ($h = 1,\cdots,H$) unit in the first layer is defined as ${}^{(1)}O_h(t)$.

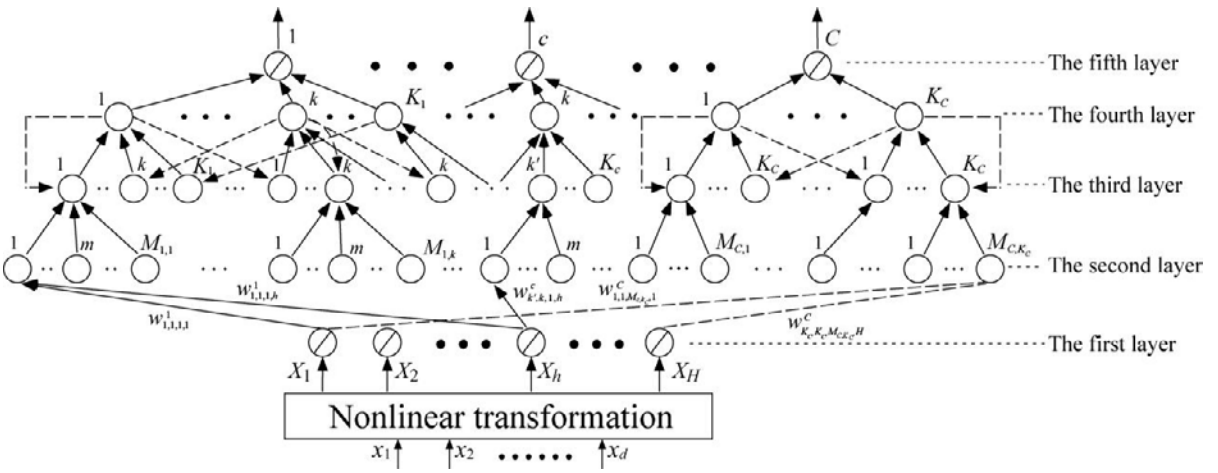
Unit $\{c,k,k',m\}$ ($c = 1,\cdots,C$; $k',k = 1,\cdots,K_c$; $m = 1,\cdots,M_{c,k}$) in the second layer receives the output of the first layer, weighted by the coefficient $w^c_{k',k,m,h}$. The input ${}^{(2)}I^c_{k',k,m}(t)$ and the output ${}^{(2)}O^c_{k',k,m}(t)$ are defined as:

$${}^{(2)}I^c_{k',k,m}(t) = \sum_{h=1}^{H} {}^{(1)}O_h(t) w^c_{k',k,m,h}\,, \tag{6}$$

Box 1.

$$\begin{aligned}\gamma^c_{k',k} b^c_k(\mathbf{x}(t)) &= \sum_{m=1}^{M_{c,k}} \gamma^c_{k',k} r_{c,k,m} g(\mathbf{x}(t); \boldsymbol{\mu}^{(c,k,m)}, \Sigma^{(c,k,m)}) \\ &= \sum_{m=1}^{M_{c,k}} \gamma^c_{k',k} r_{c,k,m} (2\pi)^{-\frac{d}{2}} \left|\Sigma^{(c,k,m)}\right|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\sum_{j=1}^{d}\sum_{l=1}^{j} (2-\delta_{jl}) s^{(c,k,m)}_{jl} x_j(t) x_l(t) \right. \\ &\quad \left. + \sum_{j=1}^{d}\sum_{l=1}^{d} s^{(c,k,m)}_{jl} \mu^{(c,k,m)}_j x_l(t) - \frac{1}{2}\sum_{j=1}^{d}\sum_{l=1}^{d} s^{(c,k,m)}_{jl} \mu^{(c,k,m)}_j \mu^{(c,k,m)}_l \right], \end{aligned} \tag{4}$$

Figure 2. The structure of R-LLGMN



$$^{(2)}O^{c}_{k',k,m}(t) = \exp(^{(2)}I^{c}_{k',k,m}(t)), \quad (7)$$

where C is the number of discriminating classes, K_c is the number of states in class c, and $M_{c,k}$ denotes the number of GMM components in the state k of class c. In (7), the exponential function is used in order to calculate the probability of the input pattern.

The outputs of units $\{c,k,k',m\}$ in the second layer are summed and input into a unit $\{c,k,k'\}$ in the third layer. Also, the output of the fourth layer is fed back to the third layer. These are expressed as follows:

$$^{(3)}I^{c}_{k',k}(t) = \sum_{m=1}^{M_{c,k}} {}^{(2)}O^{c}_{k',k,m}(t), \quad (8)$$

$$^{(3)}O^{c}_{k',k}(t) = {}^{(4)}O^{c}_{k'}(t-1)\,{}^{(3)}I^{c}_{k',k}(t), \quad (9)$$

where $^{(4)}O^{c}_{k'}(0) = 1.0$ is for the initial phase. The recurrent connections between the fourth and the third layers play an important role in the process, which corresponds to the forward computation; see Equation (3).

The activation function in the fourth layer is described as:

$$^{(4)}I^{c}_{k}(t) = \sum_{k'=1}^{K_c} {}^{(3)}O^{c}_{k',k}(t), \quad (10)$$

$$^{(4)}O^{c}_{k}(t) = \frac{^{(4)}I^{c}_{k}(t)}{\sum_{c'=1}^{C}\sum_{k'=1}^{K_{c'}} {}^{(4)}I^{c'}_{k'}(t)}. \quad (11)$$

In the fifth layer, the unit c integrates the outputs of K_c units $\{c,k\}$ ($k = 1,\cdots,K_c$) in the fourth layer. The relationship in the fifth layer is defined as:

$$^{(5)}I^{c}(t) = \sum_{k=1}^{K_c} {}^{(4)}O^{c}_{k}(t), \quad (12)$$

$$^{(5)}O^{c}(t) = {}^{(5)}I^{c}(t). \quad (13)$$

In R-LLGMN, the posterior probability of each class is defined as the output of the last layer. After optimizing the weight coefficients $w^{c}_{k',k,m,h}$ between the first layer and the second layer, the NN can estimate the posterior probability of each class. Obviously, the R-LLGMN's structure corresponds well with the HMM algorithm. R-LLGMN, however, is not just a copy of HMM. The essential point of R-LLGMN is that the parameters in HMM are replaced by the weight coefficients $w^{c}_{k',k,m,h}$, and this replacement removes restrictions of the statistical parameter in HMM (e.g., $0 \le$ the transition probability ≤ 1, and standard deviations > 0). Therefore, the learning algorithm of R-LLGMN is simplified and can be expected to have higher generalization ability than that of HMMs. That is one of the major advantages of R-LLGMN.

A Maximum Likelihood Training Algorithm

A set of input vector streams $\tilde{\mathbf{x}}^{(n)} = (\mathbf{x}(1)^{(n)}, \mathbf{x}(2)^{(n)}, \cdots, \mathbf{x}(T_n)^{(n)})$ (n=1,...,N) and the teacher vector $\mathbf{T}^{(n)} = (T_1^{(n)}, \cdots, T_c^{(n)}, \cdots, T_C^{(n)})^{\mathrm{T}}$ are given for the learning of R-LLGMN. We assume that the network acquires the characteristics of the data through learning if, for all the streams, the last output of stream $\tilde{\mathbf{x}}^{(n)}$, namely $^{(5)}O^{c}(T_n)$, is close enough to the teacher signal $\mathbf{T}^{(n)}$. The objective function for the network is defined as:

$$J = \sum_{n=1}^{N} J_n = -\sum_{n=1}^{N}\sum_{c=1}^{C} T_c^{(n)} \log {}^{(5)}O^{c}(T_n). \quad (14)$$

The learning process attempts to minimize J, that is, to maximize the likelihood that each teacher vector $\mathbf{T}^{(n)}$ is obtained for the input stream $\tilde{\mathbf{x}}^{(n)}$.

The weight modification $\Delta w^{c}_{k',k,m,h}$ for $w^{c}_{k',k,m,h}$ is defined as:

$$\Delta w^{c}_{k',k,m,h} = \eta \sum_{n=1}^{N} \frac{\partial J_n}{\partial w^{c}_{k',k,m,h}}, \quad (15)$$

in a collective learning scheme, where $\eta > 0$ is the learning rate. Due to the recurrent connections in R-LLGMN, a learning algorithm based on the BPTT algorithm has been applied. It is supposed that the error gradient within a stream is accumulated, and weight modifications are only computed at the end of each stream; the error is then propagated backward to the beginning of the stream. The term $\frac{\partial J_n}{\partial w^c_{k',k,m,h}}$ in (15) can be defined as seen in Box 2, where $\Gamma_{(c',k''),\ (c,k)}$ is defined as:

$$\Gamma_{(c',k''),(c,k)} = \begin{cases} 1 & (c' = c;\ k'' = k) \\ 0 & \text{(otherwise)} \end{cases}, \qquad (17)$$

and ${}^{(n)}\Delta^{c'}_{k''}(t)$ is the partial differentiation of J_n to ${}^{(4)}O^{c'}_{k''}(T_n - t)$ (see Box 3 for Equation (19)):

$${}^{(n)}\Delta^{c'}_{k''}(0) = \frac{T^{(n)}_{c'}}{{}^{(5)}O^{c'}(T_n)}, \qquad (18)$$

It should be mentioned that all intermediate values of the R-LLGMN's feedforward computation are used in the calculation of Equations (16)-(19).

EMG PATTERN RECOGNITION USING R-LLGMN

The structure of the proposed EMG pattern recognition system is shown in Figure 3. This system consists of three parts in sequence: (1) EMG signal processing, (2) recurrent probabilistic neural network, and (3) discrimination rule.

1. *EMG signal processing*
   The EMG signals are processed to extract the feature patterns. In this study, feature patterns extracted from filtered EMG signals

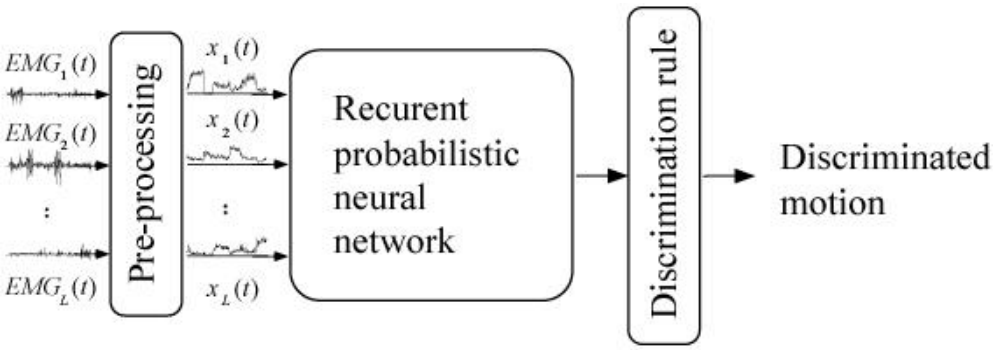
Box 2.

$$\begin{aligned}\frac{\partial J_n}{\partial w^c_{k',k,m,h}} &= \sum_{t=0}^{T_n-1}\sum_{c'=1}^{C}\sum_{k''=1}^{K_{c'}} {}^{(n)}\Delta^{c'}_{k''}(t)\frac{\partial^{(4)}O^{c'}_{k''}(T_n-t)}{\partial^{(4)}I^c_k(T_n-t)}\frac{\partial^{(4)}I^c_k(T_n-t)}{\partial^{(3)}O^c_{k',k}(T_n-t)} \\ &\quad\times\frac{\partial^{(3)}O^c_{k',k}(T_n-t)}{\partial^{(3)}I^c_{k',k}(T_n-t)}\frac{\partial^{(3)}I^c_{k',k}(T_n-t)}{\partial^{(2)}O^c_{k',k,m}(T_n-t)} \\ &\quad\times\frac{\partial^{(2)}O^c_{k',k,m}(T_n-t)}{\partial^{(2)}I^c_{k',k,m}(T_n-t)}\frac{\partial^{(2)}I^c_{k',k,m}(T_n-t)}{\partial w^c_{k',k,m,h}} \\ &= \sum_{t=0}^{T_n-1}\sum_{c'=1}^{C}\sum_{k''=1}^{K_{c'}} {}^{(n)}\Delta^{c'}_{k''}\ t)(\Gamma_{(c',k''),(c,k)} - {}^{(4)}O^{c'}_{k''}(T_n-t)) \\ &\quad\times\frac{{}^{(4)}O^{c'}_{k''}(T_n-t)}{{}^{(4)}I^{c'}_{k''}(T_n-t)}\ {}^{(4)}O^c_{k'}\ T_n-t-1)^{(2)}O^c_{k',k,m}(T_n-t)X_h(T_n-t), \end{aligned} \qquad (16)$$

Box 3.

$$\begin{aligned}{}^{(n)}\Delta^{c'}_{k''}(t+1) &= \sum_{c''=1}^{C}\sum_{k'''=1}^{K_{c''}} {}^{(n)}\Delta^{c''}_{k'''}(t)\sum_{k''''=1}^{K_{c''}}(\Gamma_{(c'',k'''),(c',k'''')} - {}^{(4)}O^{c''}_{k'''}(T_n-t)) \\ &\quad\times\frac{{}^{(4)}O^{c''}_{k'''}(T_n-t)}{{}^{(4)}I^{c''}_{k'''}(T_n-t)}\ {}^{(3)}I^{c'}_{k'',k''''}\ T_n-t). \end{aligned} \qquad (19)$$

Figure 3. Structure of the proposed EMG pattern recognition system



and raw EMG signals are used for motion discrimination. Also, the force information is extracted for motion onset detection and to determine the speed of the motion classified.

2. *Recurrent probabilistic neural network*
   The R-LLGMN described in the previous section is employed for motion discrimination. Using samples labeled with the corresponding motions, R-LLGMN learns the non-linear mapping between the EMG patterns and the forearm motions. Given an EMG feature stream with length T, the output ${}^{(5)}O^c(T)$ (c = 1,···,C) presents the posterior probability of each discriminating motion.
3. *Discrimination rule*
   In order to recognize whether the motion has really occurred or not, the force information $\sigma(t)$ is compared with a prefixed motion appearance threshold M_d. The motion is considered to have occurred if $\sigma(t)$ exceeds M_d. The entropy of R-LLGMN's outputs is also calculated to prevent the risk of misdiscrimination. The entropy is defined as:

$$H(t) = -\sum_{c=1}^{C} {}^{(5)}O^c(t) \log_2 {}^{(5)}O^c(t). \qquad (20)$$

If the entropy $H(t)$ is less than the discrimination threshold H_d, the specific motion with the largest probability is determined according to the Bayes decision rule. If not, the determination is suspended.

The discriminated motion can be used as *control commands* for HMIs, for example, powered prosthetic limbs.

Experimental Conditions

Five subjects (amputee subjects A and B, and normal subjects C, D, and E) participated in this study. Six pairs of Ag/AgCl electrodes (NT-511G: NIHON KOHDEN Corp.) with conductive paste were attached to the forearm and upper arm (Flexor Carpi Radialis (FCR), Extensor Carpi Ulnaris (ECU), Flexor Carpi Ulnaris (FCU), Biceps Brachii (BB), Triceps Brachii (TB): two pairs on FCR and one pair on the others). The subjects were asked to continuously perform six motions (C = 6) : flexion, extension, supination, pronation, hand grasping, and hand opening. The motions are shown in Figure 4.

The differential EMG signals were amplified (70 [dB]) and filtered out with a low-pass filter (cut-off frequency: 100 [Hz]) by a multi-telemeter (Web5000: NIHON KOHDEN Corp.), as shown in Figure 5, then digitized by an A/D converter (sampling frequency: 200 [Hz]; quantization: 12 [bits]).

In the experiments, the network structure of R-LLGMN is set as (C = 6), K_c = 1 (c = 1,···C),

Figure 4. Six motions used in the experiments

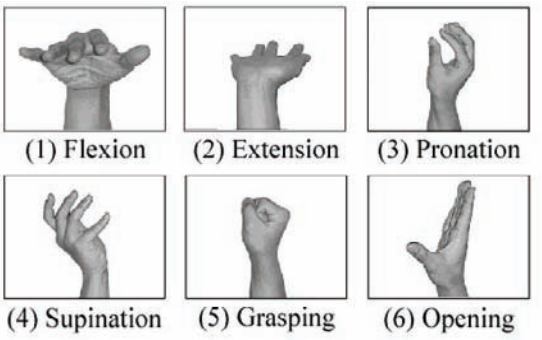


Figure 5. The multi-telemeter (Web5000) and electrodes (NT-511G) used in the experiments

and the component for each unit in the third layer is one. The parameters used are chosen to make conditions of comparison experiments as equal as possible. The lengths of training sample streams, T_n (n = 1,···,N), are set as T, which was determined with respect to the EMG features. In accordance with previous researches on EMG pattern classification (Tsuji et al., 1993; Fukuda et al., 2003), the determination threshold H_d was set to 0.5, and the motion appearance threshold M_d to 0.2. All pattern recognition experiments were conducted off-line.

Pattern Recognition of Filtered EMG Signals

First, motion discrimination experiments using filtered EMG signals were conducted to examine the performance of the proposed method. In the experiments, the training sample consists of 20 EMG patterns extracted from the filtered EMG signals for each motion.

Six channels of EMG signals (L = 6) are rectified and filtered by a second-order Butterworth filter (cut-off frequency: 1 [Hz]). The filtered EMG signals are defined as $FEMG_l(t)$ (l = 1,···,L) and are normalized to make the sum of L channels equal to 1:

$$x_l(t) = \frac{FEMG_l(t) - FEMG_l^{st}}{\sum_{l'=1}^{L} FEMG_{l'}(t) - FEMG_{l'}^{st}} \quad (l = 1, \cdots L), \tag{21}$$

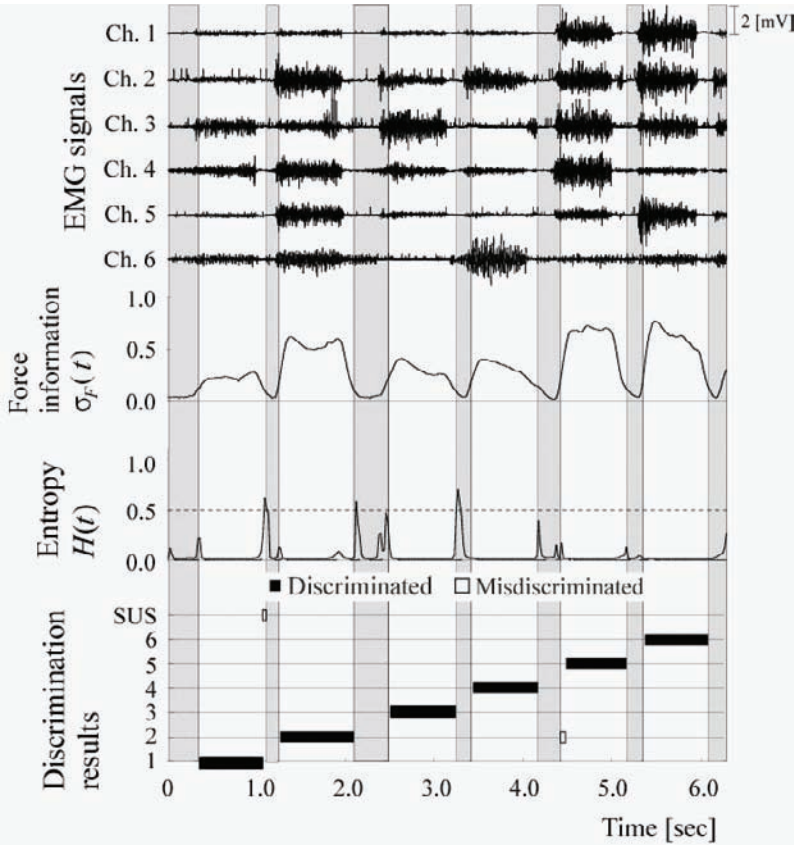
where $FEMG_l^{st}$ is the mean value of $FEMG_l(t)$ measured while the arm is relaxed. The feature vector $\mathbf{x}(t) = [x_1(t), x_2(t), \cdots x_L(t)]$ is used for the input of the neural classifier, R-LLGMN, where the dimension of R-LLGMN's input, d, is set as $d = L$. In this study, it is assumed that the amplitude level of EMG signals varies in proportion to muscle force. Force information $\sigma_F(t)$ for the input vector $\mathbf{x}(t)$ is defined as follows:

$$\sigma_F(t) = \frac{1}{L}\sum_{l=1}^{L} \frac{FEMG_l(t) - FEMG_l^{st}}{FEMG_l^{\max} - FEMG_l^{st}}, \tag{22}$$

where $FEMG_l^{\max}$ is the mean value of $FEMG_l(t)$ measured while maintaining the maximum arm voluntary contraction.

An example of the discrimination results of subject A is shown in Figure 6. The subject was an amputee (51-year-old male), whose forearm, three cm from the left wrist joint, was amputated when he was 18 years old as the result of an accident. He has never used EMG controlled prosthetic limbs and usually uses a cosmetic hand. In the experiments, he was asked to perform six motions in the order continuously for six seconds. Figure 6

Figure 6. Example of the discrimination results for filtered EMG signals (subject A)



plots six channels of the input EMG signals, the force information $\sigma_F(t)$, the entropy $H(t)$, and the discrimination results. The labels of the vertical axis in the discrimination results correspond to the motions shown in Figure 4, and SUS means that the determination was suspended. The gray areas indicate that no motion was determined because the force information was less than M_d. Incorrect determination was eliminated using the entropy. Figure 6 demonstrates that the proposed method achieves high discrimination accuracy with filtered EMG signals during continuous motion.

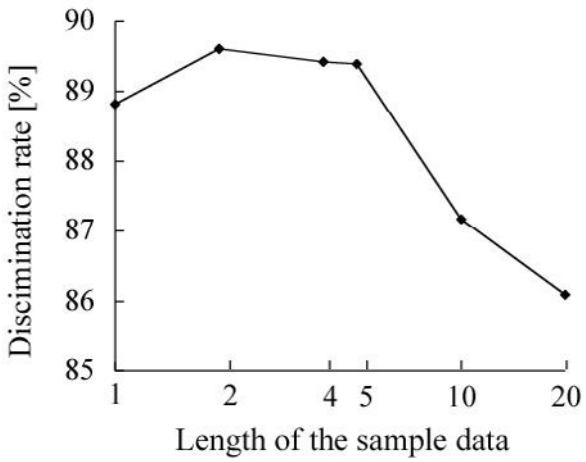
The discrimination accuracy for five subjects was then investigated, and LLGMN and an MLP classifier were used for comparison. The same preprocessing method and discrimination rule were applied to the experiments using LLGMN and MLP. The number of units in the input layer of LLGMN was equal to the dimension of input signal (L). Units in the hidden layer corresponded to the Gaussian components in GMM, the number of which was set in the same manner as for the R-LLGMN. The output layer included C units, and each unit gave the posterior probability for the input pattern. In contrast, MLP had four layers (two hidden layers), and the units of the layers were set at 6, 10, 10, and 6. Each output of MLP corresponded to a motion, and all six outputs were normalized to make the sum of all outputs equal 1.0 for comparison with R-LLGMN and LLGMN. The learning procedure of MLP continued until the sum of the square error was less than 0.01, where the learning rate was 0.01. However, if the

Table 1. Discrimination results of five subjects with filtered EMG signals

Type of the methods		MLP	LLGMN	R-LLGMN
Subject A (Amputee)	*DR*	73.4	94.0	99.1
	SD	7.9	5.5	0.0
Subject B (Amputee)	*DR*	46.5	82.8	89.3
	SD	12.3	0.0	0.4
Subject C (Normal)	*DR*	44.2	88.5	93.0
	SD	10.4	0.0	0.1
Subject D (Normal)	*DR*	69.8	88.7	93.5
	SD	10.0	0.2	0.0
Subject E (Normal)	*DR*	69.2	89.3	92.8
	SD	7.0	0.1	0.0

DR : Discrimination rate [%], *SD* : Standard deviation [%]

Figure 7. Discrimination rates for various data lengths (subject B)



sum of the square error after 50,000 iterations was still not less than 0.01, the learning procedure was stopped. In all three methods, ten different sets of initial weights (all randomized between [0, 1]) were used.

Discrimination rate, which is defined as the ratio of correctly classified data to the total test set, is used to evaluate discrimination accuracy of three methods. The mean values and the standard deviations of the discrimination rates are shown in Table 1. It can be seen that R-LLGMN achieved the best discrimination rate among all three methods and had the smallest standard deviation.

Also, the classification results were examined by altering the experiment conditions, such as the length of sample data. Experiments were performed using various lengths of sample data. For each sample data, R-LLGMN was trained with ten different sets of initial weights, which were randomly chosen in the range [0, 1]. The mean values of the discrimination rates for each length are shown in Figure 7, where the standard deviations are all very small, close to 0. It can be seen from Figure 7 that the discrimination rate maintains a high level when the sample data is of an appropriate length (T). However, if $T > 5$, it is

too long to train R-LLGMN using filtered EMG signals. The discrimination rate tends to deteriorate because R-LLGMN, which was trained using the long-length sample data, failed to discriminate the switching of motions.

Pattern Recognition of Raw EMG Signals

This subsection presents pattern recognition experiments of time series of raw EMG signals. In the previously proposed methods for classifying the intended motion of an operator, the filtered or smoothed EMG signals (Fukuda et al., 1997, 2003; Kelly et al., 1990; Tsuji et al., 1993, 2000) or the extracted characteristics in a fixed time window (Hiraiwa et al., 1989; Hudgins et al., 1993) have been used as the input vector to the NN classifier. However, these signal-processing steps result in considerable phase delay and time delay caused by the low-pass filtering and the time window operation. To avoid such delay, raw EMG signals without any preprocessing are used as the input to R-LLGMN. The experiments were performed with the subjects (A, B, C, D, and E) who had experience in manipulating the EMG signals.

As raw EMG signals, six channels of EMG signals (L=6) sampled from the input of multi-telemeter are denoted by $REMG_l(t)$ (l = 1,···,L). For the case of raw EMG signals, force information $\sigma_R(t)$ is obtained calculating moving average within the length T:

$$\sigma_R(t) = \frac{1}{L}\sum_{l=1}^{L}\frac{\overline{REMG_l}(t)}{\overline{REMG_l}^{\max}}, \tag{23}$$

$$\overline{REMG_l}(t) = \frac{1}{T}\sum_{j=0}^{T-1}\left|REMG_l(t-j)\right|, \tag{24}$$

where $\overline{REMG_l}^{\max}$ is the premeasured integral EMG of each channel under the maximum voluntary contraction. Also, it should be noted that $REMG_l(t-j) = 0$ when $t - j < 0$.

The input vector $\mathbf{x}(t)$ (t = 1,···,T) of R-LLGMN is normalized $REMG_l(t)$ with $\sigma_R(t)$ as

$$x_l(t) = \sigma_R^{-1}(T)\,REMG_l(t). \tag{25}$$

Here, the normalization enables R-LLGMN to discriminate motions from a pattern of all channels as well as from the amplitude of the raw EMG signals.

In pattern recognition experiments of raw EMG signals, the length of training sample stream T is set as 20. Eight sample streams are used for each motion. The threshold for motion onset detection M_d is 0.155.

Figure 8 provides an example of the classification results of subject A. The figure shows six channels of the raw EMG signals, the force information $\sigma_R(t)$, the entropy $H(t)$ calculated from the output probability of R-LLGMN, and the classification results of the R-LLGMN. The discrimination rate was about 95.5% in this experiment. It can be seen that R-LLGMN generates acceptable classification results during continuous motion, and the entropy is low during motions except for the motion one (Flexion). It indicates that R-LLGMN can discriminate the hand and forearm motions from the raw EMG signals, even for control purposes.

Comparisons were conducted with discrimination results of MLP, LLGMN, and R-LLGMN using filtered EMG signals. It should be noted that due to the stochastic nature of raw EMG signals, MLP and LLGMN could not learn motion patterns of raw EMG signals. The network structures of MLP and LLGMN were set to the same as those in pattern recognition experiments of filtered EMG, and the beginning and ending of motions were recognized according to the force information $\sigma_R(t)$. The discrimination threshold H_d was not used in the comparison, so that all classification results were used for comparison. Each experiment was repeated ten times with different randomly chosen initial weights. Table 2 depicts the mean values and the standard deviations of the discrimina-

Figure 8. Example of the discrimination result for raw EMG signals (subject A)

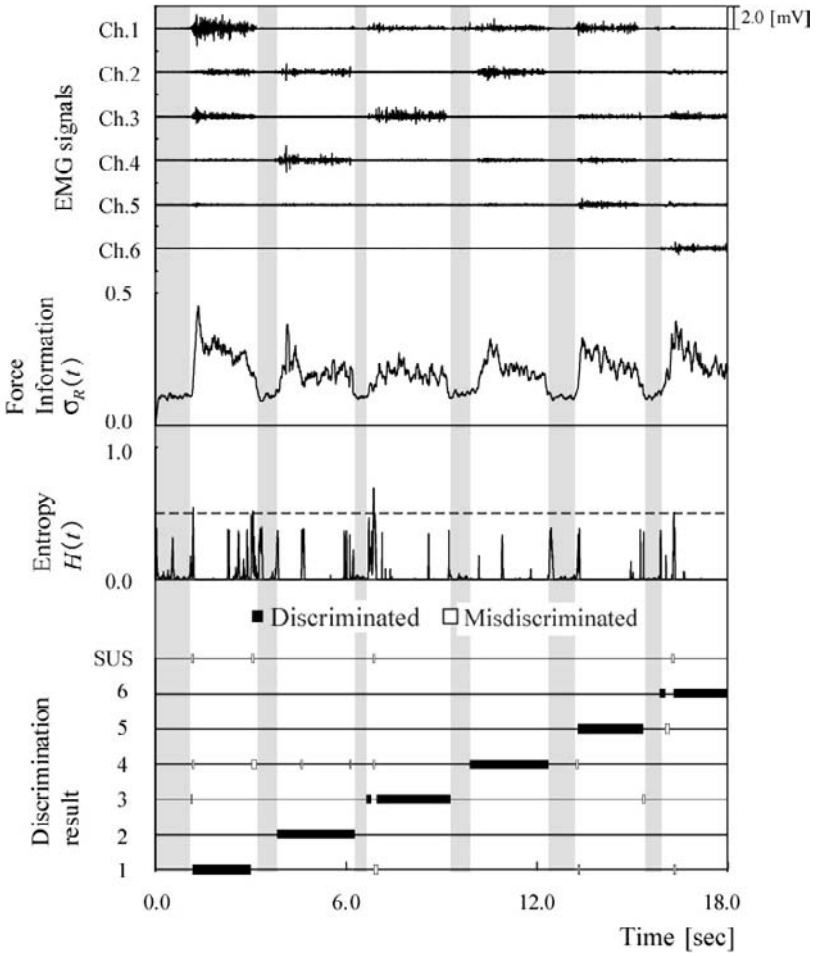


Table 2. Motion discrimination results for raw EMG signals, comparing with methods using filtered EMG signals

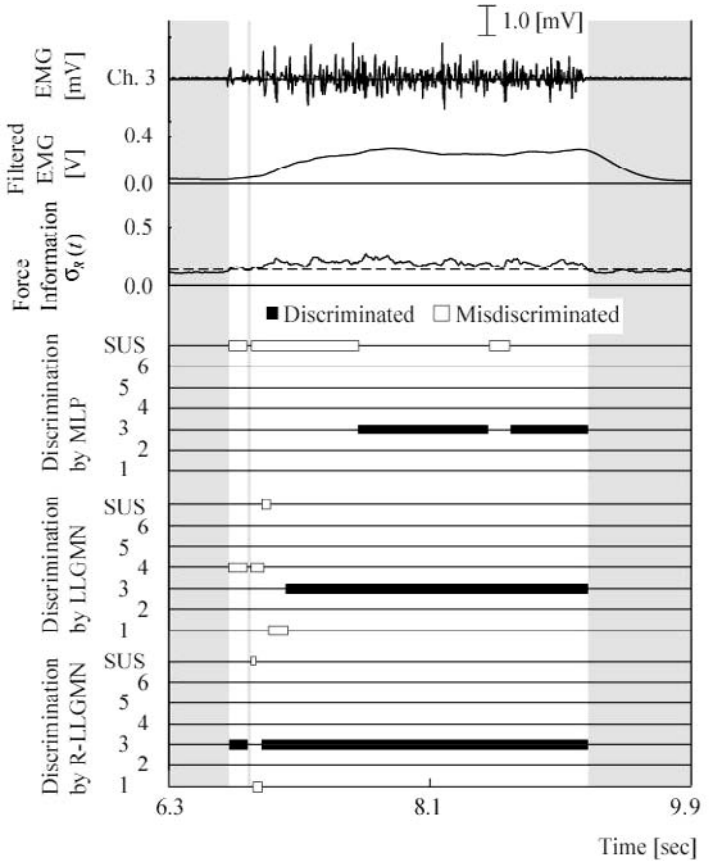
Type of the methods		MLP (Filtered EMG)	LLGMN (Filtered EMG)	R-LLGMN (Filtered EMG)	R-LLGMN (Raw EMG)
Subject A (Amputee)	*DR*	66.1	89.8	96.1	93.8
	SD	14.0	0.0	0.0	0.0
Subject B (Amputee)	*DR*	70.1	89.3	92.5	91.2
	SD	10.8	0.0	0.0	1.3
Subject C (Normal)	*DR*	80.5	82.9	94.2	94.1
	SD	8.1	0.0	0.0	0.4
Subject D (Normal)	*DR*	78.9	88.3	97.4	90.4
	SD	4.1	0.0	0.0	0.9
Subject E (Normal)	*DR*	75.8	85.9	90.7	91.0
	SD	4.5	0.0	0.0	1.8

DR : Discrimination rate [%], *SD* : Standard deviation [%]

tion rates for five subjects. Due to the filtering processes, onsets of the filtered EMG signals are delayed, and the EMG patterns vary significantly in time domain during the transient phase. Since MLP cannot deal well with time-varying patterns, MLP's discrimination result is the worst among these methods. Although LLGMN shows better discrimination accuracy than MLP due to the statistical model incorporated in its structure, it still provides poor discrimination accuracy since the model is static. Consequently, it is can be concluded that phase delay due to the filtering processes is one of the major causes of degradation in the discrimination results in cases of MLP and LLGMN. In contrast, R-LLGMN provides superior discrimination results for both the filtered EMG signals and the raw EMG. Also, we found that patterns of raw EMG signals are much more complicated than that of filtered EMG signals, and training and estimation of R-LLGMN using raw EMG signals are more difficult. Therefore, the classification performance of R-LLGMN with filtered EMG signals is a little higher than that using raw EMG signals. However, since no signal processing is used, the latter has a faster response. There is thus a trade-off between discrimination accuracy and response speed.

The response time of raw EMG-based motion discrimination was further investigated, and the proposed method and traditional classifiers (MLP and LLGMN) were compared. Figure 9 illustrates the signals magnified from 6.3 s to 9.9 s in Figure 8 during the wrist extension motion. This figure depicts the EMG signal of the channel 3, the fil-

Figure 9. Changes of the discrimination results by three types of neural networks (subject A)



tered EMG signal that is rectified and filtered out by the second-order Butterworth low-pass filter (cut-off frequency: 1.0 [Hz]), the force information $\sigma_R(t)$, and the discrimination results of three comparison methods. The MLP and LLGMN used the features extracted from filtered EMG signals as the input, while the R-LLGMN achieved motion discrimination based on the raw EMG signals. It can be seen from the figure that there is a considerable phase delay between the raw EMG and the filtered EMG signals, which causes the misdiscrimination in the results of MLP and LLGMN. In contrast, using the raw EMG signals, R-LLGMN achieves higher discrimination accuracy than the others, and a correct classification is made just after the beginning of motion. It was also found that the discrimination rates of both MLP and LLGMN decreased considerably when the cut-off frequency of the low-pass filter increased. The increase of the cut-off frequency results in filtered EMG signals containing high frequency components, so that the learning of the NNs becomes very difficult.

Then, discrimination accuracy during the beginning and ending of motions was investigated. In these experiments, EMG signals during 100 msec from onset and 100 msec before ending of each motion were used. Similarly, MLP and LLGMN using filtered EMG signals were used for comparison. Table 3 presents the discrimination results for five subjects using three different methods. The mean values and the standard deviations of the discrimination rates are computed for ten randomly chosen initial weights. From this table, it can be seen that R-LLGMN attained the best discrimination rates during the beginning and ending of motions; therefore, the R-LLGMN provides superior response performance.

DISCUSSIONS

A new EMG pattern recognition method using R-LLGMN is proposed to improve discrimination accuracy when dealing with non-stationary EMG signals. R-LLGMN performs both the filtering process and pattern classification within the same network architecture, so the proposed method outperforms the previous methods with filtered EMG and raw EMG patterns. What is even more encouraging is that the response time of discrimination results can also be shortened by using raw EMG signals.

In the studies on human-machine interfaces (HMIs), it is widely believed that the response time is an important aspect, especially for practi-

Table 3. Discrimination results for five subjects

Type of the methods		BPNN	LLGMN	R-LLGMN
Subject A (Amputee)	*DR*	30.4	58.3	89.6
	SD	13.6	4.1	0.3
Subject B (Amputee)	*DR*	67.2	63.7	75.0
	SD	11.0	0.2	0.0
Subject C (Normal)	*DR*	49.4	50.0	91.7
	SD	11.5	0.0	0.0
Subject D (Normal)	*DR*	52.8	58.3	73.8
	SD	3.8	0.0	0.0
Subject E (Normal)	*DR*	64.4	63.3	66.7
	SD	3.0	0.0	0.0

DR : Discrimination rate [%], *SD* : Standard deviation [%]

cal application systems. For HMIs used in daily activities, it has been mentioned that techniques for real-time classification are needed in order to decrease global time delay of response, which would reduce the operator's mental burden and increase the range of applications and number of potential users (Chang et al., 1996; Vuskovic & Du, 2002). A classification system based on digital signal processors (DSP) was used to realize the pattern classification algorithm for fast processing (Chang et al., 1996). Vuskovic and Du (2002) attempted to simplify a fuzzy ARTMAP network used for EMG classification, which resulted in overall smaller computational times. On the other hand, since the EMG signals include high-frequency components, adequate signal processes such as low-pass filtering are necessary in order to extract meaningful information for HMIs. Actually, this low-pass filtering process increases the time delay.

In contrast to these previous studies, which focus on reduction of the computational time (complexity) of classifiers, a pattern recognition technique that directly uses raw EMG signals is an interesting choice. Given the experimental results in the previous section, it is expected that improved response performance is possible by adopting the proposed raw EMG pattern recognition scheme into traditional HMIs, which use filtered EMG patterns (Fukuda et al., 1997, 2003; Kelly et al., 1990; Tsuji et al., 1993, 2000). Further studies should focus on this idea.

This chapter introduced R-LLGMN in order to make effective use of the non-stationary (time-varying) characteristics in EMG signals. In recent years, time-frequency analysis has attracted increasing attention for representing the non-stationary essence of frequency domain (Englehart et al., 1999, 2001; Hussein & Granat, 2002). Since the wavelet transform results in a good time-frequency resolution, it has become a very popular feature extraction method for time-frequency representation of EMG signals. Based on the idea of building prior information into neural network design, the algorithm of wavelet transform can be incorporated into the probabilistic neural network, so that the PNNs could process frequency information of EMG signals more effectively.

SUMMARY

This chapter proposes a new EMG pattern recognition method based on a recurrent log-linearized Gaussian mixture network (R-LLGMN). Because of the recurrent connections in the R-LLGMN's structure, the temporal information of EMG signals can be used to improve discrimination accuracy.

To examine the discrimination capability and accuracy of the proposed method, EMG pattern recognition experiments were conducted with five subjects. In the experiments, the proposed method achieved high discrimination accuracy for varying EMG signals, and its discrimination results are superior to those of the LLGMN and MLP classifiers. We found that the discrimination results change when different lengths of sample stream T are used. The length T should be well modulated according to the input signals. For example, to discriminate filtered EMG signals, T should be less than five.

Even more encouraging is the outcome of EMG pattern recognition experiments using the non-stationary time series of raw EMG signals. Results of these experiments demonstrate that R-LLGMN performs both the filtering process and pattern recognition within the same network architecture and can realize a relatively high discrimination rate that is good enough for control purposes. It should be noted that there is a trade-off between discrimination accuracy and response speed when using R-LLGMN as a classifier. In practical applications, such as prosthetic control, the latter may be preferred.

REFERENCES

Chang, G. C., Kang, W. J., Luh, J. J., Cheng, C. K., Lai, J. S., Chen, J. J., & Kuo, T. S. (1996). Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface. *Medical Engineering & Physics, 18*, 529-537.

Englehart, E., Hudgins, B., & Parker, A. (2001). A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering, 48*, 302-311.

Englehart, E., Hudgins, B., Parker, A., & Stevenson, M. (1999). Classification of the myoelectric signal using time-frequency based representations. *Medical Engineering & Physics, 21*, 431-438.

Fukuda, O., Tsuji, T., & Kaneko, M. (1997). An EMG controlled robotic manipulator using neural networks. *Proceedings of the IEEE International Workshop on Robot and Human Communication*, 442-447.

Fukuda, O., Tsuji, T., & Kaneko, M. (1999). An EMG-controlled pointing device using a neural network. *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics, 4*, 63-68.

Fukuda, O., Tsuji, T., Kaneko, M., & Ohtsuka, A. (2003). A human-assisting manipulator teleoperated by EMG signals and arm motions. *IEEE Transactions on Robotics and Automation, 19*, 210-222.

Hiraiwa, A., Shimohara, K., & Tokunaga, Y. (1989). EMG pattern analysis and classification by neural network. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 3*, 1113-1115.

Hudgins, B., Parker, P., & Scott, R.N. (1993). A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering, 40*, 82-94.

Hussein, S. E., & Granat, M. H. (2002). Intention detection using a neuro-fuzzy EMG classifier. *IEEE Engineering in Medicine and Biology Magazine, 21*(6), 123-129.

Kang, W. J., Shiu, J. R., Cheng, C. K., Lai, J. S., Tsao, H. W., & Kuo, T. S. (1995). The application of cepstral coefficients and maximum likelihood method in EMG pattern recognition. *IEEE Transactions on Biomedical Engineering, 42*, 777-785.

Kelly, M. F., Parker, P. A., & Scott, R. N. (1990). The application of neural networks to myoelectric signal analysis: A preliminary study. *IEEE Transactions on Biomedical Engineering, 37*, 221-230.

Lamounier, E., Soares, A., Andrade, A., & Carrijo, R. (2002). A virtual prosthesis control based on neural networks for EMG pattern classification. *Proceedings of the 6th IASTED International Conference on Artificial Intelligence and Soft Computing*, 456-461.

Lusted, H. S., & Knapp, R. B. (1996, October). Controlling computers with neural signals. *Scientific American,* 82-87.

Rabiner, L.R. (1989). A tutorial on hidden Markov model and selected applications in speech recognition. *Proceedings of the IEEE, 77*, 257-286.

Specht, D.F. (1990). Probabilistic neural networks. *Neural Networks, 3*, 109-118.

Tsuji, T., Bu, N., Fukuda, O., & Kaneko, M. (2003). A recurrent log-linearized Gaussian mixture network. *IEEE Transactions on Neural Networks, 14*, 304-316

Tsuji, T., Fukuda, O., Ichinobe, H., & Kaneko, M. (1999). A log-linearized Gaussian mixture network and its application to EEG pattern classification. *IEEE Transactions on Systems, Man,*

and Cybernetics, Part C: Applications and Reviews, 29, 60-72.

Tsuji, T., Fukuda, O., Kaneko, M., & Ito, K. (2000). Pattern classification of time-series EMG signals using neural networks. *International Journal of Adaptive Control and Signal Processing, 14,* 829-848.

Tsuji, T., Ichinobe, H., Ito, K., & Nagamachi, M. (1993). Discrimination of forearm motions from EMG signals by error back propagation typed neural network using entropy (in Japanese). *Transactions of the Society of Instrument and Control Engineers, 29,* 1213-1220.

Vuskovic, M., & Du, S. (2002). Classification of prehensile EMG patterns with simplified fuzzy ARTMAP networks. *Proceedings of the 2002 International Joint Conference on Neural Networks,* 2539-2545.

Zhang, G.D. (2000). Neural network for classification: A survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 30,* 451-462.

Compilation of References

Aarajt, N., Ravit, S., Raghunathant, A., & Jhat, N.K. (2006). Architectures for efficient face authentication in embedded systems. *Proceedings of the Design, Automation and Test in Europe Conference*. 2, 1–6.

Abbass, H.A. (2003). Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. *Proceedings of the IEEE 2003 Conference on Evolutionary Computation, 3,* 2074–2080.

Abdelghani, I., & Amara, N. (2006). Deux approches neuronales pour la vérification hors-ligne de la signature manuscrite. *Proceedings of the Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle*.

Adini, Y., Moses, Y., & Ullman, S. (1997). Face recognition: The problem of compensating for changes in illumination direction. *Pattern Analysis and Machine Intelligence, IEEE Transactions, 19*(7), 721–732.

Agazzi, O., Kuo, S., Levin, E., & Pieraccini, R. (1993). Connected and degraded text recognition using planar hidden Markov models. Proc. ICASSP, V113–116.

Alizadeh, A.A., et al. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature, 403*(6769), 503–511.

Alkoot, F., & Kittler, J. (2000). Feature selection for an ensemble of classifiers. *Proceedings of the World Multiconference on Systematics, Cybernetics and Informatics.*

Almauallium, H., & Dietterich, T.G. (1991). Learning with many irrelevant features. *Proceedings of Ninth National Conference on Artificial Intelligence*, 2, 547–552.

Almeida, L.B. (1994). The fractional Fourier transform and time-frequency representations. *IEEE Transactions on Signal Processing, 42*(11), 3084–3091.

Al-Shoshan, A. (2006). Handwritten signature verification using image invariants and dynamic features. *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation,* 173–176.

Ammar M., Yoshida, Y., &. Fukumura, T. (1990). Structural description and classification of signature images. *Pattern Recognition, 23,* 697–710.

Ammar, M. (1991). Progress in verification of skilfully simulated handwritten signatures. *International Journal of Pattern Recognition and Artificial Intelligence, 5*(1-2), 337–351.

Ammar, M., Yoshida, Y., & Fukumura, T. (1986). A new effective approach for off-line verification of signatures by using pressure features. *Proc. 8th Int. Conf. on Pattern Recognition* (pp. 566–569).

Ammar, M., Yoshida, Y., & Fukumura, T. (1999). Off-line preprocessing and verification of signatures. *International Journal of Pattern Recognition and Artificial Intelligence, 2*(4), 589–602.

Anderson, J.S., Carandini, M., & Ferster, D. (2000). Orientation tuning of input conductance, excitation, and inhibition in cat primary visual cortex. *Journal of Neurophysiology, 84,* 909–926.

Ando, S., & Nakajima, M. (2003). An active search method for local individual features in off-line signature verification. *Systems and Computers in Japan, 34*(12), 64–76.

Arica, N., & Yarman-Vural, F.T. (2002). Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(6), 801–813.

Armand, S., Blumenstein, M., & Muthukkumarasamy, V. (2006). Off-line signature verification using the enhanced modified direction feature and neural-based classification. *Proceedings of the International Joint Conference on Neural Networks*, 1663–1669.

Arulampalam, G., & Bouzerdoum, A. (2003). A generalized feedforward neural network architecture for classification and regression. *Neural Networks, 16*(5-6), 561–568.

Asur, S., Parthasarathy, S., & Ucar, D. (2006). An ensemble approach for clustering scalefree graphs. *Proceedings of the Workshop on Link Analysis: Dynamics and Static of Large Networks*. Philadelphia, Pennsylvania.

Ayad, H., & Kamel M.S. (2003). Refined shared nearest neighbors graph for combining multiple data clusterings. *Proceedings of the 5th International Symposium on Intelligent Data Analysis. Lecture Notes in Computer Science, 2810*, 307–318.

Azuaje, F. (2000). *Gene expression patterns and cancer classification: A self adaptive and incremental neural approach* (Tech. Rep. TCD-CS-2000-26). Dublin: Trinity College, Computer Science Department.

B.D.T. Inc. (2002). Comparing FPGAs and DSPs for embedded signal processing.

B.D.T. Inc. (2004). Using general-purpose processors for signal processing. *Proceedings of the ARM Developers' Conference*.

B.D.T. Inc. (2005). Processors for consumer audio/video applications. *GSPx*.

Back, T., Fogel, D., & Michalewicz, Z. (1997). *Handbook on evolutionary computation*. Oxford University Press.

Bajaj, R., & Chaudhury, S. (1997). Signature verification system using multiple neural classifiers. *Pattern Recognition, 30*(1), 1–7.

Bajscy, R., & Broit, C. (1982). Matching of deformed images. *Proc. ICPR*, 351–353.

Balci, K., & Atalay, V. (2002). PCA for gender estimation: Which eigenvector contribute? *Proceedings of the Sixteenth International Conference on Pattern Recognition*, Vol. 3, 363–366.

Barbara, D. (2000). *An introduction to cluster analysis for data mining*. Retrieved November 12, 2003, from http://www-users.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf.

Barrón, R., Suárez-Guerra, S., & Moctezuma, C. (1999). *Reconocimiento de comandos verbales utilizando cuantización vectorial y redes neuronales* (Tech. Rep. Serie Roja, No. 40). Computing Research Center, The National Polyechnic Institute of Mexico.

Bastos, L., Bortolozzi, F., Sabourin, R., & Kaestner, C. (1997). Mathematical modelation of handwritten signatures by conics. *Revista da Sociedade Paranaese de Matemática, 18*, 135–146.

Batur, A.U., Flinchbaugh, B.E., & Hayes III, M.H. (2003). A DSP-based approach for the implementation of face recognition algorithms. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2, 253–256.

Beare, R., & Bouzerdoum, A. (1999). Biologically inspired local motion detector architecture. *Journal of the Optical Society of America A: Optics, Image Science, and Vision, 16*(9), 2059–2068.

Belhumeur, P.N., Hespanha, J.P., & Kriegman, D.J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 19*(7), 711–720.

Bernal, J., Bobadilla, J., & Gomez, P. (2000). *Reconocimiento de voz y fonética acústica*. Madrid, España: Alfa-Omega.

Beymer, D., & Poggio, T. (1995). Face recognition from one example view. *Proceedings of Fifth International Conference on Computer Vision*, 500–507.

Bicego, M., Murino, V., & Figueiredo, M. (2004). Similarity based clustering of sequences using hidden Markov models. *Pattern Recognition, 37*(12), 2281–2291.

Bigdeli, A., Biglari-Abhari, M., Leung, S.H.S., & Wang, K.I.K. (2004). Multimedia extensions for a reconfigurable processor. *Proceedings of the 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing*, 426–429.

Bins, J. (2000). *Feature selection from huge sets in the context of computer vision* [doctoral dissertation]. Colorado State University.

Bippus, R., & Märgner, V. (1999). Script recognition using inhomogeneous P2DHMM and hierarchical search space reduction. *Proc. ICDAR*, 773–776.

Bishop, C.M. (1996). *Neural networks for pattern recognition*. Oxford University Press.

Bittner, M., et al. (2000). Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature, 406*(6795), 536–540.

Black, M.J., Fleet, D.J., & Yacoob, Y. (2000). Robustly estimating changes in image appearance. *Computer Vision and Image Understanding, 78*(1), 8–31.

Blake, C., & Merz, C. (1998). *UCI repository of machine learning databases*. University of California, Department of Information and Computer Science. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Blatzakis, H., & Papamarkos, N. (2001). A new signature verification technique based on a two-stage neural network classifier. *Engineering Applications of Artificial Intelligence, 14*, 95–103.

Blum, A.L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 245–271.

Blumenstein, M., & Verma, B. (1999). Neural-based solutions for the segmentation and recognition of difficult handwritten words from a benchmark database. *Proceedings of the Fifth International Conference on Document Analysis and Recognition* (pp. 281–284). Bangalore, India: IEEE Computer Society Press.

Blumenstein, M., & Verma, B. (2001). Analysis of segmentation performance on the CEDAR benchmark database. *Proceedings of the Sixth International Conference on Document Analysis and Recognition* (pp. 1142–1146). Seattle: IEEE Computer Society Press.

Blumenstein, M., Liu, X.Y., & Verma, B. (2004). A modified direction feature for cursive character recognition. *Proceedings of the International Joint Conference on Neural Networks* (pp. 2983–2987). Budapest, Hungary: IEEE Computer Society Press.

Blumenstein, M., Verma, B., & Basli, H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 137–141). Edinburgh, UK: IEEE Computer Society Press.

Borg-Graham, L.J., Monier, C., & Fregnac, Y. (1998). Visual input evokes transient and strong shunting inhibition in visual cortical neurons. *Nature, 393*(6683), 369–373.

Bottou, L., et al., (1994). Comparison of classifier methods: A case study in handwriting digit recognition. *International Conference of Pattern Recognition (ICPR-94)*, 77–87.

Boulard, D. S. (1996). A new ASR approach based on independent processing and recombination of frequency bands. In *Proceedings of the International Conference on Spoken Language Processing* (ICSLP 1996) (Vol 1. pp. 426-429) ACM Digital Library.

Boulis, C., & Ostendorf, M. (2004). Combining multiple clustering systems. *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Lecture Notes in Computer Science, 3202*, 63-74.

Bouzerdoum, A. (1993). The elementary movement detection mechanism in insect vision. *Philosophical Transactions of the Royal Society of London, B-339*, 375–384.

Bouzerdoum, A. (1994). A hierarchical model for early visual processing. *Proceedings of the Human Vision, Visual Processing, and Digital Display V, Proceedings of SPIE 2179*, San Jose, California, 10–17.

Bouzerdoum, A. (1999). A new class of high-order neural networks with nonlinear decision boundaries. *Proceedings of the Sixth International Conference on Neural Information Processing, 3*, 1004–1009. Perth, Australia.

Bouzerdoum, A. (2000). Classification and function approximation using feedforward shunting inhibitory artificial neural networks. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 613–618.

Bouzerdoum, A., & Pinter, R.B. (1992). Nonlinear lateral inhibition applied to motion detection in the fly visual system. In R.B. Pinter & B. Nabet (Eds.), *Nonlinear vision* (pp. 423–450). Boca Raton, FL: CRC Press.

Bouzerdoum, A., & Pinter, R.B. (1993). Shunting inhibitory cellular neural networks: Derivation and stability analysis. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40*, 215–221.

Bower, J. (2004). A system-on-a-chip for audio encoding. *Proceedings of the International Symposium on System-on-Chip*, 149–155.

Bozinovic, R.M., & Srihari, S.N. (1989). Off-line cursive script word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 11*(1), 68–83.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.

Brislawn, C.M., Bradley, C.B.J., Onyshczak, R., & Hopper, T. (1996). The FBI compression standard for digitized fingerprint images. *Proceedings of the SPIE Conference 2847*, Denver, Colorado, 344–355.

Britto Jr., A. Sabourin, R., Bortolozzi, F., & Suen, C.Y. (2004). Foreground and background information in an HMM-based method for recognition of isolated characters and numeral strings. *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition* (pp. 371–376). Tokyo, Japan: IEEE Computer Society Press.

Brodatz, P. (1966). *Texture: A photographic album for artists and designers.* New York: Dover.

Broeders, A.P.A. (2001). Forensic speech and audio analysis forensic linguistics. *Proc. 13th INTERPOL Forensic Science Symposium*. Lyon, France, October 16-19, D2.51-D2.84.

Broggi, A., Bertozzi, M., Fascioli, A., & Sechi, M. (2000). Shape-based pedestrian detection. *Proceedings of the IEEE Intelligent Vehicles Symposium*, 215–220.

Brown, G. (2001). On the effectiveness of negative correlation learning. *Proceedings of the First UK Workshop on Computational Intelligence*, 57–62.

Brown, G. (2003). *Diversity in neural network ensembles* [doctoral thesis]. University of Birmingham, School of Computer Science.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: A survey and categorisation. *Journal of Information Fusion, 6*(1), 5–20.

Brown, M.K., & Ganapathy, S. (1983). Preprocessing techniques for cursive script word recognition. *Pattern Recognition, 16*(5), 447–458.

Brummer, N., & du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech and Language, 20*, 230–275.

Brunelli, R., & Poggio, T. (1992). Hyberbf networks for gender classification. *Proceedings of the DARPA Image Understanding Workshop*, 311–314.

Brzozowski, J.A., & Łuba, T. (2003). Decomposition of Boolean functions specified by cubes. *Journal of Multiple-Valued Logic and Soft Computing, 9*, 377–417.

Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining, 2*(2), 121–167.

Burges, C.J.C., Be, J.I., & Nohl, C.R. (1992). Recognition of handwritten cursive postal words using neural

networks. *Proceedings of the 5th USPS Advanced Technology Conference* (pp. 117–124).

Burr, D.J. (1981). A dynamic model for image registration. *Computer Graphics, and Image Proc., 15*, 102–112.

Buxton, H. (2003). Learning and understanding dynamic scene activity: A review. *Image and Vision Computing, 21*(1), 125–136.

Buzo, A., & Gray, R. (1980). Speech coding based upon quantization. *IEEE Transaction on Communications, COM-28*(1).

Cai, J., & Liu, Z.-Q. (1999). Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(3), 263–270.

Cai, Y., Sun, X., & Jia, P. (2004). Negative correlation learning approach for t-s fuzzy models. *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics, 3*, 2254–2259.

Calderon, A., Roa, S., & Victorino, J. (2003). Handwritten digit recognition using convolutional neural networks and Gabor filter. *Proceedings of the International Congress on Computational Intelligence*, Medellin, Colombia.

Camastra, F., & Vinciarelli, A. (2003). Combining neural gas and learning vector quantization for cursive character recognition. *Neurocomputing, 51*, 147–159.

Campbell, J.P. (1997). Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(8).

Campbell, J.P. (1999). Speaker recognition. In A. Jain, R. Bolle, & S. Pankanti (Eds.), *Biometrics: Personal identification in networked society* (pp. 165–189). Kluwer Academic Publishers.

Campbell, W.M., Brady, K.J., Campbell, J.P., Granville, R., & Reynolds, D.A. (2006). Understanding scores in forensic speaker recognition. *Proceedings of the Odyssey 2006: The Speaker and Language Recognition Workshop*, June 2006.

Campbell, W.M., Reynolds, D.A., Campbell, J.P., & Brady, K.J. (2005). Estimating and evaluating confidence for forensic speaker recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1*, 717–720.

Cantu-Paz, E. (1997). *A survey of parallel genetic algorithms* [Technical Report 91003]. Urbana-Champaign, IL: University of Illinois, Department of Computer Science and Illinois Genetic Algorithms Laboratory.

Cao, J., Ahmadi, M., & Shridhar, M. (1995). Recognition of handwritten numerals with multiple feature and multistage classifier. *Pattern Recognition, 28*(3), 153–159.

Cardot, H., Revenu, M., Victorri, B., & Revillet, M. (1994). A static signature verification system based on a cooperating neural network architecture. *International Journal on Pattern Recognition and Artificial Intelligence, 8*(3), 679–692.

Casey, R.G., & Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(7), 690–706.

Caurana, R.A., & Freitag, D. (1994). Greedy attribute selection. *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, 28–36.

Cha, S. (2001). *Use of distance measures in handwriting analysis* [unpublished doctoral dissertation]. Buffalo, NY: State University of New York at Buffalo.

Chalechale, A., Naghdy, G., Premaratne, P., & Mertins, A. (2004). Document image analysis and verification using cursive signature. *IEEE International Conference on Multimedia and Expo, 2*, 887– 890.

Chan, Z.S.H., & Kasabov, N. (2005). A preliminary study on negative correlation learning via correlation-corrected data (NCCD). *Neural Processing Letters, 21*, 207–214.

Chandra, A., & Yao, X. (2004). Divace: Diverse and accurate ensemble learning algorithm. *Proceedings of the 5th International Conference on Intelligent Data Engineering and Automated Learning (LNCS 3177)*, Exeter, UK, 619–625.

Chandra, A., & Yao, X. (2006). Ensemble learning using multi- objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms, 5*, 417–445.

Chang, C.-C., & Lin, C.-J. (2001). LIBSVM: A library for support vector machines. Retrieved from http://www.csie.ntu.edu.tw/~cjlin/libsvm

Chang, E., Zhou, J., Di, S., Huang, C., & Lee, K. (2000). Large vocabulary Mandarin speech recognition with different approaches in modeling tones. In *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China (pp. 983-986). ACM Digital Library.

Chang, G. C., Kang, W. J., Luh, J. J., Cheng, C. K., Lai, J. S., Chen, J. J., & Kuo, T. S. (1996). Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface. *Medical Engineering & Physics, 18*, 529-537.

Chang, S.C., Marek-Sadowska, M., & Hwang, T.T. (1996). Technology mapping for TLU FPGAs based on decomposition of binary decision diagrams. *IEEE Trans on CAD, 15*(10), 1226–1236.

Chellappa, R., Wilson, C.L., & Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, 705–741.

Chen, M.C., & Willson Jr., A.N. (2000). Motion-vector optimization of control grid interpolation and overlapped block motion compensation using iterated dynamic programming. *IEEE Trans. Image Proc., 9*(7), 1145–1157.

Chen, M.-Y., Kundu, A., Zhou, J., & Srihari, S.N. (1992). Off-line handwritten word recognition using hidden Markov model. *Proceedings of the 5th USPS Advanced Technology Conference* (pp. 563–579).

Chen, S., & Lovell, B.C. (2004). Illumination and expression invariant face recognition with one sample image. *Proceedings of the 17th International Conference on Pattern Recognition*. 1, 300–303.

Chen, S., & Srihari, S. (2006). Combining one- and two-dimensional signal recognition approaches to off-line signature verification. *Proceedings of the Document Recognition and Retrieval XIII*, 606701 1- 606701 10.

Chen, X., Zhou, X., & Wong, S.T.C. (in press). Automated segmentation, classification, and tracking cancer cell nuclei in time-lapse microscopy. *IEEE Trans on Biomedical Engineering.*

Cheng, H., Cao, J., Wang, X., & Das, S.K. (2006). Stability-based multi-objective clustering in mobile ad hoc networks. *Proceedings of the 3rd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, Waterloo, Ontario, 191, 27.

Cheung, H.N., Bouzerdoum, A., & Newland, W. (1999). Properties of shunting inhibitory cellular neural networks for colour image enhancements. *Proceedings of the Sixth International Conference on Neural Information Processing*, Vol. 3, 1219–1223.

Cheung, K.-W., Yeung, D.-T., & Chin, R.T. (1998). A Bayesian framework deformable pattern recognition with application to handwritten character recognition. *IEEE Trans Pattern Anal Mach Intell, 20*(12), 1382–1388.

Chevalier, S., Geoffrois, E., Preteux, F., & Lemaltre, M. (2005). A generic 2D approach of handwriting recognition. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 489–493). Seoul, Korea: IEEE Computer Society Press.

Chiang, J.-H. (1998). A hybrid neural model in handwritten word recognition. *Neural Networks, 11*(2), 337–346.

Childers, D. G. (2000). *Speech processing and synthesis toolboxes.* New York: John Wiley & Sons.

Cho, S.-B. (1997). Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Transactions on Neural Networks, 8*(1), 43–53.

Cho, W., Lee, S.-W., & Kim, J.H. (1995). Modeling and recognition of cursive words with hidden Markov models. *Pattern Recognit, 28*(12), 1941–1953.

Chong, S.W. (1996). *Tools for forensic questioned document examination* [unpublished project report]. Nanyang Technological University, School of Applied Science.

Christensen, G.E., Rabbitt, R.D., & Miller, M.I. (1996). Deformable templates using large deformation kinematics. *IEEE Trans Image Processing, 5*(10), 1435–1447.

Chuang, P. (1977). *Machine verification of handwritten signature image*. International Conference on Crime Countermeasures-Sci, 105–109.

Cochran, J.A. (1972). *The analysis of linear integral equations*. McGraw-Hill.

Coetzer, J. (2005). *Off-line signature verification* [unpublished doctoral dissertation]. University of Stellenbosch.

Coetzer, J., Herbst, B., & du Preez, J. (2004). Off-line signature verification using the discrete radon transform and a hidden Markov model. *EURASIP Journal on Applied Signal Processing, 4*, 559–571.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603–619.

Comaniciu, D., Ramesh, V., & Meer, P. (2003). Kernel-based object tracking. *IEEE Transaction on Pattern Analysis Machine Intelligence*, *25*(5), 564–575.

Cootes, T.F., & Taylor, C.J. (1992) Active Shape Models - 'Smart Snakes'. *Proc. British Machine Vision Conference* (pp. 266-275). Springer-Verlag.

Cootes, T.F., Edwards, G.J., & Taylor, C.J. (2001). Active appearance models. *IEEE Transaction on Pattern Analysis and Machine Intelligence, 23*(6), 681–685.

Cootes, T.F., Taylor, C.J., Cooper, D.H., & Graham, J. (1995). Active shape models—Their training and application. *Computer Vision and Image Understanding, 61*(1), 38–59.

Cootes, T.F., Walker, K., & Taylor, C.J. (2000). View-based active appearance models. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 227–232.

Cordeiro, H., & Ribeiro, C.M. (2006). Speaker characterization with MLSF. *Proceedings of the Odyssey 2006: The Speaker and Language Recognition Workshop*, San Juan, Puerto Rico.

Córdoba, R., Menéndez-Pidal, X., & Macías Guasara, J. (1995). Development and improvement of a real-time ASR system for isolated digits in Spanish over the telephone line. In *Proceedings of Eurospeech'95* (pp. 1537-1540). Madrid.

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning, 20*(3). 273–297.

Costa, I.G., Carvalho, F.A.T., & Souto, M.C.P. (2004). Comparative analysis of clustering methods for gene expression time course data. *Genetics and Molecular Biology, 27*(4), 623–631.

Croisier, A., Esteban, D., Levilion, M., & Rizo, V. (1973). *Digital filter for PCM encoded signals*. US Patent No. 3777130.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*, 303–314.

D'haeseleer, P. (2005). How does gene expression clustering work? *Nature Biotechnology, 23*, 1499–1501.

Dalal, N. (2006). *Finding people in images and videos* [unpublished doctoral dissertation]. Institut National Polytechnique Grenoble.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2, 886–893.

Dalal, N., Triggs, B., & Schimid C. (2006). Human detection using oriented histograms of flow and appearance. *Proceedings of the European Conference on Computer Vision*, 2, 428–441.

Darrell, T., & Pentland, A. (1993). Space-time gestures. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 335–340.

Darwen, P.J., & Yao, X. (1997). Speciation as automatic categorical modularization. *IEEE Transactions on Evolutionary Computation, 1*(2), 101–108.

Daubechies, I. (1992). *Ten lectures on wavelets*. SIAM.

Daubert, et al., vs. Merrell Dow Pharmaceuticals. (1993). 509 U.S. 579.

Davis, M. (2005). *Waiting in vain for the paperless office.* Butler Group, ZDNet UK posting. Retrieved July 14, 2007, from http://news.zdnet.co.uk/itmanagement/0,1000000308,39223857,00.htm

Deb, K., Pratap, A., Agarwal, S., & Meyrivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Deng, P., Jaw, L.-J., Wang, J.-H., & Tung, C.-T. (2003). Trace copy forgery detection for handwritten signature verification. *Proceedings of the IEEE International Carnahan Conference on Security Technology.*

Deng, P.S., Liao, H.-Y.M., Ho, C.W., & Tyan, H.-R. (1999). Wavelet based off-line handwritten signature verification. *Computer Vision and Image Understanding, 76*(3), 173–190.

Devijver, P.A., & Kittler, J. (1982). *Pattern recognition: A statistical approach.* New York: Prentice Hall.

Dietterich, T. (2000). Ensemble methods in machine learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, 1–15.

Dimauro, G., Impedovo, S., Pirlo, G., & Salzo, A. (1997). Removing underlines from handwritten text: An experimental investigation. In A.C. Downton, & S. Impedovo (Eds.), *Progress in handwriting recognition* (pp. 497–501). World Scientific Publishing.

Dimauro, G., Impedovo, S., Pirlo, G., & Salzo, A. (1998). An advanced segmentation technique for cursive word recognition. In S.W. Lee (Ed.), *Advances in handwriting recognition* (pp. 255–264). World Scientific Publishing.

Ding, Y., Kimura, F., Miyake, Y., & Shridhar, M. (1999). Evaluation and improvement of slant estimation for handwritten words. *Proceedings of the 5th International Conference on Document Analysis and Recognition* (pp. 753–756). Bangalore, India: IEEE Computer Society Press.

Doak, J. (1992). *An evaluation of feature-selection methods and their application to computer security* (Tech. Rep. CSE-92-18). Davis: University of California, Department of Computer Science.

Doddington, G.R. (1985). Speaker recognition—Identifying people by their voices. *Proceedings of the IEEE*, 73(11), 1651–1664.

Drouhard, J.P., Sabourin, R., & Godbout, M. (1996). A neural network approach to off-line signature verification using directional PDF. *Pattern Recognition, 29*(3), 415–424.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In M.C. Mozer, M.I. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, vol. 9* (p. 155). MIT Press.

Duell, P., Fermin, I., & Yao, X. (2006). Diversity creation in local search for the evolution of neural network ensembles. *Proceedings of the 14th European Symposium on Artificial Neural Networks (ESANN06).*

Duell, P., Fermin, I., & Yao, X. (2006). Speciation techniques in evolved ensembles with negative correlation learning. *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, Vancouver, British Columbia, 3317–3321.

Dunkle, R. (2003). Role of image informatics in accelerating drug discovery and development. *Drug Discovery World, 5,* 75–82.

Dunn, C.E., & Wang, P.S.P. (1992). Character segmentation techniques for handwritten text—A survey. *Proceedings of the 11th International Conference on Pattern Recognition* (pp. 577–580). The Hague, The Netherlands.

Eastwood, B., Jennings, A., & Harvey, A. (1997). A feature based neural network segmenter for handwritten words. In B. Verma, & X. Yao (Eds.). *Proceedings of the 1st International Conference on Computational Intelligence and Multimedia Applications* (pp. 286–290). Gold Coast, Australia.

Edwards, G. J., Taylor, C.J., & Cootes, T.F. (1998). Interpreting face images using active appearance models. *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, 300–305.

Eisen, M.B., Spellman, P., Brown, P., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA, 95*(25), 14863–14868.

Eldridge, M.A., Nimmo-Smith, I.,Wing, A.M., & Totty, R.N. (1984). The variability of selected features in cursive handwriting—categorical measures. *Journal of Forensic Science Society, 24*, 179–219.

Elliman, D.G., & Lancaster, I.T. (1990). A review of segmentation and contextual analysis techniques for text recognition. *Pattern Recognition, 23*(3-4), 337–346.

el-Yacoubi, A., Justino, E.J.R., Sabourin, R., & Bortolozzi, F. (2000). Off-line signature verification using HMMS and cross-validation. *Proceedings of the Ninth IEEE Workshop on Neural Networks for Signal Processing*, 859–868.

Englehart, E., Hudgins, B., & Parker, A. (2001). A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering, 48*, 302-311.

Englehart, E., Hudgins, B., Parker, A., & Stevenson, M. (1999). Classification of the myoelectric signal using time-frequency based representations. *Medical Engineering & Physics, 21*, 431-438.

Ertöz, L., Steinbach, M., & Kumar, V. (2002). A new shared nearest neighbor clustering algorithm and its applications. *Proceedings of the Workshop on Clustering High Dimensional Data and its Applications, 2nd SIAM International Conference on Data Mining*, 105–115.

Estivill-Castro, V. (2002). Why so many clustering algorithms—a position paper. *SIGKDD Explorations, 4*(1), 65–75.

Ezaki, H., Uchida, S., & Sakoe, H. (2005). Dewarping of document image by global optimization. *Proc. ICDAR*, 1, 302–306.

Faceli, K. (2006). *Um framework para análise de agrupamento baseado na combinação multi-objetivo de algoritmos de agrupamento (A framework for cluster analysis based in a multi-objective combination of clustering algorithms)* [doctoral thesis]. São Carlos: Institute of Mathematics and Computer Science, University of São Paulo.

Faceli, K., Carvalho, A., & Souto, M. (2005). Evaluation of the contents of partitions obtained with clustering gene expression data. *Proceedings of the Brazilian Symposium on Bioinformatics, Lecture Notes in Computer Science*, 3594, 65–76.

Faceli, K., Carvalho, A.C.P.L.F., & Souto, M.C.P. (2006). Multi-objective clustering ensemble. *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems*, Auckland, New Zealand, 51.

Fadhel, E.A., &. Bhattacharyya, P. (1999). Application of a steerable wavelet transform using neural network for signature verification. *Pattern Analysis and Applications, 2*, 184–195.

Fahlman, S.E. (1988). *An empirical study of learning speed in backpropagation networks* (Tech. Rep. No. CMU-CS-88-162). Pittsburgh, PA: Computer Science, Carnegie Mellon University.

Falkowski, B.J. (2004). Compact representation of logic functions for lossless compression of grey scale images. *IEEE Proceedings, Computers and Digital Techniques, 151*(3), 221–230).

Falkowski, B.J., & Chang, C.H. (1997). Forward and inverse transformations between Haar spectra and ordered binary decision diagrams of Boolean functions. *IEEE Trans on Computers, 46*(11), 1271–1279.

Fan, X., & Verma, B. (2002). Segmentation vs. non-segmentation based neural techniques for cursive word recognition. *International Journal of Computational Intelligence and Applications, 2*(4), 1–8.

Fang, B., & Tang, Y. (2005). Improved class statistics estimation for sparse data problems in off-line signature verification. *IEEE Transactions on Systems, Man and Cybernetics, 35*(3), 276–286.

Fang, B., Leung, C., Tang, Y., Tse, K., & Wong, Y. (2002). Off-line signature verification with generated training samples. *Proceedings of the IEE Vision, Image and Signal Processing.* Vol. 149, 85–90.

Fang, B., Leung, C., Tang, Y., Tse, K., Kwok, P., & Wong, Y. (2003). Off-line signature verification by tracking of feature and stroke positions. *Pattern Recognition, 36*, 91–101.

Fang, B., Wang, Y., Leung, C., & Tse, K. (2001). Off-line signature verification by the analysis of cursive strokes. *International Journal of Pattern Recognition and Artificial Intelligence, 15*(4), 659–673.

Fanty, M. (1996). *Overview of the CSLU toolkit,* (Tech. Rep. No. CSLU-011-1995). Center for Spoken Language Understanding, Oregon Graduate Institute of Science & Technology.

Fasquel, J., & Bruynooghe, M. (2004). A hybrid opto-electronic method for fast off-line handwritten signature verification. *International Journal on Document Analysis and Recognition, 7*(1), 56–68.

Fawcett, T. (2006). An Introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861-874.

Feal, L. (2000). *Sobre el uso de la sílaba como unidad de síntesis en el español* (Tech. Rep. No. IT-DI-2000-0004). Informatics Department, Universidad de Valladolid.

Feng, Y. (2002). Practicing cell morphology based screen. *European Pharmaceutical Review, 7,* 7–11.

Fern, X.Z., & Brodley, C.E. (2004). Solving cluster ensemble problems by bipartite graph partitioning. *Proceedings of the Twenty-First International Conference on Machine Learning,* Banff, Alberta, 69, 36.

Ferrer, M., Alonso, J., & Travieso, C. (2005). Off-line geometric parameters for automatic signature verification using fixed-point arithmetic. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 993–997.

Fierrez-Aguilar, J., Alonso-Hermira, N., Moreno-Marquez, G., & Ortega-Garcia, J. (2004). An off-line signature verification system based on fusion of local and global information. *Lecture Notes in Computer Science: Biometric Authentication, 3087,* 295–306.

Fitzgibbon, A., Pilu, M., & Fisher, R.B. (1999). Direct least square fitting of ellipse. *Term Analysis and Machine Intelligence, 21,* 476–480.

Fletcher, R. (1981). *Practical methods of optimisation, volume 2: Constrained optimisation*. Chichester: John Wiley and Sons.

Fleuret, F., & Geman, D. (2001). Coarse-to-fine face detection. *International Journal of Computer Vision, 41*(1-2), 85–107.

Forsyth, D., Fleck, M., & Bregler, C. (1996). Finding naked people. *Proceedings of the 1996 European Conference on Computer Vision*, Vol. 2, 592–602.

Forsyth, D.A., & Ponce, J. (2003). *Computer vision: A modern approach*. Pearson Education Press.

Fosler-Lussier, E., Greenberg, S., & Morgan, N. (1999). Incorporating contextual phonetics into automatic speech recognition. In *Proceedings of the ESCA Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition* (pp. 611-614). San Francisco.

Found, B., & Rogers, D. (1995). Contemporary issues in forensic handwriting examination: A discussion of key issues in the wake of the Starzecpyzel decision. *Journal of Forensic Document Examination, 8,* 1–31.

Found, B., & Rogers, D. (1998). A consideration of the theoretical basis of forensic handwriting examination. *Journal of Forensic Document Examination, 4* 2), 109–118.

Fox, S. (2003). Accommodating cells in HTS. *Drug Discovery World, 5,* 21–30.

Franke, K., Zhang, Y.-N., & Koppen, M. (2002). Static signature verification employing a Kosko-neuro-fuzzy approach. *Proceedings of the International Conference on Fuzzy Systems - Advances in Soft Computing,* 185–190.

Fred, A.L.N., & Jain, A.K. (2005). Combining multiple clusterings using evidence accumulation. *IEEE Trans-*

actions on Pattern Analysis and Machine Intelligence, 27(6), 835–850.

Freeman, H. (1961). On the encoding of arbitrary geometric configuration. *Trans Electronics Computers, EC-10*, 264–268.

Fregnac, Y., Monier, C., Chavane, F., Baudot, P., & Graham, L. (2003). Shunting inhibition, a silent step in visual computation. *Journal of Physiology, Paris, 97*, 441–451.

Freund Y., & Schapire R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119-139.

Frias-Martinez, E., Sanchez, A., & Velez, J. (2006). Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence, 19*(6), 693–704.

Fu, A.M.N., Yan, H., & Huang, K. (1997). A curvature angle bend function based method to characterize contour shapes. *Patt Recog, 30*(10), 1661–1671.

Fu, X.J., & Wang, L.P. (2002). *A GA-based novel RBF classifier with class-dependent features*. Proceedings of the 2002 Congress on Evolutionary Computation, 2, 1890–1894.

Fu, X.J., & Wang, L.P. (2003). Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, 33*(3), 399–400.

Fujimoto, Y., Kadota, S., Hayashi, S., Yamamoto, M., Yajima, S., & Yasuda, M. (1976). Recognition of handprinted characters by nonlinear elastic matching. *Proc. ICPR*, 113–118.

Fujisawa, H., Nakano, Y., & Kurino, K. (1992). Segmentation methods for character recognition: From segmentation to document structure analysis. *Proceedings of the IEEE, 80*(7), 1079–1092.

Fukuda, O., Tsuji, T., & Kaneko, M. (1997). An EMG controlled robotic manipulator using neural networks. *Proceedings of the IEEE International Workshop on Robot and Human Communication*, 442-447.

Fukuda, O., Tsuji, T., & Kaneko, M. (1999). An EMG-controlled pointing device using a neural network. *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics, 4*, 63-68.

Fukuda, O., Tsuji, T., Kaneko, M., & Ohtsuka, A. (2003). A human-assisting manipulator teleoperated by EMG signals and arm motions. *IEEE Transactions on Robotics and Automation, 19*, 210-222.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE Transactions Systems, Man, and Cybernetics, SMC-13*(5), 826–834.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks, 2*(3), 183–192.

Gader, P.D., Mohamed, M., & Chiang, J.-H. (1997). Handwritten word recognition with character and inter-character neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, 27*(1), 158–164.

Galata, A., Cohn, A.G., Magee, D., & Hogg, D. (2002). *Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models.* Proceedings of the European Conference on Artificial Intelligence, Lyon, France, 741–745.

Galata, A., Johnson, N., & Hogg, D. (2001). Learning variable-length Markov models of behavior. *Computer Vision and Image Understanding, 81*(3), 398–413.

Ganapathiraju, A., Hamaker, J., Picone, J., & Doddington, G. (2001). Syllable-based large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing, 9*(4), 358-366.

Gang, L., Verma, B., & Kulkarni, S. (2002). Experimental analysis of neural network based feature extractors for cursive handwriting recognition. *Proceedings of the IEEE World Congress on Computational Intelligence* (pp. 2837–2841). Hawaii: IEEE Computer Society Press.

Garcia, C., & Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(11), 1408–1423.

Garris, M.D., & Wilkinson, R.A. (1992). *NIST special database 3 handwritten segmented characters.* National Institute of Standards and Technology.

Gatos, B., Pratikakis, I., Kesidis, A.L., & Perantonis, S.J. (2006). Efficient off-line cursive handwriting word recognition. *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition*.

Gavrila, D.M., & Philomin, V. (1999). Real-time object detection for "smart" vehicles. *Proceedings of the International Conference on Computer Vision*, 87–93.

Gavrila, D.M., Giebel, J., & Munder, S. (2004). Vision-based pedestrian detection: The protector system. *Proceedings of the IEEE Intelligent Vehicles Symposium*, 13–18.

Gavrilla, D. (1999). Visual analysis of human movement: A survey. *Computer Vision and Image understanding, 73*(1), 82–98.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation, 4*(1), 1–58.

Gilad-Bachrach, R., Navot, A., & Tishby, N. (2004). Margin based feature selection—theory and algorithms. *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada.

Gilloux, M. (1993). Research into the new generation of character and mailing address recognition systems at the French post office research centre. *Pattern Recognition Letters, 14*(4), 267–276.

Glasbey, C.A., & Mardia, K.V. (1998). A review of image-warping methods. *J Applied Statistics, 25*(2), 155–171.

Goldberg, D.E. (1989). *Genetic algorithms in search optimization, and machine learning.* Reading, MA: Addison-Wesley.

Golomb, B., Lawrence, D., & Sejnowski, T. (1991). Sexnet: A neural network identifies sex from human faces. *Advances in Neural Information Processing Systems*, 572–577.

Golub, T.R., et al. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science, 286*(5439), 531–537.

Gonzalez, R. & Woods, R. (2002). In *Digital Image Processing, 2nd edition*. Upper Saddle River, NJ: Prentice Hall.

Gonzalez, R.C., & Woods, R.E. (1993). *Digital image processing.* 2nd ed. Adisson-Wesley.

Goodman, D.J., & Carey, M.J. (1977). Nine digital filters for decimation and interpolation. *IEEE Trans on Acoustics, Speech and Signal Processing, 25*(2), 121–126.

Govindaraju, V., Srihari, S.N., & Shin, Y.-C. (1999). Use of handwriting recognition features in handwriting identification. *Proceedings of the 9th Biennial Conference of the International Graphonomics Society*, (pp. 73–78). Singapore.

Granlund, G. (2003). *Organization of architectures for cognitive vision systems.* Proceedings of the Workshop on Cognitive Vision, Schloss Dagstuhl, Germany.

Greene, D., & Cunningham, P. (2006). *Efficient ensemble methods for document clustering* (Tech. Rep. TCD-CS-2006-48). Dublin: Trinity College, Department of Computer Science.

Greene, D., Tsymbal, A., Bolshakova, N., & Cunningham, P. (2004). Ensemble clustering in medical diagnostics. *Proceedings of the 17th IEEE Symposium on Computer-Based Medical Systems*, 576–581).

Griess, F., & Jain, A. (2002). On-line signature verification. *Pattern Recognition, 35*, 2963-2972.

Groszschaedl, J., Kumar, S.S., & Paar, C. (2004). Architectural support for arithmetic in optimal extension fields. *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 111–124.

Guha, P., Mukerjee, A., & Venkatesh, K.S. (2005). *Efficient occlusion handling for multiple agent tracking with surveillance event primitives.* Proceedings of the Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Beijing, 49–56.

Guillevic, D., & Suen, C.Y. (1994). Cursive script recognition: A sentence level recognition scheme. *Proceedings of the 4th International Workshop on the Frontiers of Handwriting Recognition* (pp. 216–223).

Günter, S., & Bunke, H. (2002). Creation of classifier ensembles for handwritten word recognition using feature selection algorithms. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 183–188.

Günter, S., & Bunke, H. (2004). Feature selection algorithms for the generation of multiple classier systems and their application to handwritten word recognition. *Pattern Recognition Letters*, *25*(11), 1323–1336.

Günter, S., & Bunke, H. (2005). Off-line cursive handwriting recognition using multiple classifier systems—On the influence of vocabulary, ensemble, and training set size. *Optics and Lasers in Engineering*, *43*(3-5), 437–454.

Guo, J., Doermann, D., & Rosenfeld, A. (2000). Off-line skilled forgery detection using stroke and sub-stroke properties. *Proceedings of the 15th International Conference on Pattern Recognition*, 2355–2358.

Guo, Z., & Hall, R.W. (1989). Parallel thinning with two-subiteration algorithms. *Comm. ACM*, *32* 3), 359–373.

Gupta, P., Ravit, S., Raghunathan, A., & Jha, N.K. (2005). Efficient fingerprint-based user authentication for embedded systems. *Proceedings of the Design Automation Conference*, 244–247.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Guyon, I., Gunn, S.R., Ben-Hur, A., & Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. *Proceedings of Advances in Neural Information Processing Systems (NIPS 2004)*, Vancouver, Canada.

Ha, T.M., & Bunke, H. (1997). Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans Pattern Anal Mach Intell*, *19*(5), 535–539.

Halloui, K., Likforman-Sulem, L., & Sigelle, M. (2002). A comparative study between decision fusion and data fusion in Markovian printed character recognition. *Proc. ICPR*, 3 of 4, 147–150.

Handl, J. (2006). *Multiobjective approaches to the data-driven analysis of biological systems* [doctoral thesis]. Manchester, UK: University of Manchester, School of Chemistry.

Handl, J., & Knowles, J. (2005). Computational cluster validation in postgenomic data analysis. *Bioinformatics*, *21*(15), 3201–3212.

Handl, J., & Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, *11*(1), 56–76.

Hanmandlu, M., Murali, K.R.M., Chakraborty, S., Goyal, S., & Choudhury, D.R. (2003). Unconstrained handwritten character recognition based on fuzzy logic. *Pattern Recognition*, *36*(3), 603–623.

Hanmandlu, M., Yusof, M., & Madasu, V. (2005). Off-line signature verification and forgery detection using fuzzy modeling. *Pattern Recognition*, *38*(3), 341–356.

Hansen, L.K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Haritaoglu, I., Harwood, D., & Davis, L. (2000). W4: Real time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 809–830.

Harmut, R., Pfitzinger, S. B., & Heid, S. (1996) Syllable detection in read and spontaneous speech. In *Proceedings of 4th International Conference on Spoken Language Processing*, Philadelphia (pp. 1261-1264). Washington, DC: IEEE Computer Society.

Harrison, W.R. (1958). *Suspect documents: Their scientific examination*. London: Sweet & Maxwell Ltd.

Harrison, W.R. (1981). *Suspect documents, their scientific examinations.* Illinois: Nelson-Hall.

Hastie, T., Simard, P.Y., & Säckinger, E. (1995). Learning prototype models for tangent distance. *Advances in Neural Information Processing Systems, 7,* 999–1006.

Hastie, T.J., & Tibshirani, R. (1994). *Handwritten digit recognition via deformable prototypes.* AT&T Bell Laboratories Technical Report.

Hattori, T., Watanabe, Y., Sanada, H., & Tezuka, Y. (1983). Absorption of local variations in handwritten character by an elastic transformation using vector field [in Japanese]. *IEICE Trans, J66-D*(6), 645–652.

Hauenstein, A. (1996). *The syllable re-revisited* (Tech. Rep. No. tr-96-035). Munich, Germany: Siemens AG, Corporate Research and Development.

Hautaniemi, S., et al. (2003). Analysis and visualization of gene expression microarray data in human cancer using self-organizing maps. *Machine Learning, 52*(1-2), 45–66.

Haykin, S. (1999). *Neural networks—A comprehensive foundation*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Herbrich, R., & Weston, J. (1999). *Adaptive margin support vector machines for classification learning.* Proceedings of the Ninth International Conference on Artificial Neural Networks, 880–885.

Hilton, O. (1992). Signatures review and a new view. *Journal of Forensic Sciences, 37*(1), 125–129.

Hilton, O. (1993). *Scientific examination of questioned documents.* Florida: CRC Hall.

Hiraiwa, A., Shimohara, K., & Tokunaga, Y. (1989). EMG pattern analysis and classification by neural network. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 3,* 1113-1115.

Hiraoka, Y., & Haraguchi, T. (1996). Fluoresence imaging of mammalian living cells. *Chromosome Res, 4,* 173–176.

Hjelmas, E., & Low, B.K. (2001). Face detection: A survey. *Computer Vision and Image Understanding, 83*(3), 236–274.

Hoiem, D., Efros, A.A., & Hebert, M. (2006). Putting objects in perspective. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* New York, 2137–2144.

Holcombe, G., & Leedham, C.G. (1995). An experimental imaging environment for examination of questioned documents. *Proceedings of the 7th International Graphonomics Conference* (pp. 80–81). London, Ontario.

Horn, J., Goldberg, D.E., & Deb, K. (1994). Implicit niching in a learning classifier system: Nature's way. *Evolutionary Computation, 2*(1), 37–66.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366.

Howe, N.R., Rath, T.M., & Manmatha, R. (2005). Boosted decision trees for word recognition in handwritten document retrieval. *Proceedings of the 28th Annual SIGIR Conference on Research and Development in Information Retrieval* (pp. 377–383).

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Network, 13,* 415–425.

Hsu, C.W., Chang, C.C., & Lin, C.J. (2003). *A practical guide to support vector classification*. Taipei, Taiwan: National Taiwan University, Department of Computer Science and Information Engineering.

Hu, J., & Yan, H. (1998). Structural primitive extraction and coding for handwritten numeral recognition. *Patt Recog, 31*(5), 493–509.

Hu, X. (2006). Gene-miner: Integration of cluster ensemble and text mining for comprehensive gene expression analysis. *International Journal of Bioinformatics Research and Application, 2*(3), 325–338.

Hu, X., Yoo, I., Zhang, X., Nanavati, P., & Das, D. (2005). Wavelet transformation and cluster ensemble for gene

expression analysis. *International Journal of Bioinformatics Research and Applications, 1*(4), 447–460.

Hu, Z., Schalkwyk, J., Barnard, E., & Cole, R. (1996). Speech recognition using syllable-like units. In *Proceedings of the International Conference on Spoken Language Processing,* Philadelphia, PA, (vol. 2, pp. 1117-1120). Washington DC: IEEE Computer Society.

Hua, S., & Sun, Z. (2001). A novel method of protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology, 308*, 397–407.

Huang, K., & Yan, H. (1997). Off-line signature verification based on geometric feature extraction and neural network classification. *Pattern Recognition, 30*(1), 9–171.

Huang, K., & Yan, H. (2002). Off-line signature verification using structural feature correspondence. *Pattern Recognition, 35*, 2467–2477.

Hubel, D.H., & Wiesel, T.N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology, 28*, 229–289.

Huber, R.A., & Headrick, A.M. (1999). *Handwriting identification: Facts and fundamentals*. CRC Press.

Hudgins, B., Parker, P., & Scott, R.N. (1993). A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering, 40*, 82-94.

Hussein, S. E., & Granat, M. H. (2002). Intention detection using a neuro-fuzzy EMG classifier. *IEEE Engineering in Medicine and Biology Magazine, 21*(6), 123-129.

I.M. Lab. *Asian face image database*. Retrieved from http://nova.postech.ac.kr/

Igarza, J., Hernaez, I., & Goirizelaia, I. (2005). Static signature recognition based on left-to-right hidden Markov models. *Journal of Electronic Imaging, 14*(4).

Islam, M.M., Yao, X., & Murase, K. (2003). A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks, 14*(4), 820–834.

Ismail, M.A., & Gad, S. (2000). Off-line Arabic signature recognition and verification. *Pattern Recognition, 33*(10), 1727–1740.

Isomichi, Y., & Ogawa, T. (1975). Pattern matching by using dynamic programming [in Japanese]. *J Inf Process Soc Japan, 16*(1), 15–22.

Izui, Y., Harashima, H., & Miyagawa, H. (1985). Handprinted Chinese characters recognition by hierarchical modification of dictionary [in Japanese]. *IEICE Trans, J68-D*(3), 361–368.

Jackson, L. B. (1996). *Digital filters and signal processing.* New York: Kluwer Academic Publishers.

Jain, A., & Dubes, R. (1988). *Algorithms for clustering dat*a. Prentice Hall.

Jain, A.K., & Karu, K. (1996). Learning texture discrimination masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 18*(2), 195–205.

Jain, A.K., & Zongker, D. (1997). Representation and recognition of handwritten digits using deformable templates. *IEEE Trans Pattern Anal Mach Intell, 19*(12), 1386–1391.

Jain, A.K., Myrthy, M.N., & Flynn, P.J. (1999). Data clustering: A survey. *ACM Computing Survey, 31*(3), 264–323.

Jain, A.K., Zhong, Y., & Dubuisson-Jolly, M.-P. (1998). Deformable template models: A review. *Signal Processing, 71*(2), 109–129.

Jin, B., & Zhang, Y.Q. (2005). Support vector machines with evolutionary feature weights optimization for biomedical data classification. *Proceedings of the Soft Computing for Real World Applications, Annual Meeting of the North America Fuzzy Information Processing Society (NAFIPS05),* Ann Arbor, Michigan, 177–180).

John, G.H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning,* Portland, Oregon, 367–370.

Johnson, N., & Hogg, D. (1995). *Learning the distribution of object trajectories for event recognition*. Proceedings of the 6th British Conference on Machine vision, Birmingham, 2, 583–592.

Johnson, N., Galata, A., & Hogg, D. (1998). *The acquisition and use of interaction behavior models*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California, 866–871.

Jones, R., Downey, S., & Mason, J. (1999). Continuous speech recognition using syllables. In *Proceedings of Eurospeech'96* (vol. 3, pp. 1171-1174). Rhodes, Greece.

Jouve, P., & Eric, N.N.L. (2003). A new method for combining partitions, applications for cluster ensembles in KDD. *Proceedings of Parallel and Distributed Computing for Machine Learning*, in conjunction with *the 14th European Conference on Machine Learning* and *7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Cavtat-Dubrovnik, Croatia.

Juliato, M., Araujo, G., Lopez, J., & Dahab, R. (2005). A custom instruction approach for hardware and software implementations of finite field arithmetic over F/sub 2163/ using Gaussian normal bases. *Proceedings of the IEEE International Conference on Field-Programmable Technology*, 5–12.

Justino, E. (2001). *O grafismo e os modelos escondidos de Markov na verificação automática de assinaturas* [unpublished doctoral dissertation]. Brazil: PUC-PR.

Justino, E., Bortolozzi, F., & Sabourin, R. (2001). Off-line signature verification using HMM for random, simple and skilled forgeries. *Proceedings of the International Conference on Document Analysis and Recognition*, 105–110.

Justino, E., Bortolozzi, F., & Sabourin, R. (2005). A comparison of SVM and HMM classifiers in the off-line signature verification. *Pattern Recognition Letters*, *26*(9), 1377–1385.

Justino, E.J.R., Bortolozzi, F., & Sabourin, R. (2001). Offline signature verification using HMM for random, simple and skilled forgeries. *Proceedings of the Sixth IEEE International Conference on Pattern Recognition*, 450–453.

Kalera, M., Srihari, S., & Xu, A. (2004). Off-line signature verification and identification using distance statistics. *International Journal of Pattern Recognition and Artificial Intelligence*, *18*(7), 1339–1360.

Kam, M., Fielding, G., & Conn, R. (1997). Writer identification by professional document examiners. *Journal of Forensic Sciences*, *42*(5), 778–786.

Kam, M., Fielding, G., & Conn, R. (1998). Effects of monetary incentives on performance in document examination proficiency tests. *Journal of Forensic Science*, *43*(5), 1000–1004.

Kam, M., Gummadidala, K., Fielding, G., & Conn, R. (2001). Signature authentication by forensic document examiners. *Journal of Forensic Science*, *46*(6), 884–888.

Kam, M., Wetstein, J., & Conn, R. (1994). Proficiency of professional document examiners in writer identification. *Journal of Forensic Sciences*, *39*(1), 5–14.

Kang, W. J., Shiu, J. R., Cheng, C. K., Lai, J. S., Tsao, H. W., & Kuo, T. S. (1995). The application of cepstral coefficients and maximum likelihood method in EMG pattern recognition. *IEEE Transactions on Biomedical Engineering*, *42*, 777-785.

Kapp, M.N., de Almendra Freitas, C., & Sabourin, R. (2007). Methodology for the design of NN-based month-word recognizers written on Brazilian bank checks. *Image Vision Computing*, *25*(1), 40–49.

Kasturi, J., & Acharya, R. (2005). Clustering of diverse genomic data using information fusion. *Bioinformatics*, *21*(4), 423–429.

Kato, T., Omachi, S., & Aso, H. (2000). Precise hand-printed character recognition using elastic models via nonlinear transformation. *Proc. ICPR*, 2, 364–367.

Kecman, V. (2001). *Learning and soft computing, support vector machines, neural networks and fuzzy logic models*. Cambridge, MA: MIT Press.

Kelly, M. F., Parker, P. A., & Scott, R. N. (1990). The application of neural networks to myoelectric signal analysis: A preliminary study. *IEEE Transactions on Biomedical Engineering, 37,* 221-230.

Keysers, D., & Unger, W. (2003). Elastic image matching is NP-complete. *Pattern Recog Lett, 24*(1)-3, 445–453.

Keysers, D., Dahmen, J., Theiner, T., & Ney, H. (2000). Experiments with an extended tangent distance. *Proc. ICPR*, 2 of 4, 38–42.

Keysers, D., Gollan, C., & Ney, H. (2004). Local context in non-linear deformation models for handwritten character recognition. *Proc. ICPR*, 4 of 4, 511–514.

Keysers, D., Macherey, W., Ney, H., & Dahmen, J. (2004). Adaptation in statistical pattern recognition using tangent vectors. *IEEE Trans Pattern Anal Mach Intell, 26*(2), 269–274.

Kholmatov, A. (2003). *Biometric identity verification using on-line and off-line signature verification* [unpublished master's dissertation]. Sabanci University.

Kianzad, V., et al. (2005). An architectural level design methodology for embedded face detection. *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, 136–141.

Kim, G., Govindaraju, V., & Srihari, S.N. (1999). Architecture for handwritten text recognition systems. In S.W. Lee (Ed.), *Advances in handwriting recognition* (pp. 163–182). World Scientific Publishing.

Kimura, F., Shridhar, M., & Chen, Z. (1993). Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. *Proceedings of the 2nd International Conference on Document Analysis and Recognition* (pp. 18–22). Tsukuba, Japan: IEEE Computer Society Press.

Kimura, F., Yoshimura, M., Miyake, Y., & Ichikawa, M. (1970). Unconstrainedly handprinted "KATAKANA"' character recognition by a stroke structure analysis method [in Japanese]. *Trans IEICE, J62-D*(1), 16–23.

King, S., Taylor, P., Frankel, J., & Richmond, K. (2000). Speech recognition via phonetically featured syllables. *PHONUS, 5,* 15-34.

Kira, K., & Rendell, L.A. (1992). The feature selection problem: Traditional methods and a new algorithm. *Proceedings of 10th National Conference on Artificial Intelligence*, Park, California, 129–134.

Kirschning-Albers, I. (1998). *Automatic speech recognition with the parallel cascade Neural Network*. Unpublished doctoral dissertation, University of Tokushima, Japan.

Kita, K., Morimoto, T., & Sagayama, S. (1993). LR parsing with a category reachability test applied to speech recognition. *IEICE Trans. Information and Systems, E76-D*(1), 23-28.

Kleinberg, J. (2002). An impossibility theorem for clustering. *Advances in Neural Information Processing Systems, 15*, 446–453.

Kobayashi, T., Nakamura, K., Muramatsu, H., Sugiyama, T., & Abe, K. (2001). Handwritten numeral recognition using flexible matching based on learning of stroke statistics. Proc. ICDAR, 612–616.

Koerich, A.L., Britto, A., Oliveira, L.E.S., & Sabourin, R. (2006). Fusing high- and low-level features for handwritten word recognition. *Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition*.

Koerich, A.L., Sabourin, R., & Suen, C.Y. (2003). Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis and Applications, 6*(2), 97–121.

Koerich, A.L., Sabourin, R., & Suen, C.Y. (2005). Recognition and verification of unconstrained handwritten words. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(10), 1509–1522.

Kohavi, R. & John, G. H. (1998) The wrappers approach. In Huan Liu & Hiroshi Motoda (Eds.), *Feature extraction, construction and selection: A data mining perspective* (pp. 35-46). Springer.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *Proceedings of the 13th International Conference on Machine Learning (ML),* Bari, Italy, 284–292.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. *Proceedings of the European Conference on Machine Learning (ECML_94)*, Berlin, Heidelberg, 171–182.

Korkmaz, E.E., Du, J., Alhajj, R., & Barker, K. (2006). Combining advantages of new chromosome representation scheme and multi-objective genetic algorithms for better clustering. *Intelligent Data Analysis, 10*(2), 163–82.

Kosko, B. (1992). *Neural networks for signal processing.* Englewood Cliffs, NJ: Prentice Hall.

Kottathra, K., & Attikiouzel, Y. (1996). A novel multicriteria optimization algorithm for the structure determination of multilayer feedforward neural networks. *Journal of Network and Computer Applications, 19*, 135–147.

Kressel, U. (1999). Pair-wise classification and support vector machines. *Advances in kernel methods: Support vector learning* (pp. 255–268). Cambridge, MA: MIT Press.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems*, Vol. 7 (pp. 231–238). Cambridge, MA: MIT Press.

Kroner, S., & Moratz, R. (1996). Capacity of structured multilayer networks with shared weights. *Proceedings of the International Conference on Artificial Neural Networks*, Bochum, Germany, Vol. 1112, 543–550 .

Kumar, S., & Hebert, M. (2005). A hierarchical field framework for unified context-based classification. *Proceedings of the International Conference on Computer Vision*, Beijing, China, 1284–1291.

Kuncheva, L.I., Hadjitodorov, S.T., & Todorova, L.P. (2006). Experimental comparison of cluster ensemble methods. *Proceedings of FUSION 2006*, 105–115.

Kung, S., Mak, M., & Lin, S. (2004). *Biometric authentication, a machine learning approach*. Prentice Hall.

Kunzel, H.J. (1994). Current approaches to forensic speaker recognition. *Proceedings of the ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, *48*, 135–141.

Kuo, S., & Agazzi, O.E. (1994). Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *IEEE Trans Pattern Anal Mach Intell, 16*(8), 842–848.

Kupinski, M.A., & Anastasio, M.A. (1999). Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging, 18*(8), 675–685.

Lades, M., et al. (1993). Distortion invariant object recognition in the dynamic link architecture. *Computers, IEEE Transactions, 42*(3), 300–311.

Lamounier, E., Soares, A., Andrade, A., & Carrijo, R. (2002). A virtual prosthesis control based on neural networks for EMG pattern classification. *Proceedings of the* 6^{th} *IASTED International Conference on Artificial Intelligence and Soft Computing*, 456-461.

Langeley, P., & Sage, S. (1994). *Oblivious decision trees and abstract cases.* Working Notes of the AAAI94 Workshop on Case-Based Reasoning, Seattle, Washington, 113–117.

Langeley, P., & Sage, S. (1994). Induction of selective Bayesian classifiers. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, Washington, 399–406.

Lapsley, P., Bier, J., Shoham, A., & Lee, E. (1997). DSP processor fundamentals. New York: IEEE Press.

Lau, K., Yuen, P., & Tang, Y. (2005). Universal writing model for recovery of writing sequence of static handwriting images. *International Journal of Pattern Recognition and Artificial Intelligence, 19*, 603–630.

Law, M., Topchy, A., & Jain, A.K. (2004). Multiobjective data clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2, 424–430.

Leclerc, F., & Plamondon, R. (1994). Automatic signature verification, the state of the art—1989-1993. *International*

Journal of Pattern Recognition and Artificial Intelligence, 8(3), 643–660.

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), *Connectionism in perspective*. Zurich, Switzerland: Elsevier.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. of the IEEE, 86*(11), 2278–2324.

LeCun, Y., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541–551.

Lee, E. (1988). Programmable DSP architectures: Part I. *IEEE Transactions on Acoustics, Speech and Signal Processing Magazine*, 4–19.

Lee, E. (1989). Programmable DSP architectures: Part II. *IEEE Transactions on Acoustics, Speech and Signal Processing Magazine*, 4–14.

Lee, J., & Lee, D. (2005). An improved cluster labeling method for support vector clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(3), 461–464.

Lee, L., & Coelho, S. (2005). A simple and efficient method for global handwritten word recognition applied to Brazilian bankchecks. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 950–955). Seoul, Korea: IEEE Computer Society Press.

Lee, L., Lizárraga, M., Gomes, N., & Koerich, A. (1997). A prototype for Brazilian bankcheck recognition. *International Journal of Pattern Recognition and Artificial Intelligence, 11*(4), 549–569.

Lee, S., & Pan, J.C. (1992). Offline tracing and representation of signatures. *IEEE Transactions on Systems, Man and Cybernetics, 22*(4), 755–771.

Lee, S.W. (1996). Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural networks. *IEEE Trans Pattern Anal Mach Intell, 18*(6), 648–652.

Lee, S.-W., & Park, J.-S. (1994). Nonlinear shape normalization methods for the recognition of large-set handwritten characters. *Pattern Recognit, 27*(7), 895–902.

Lee, T., & Ching, P. (1998). A neural network based speech recognition system for isolated Cantonese syllables. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97) Hong Kong* (vol. 4, p. 3269). Washington DC: IEEE Computer Society.

Leedham, C.G., Holcombe, G., & Sagar, V.K. (1995). Image processing tools for the interactive forensic examination of questioned document. *Proceedings of the European Convention on Security and Detection*, (pp. 225-228). Brighton, UK.

Lester, H., & Arridge, S.R. (1999). A survey of hierarchical non-linear medical image registration. *Pattern Recognit, 32*(1), 129–149.

Levin, E., & Pieraccini, R. (1992). Dynamic planar warping for optical character recognition. Proc. ICASSP, III 149–152.

Lewis, D.M., Galloway, D.R., Van Ierssel, M., Rose, J., & Chow, P. (1998). The transmogrifier-2: A 1 million gate rapid-prototyping system. *Proceedings of the IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 6, 188–198.

Liang, J., Doermann, D., & Li, H. (2005). Camera-based analysis of text and documents: A survey. *Int J Doc Anal Recog, 7*, 84–104.

Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. *Proceedings of the International Conference on Image Processing*, 1, 900–903.

Lin, C.-T., & Shou, Y.-W. (2005). Texture classification and representation by CNN based feature extraction. *Proceedings of the 9th International Workshop on Cellular Neural Networks and Their Applications*, 210–213.

Liolios, N., Kavallieratou, E., Fakotakis, N., & Kokkinakis, G. (2002). A new shape transformation approach to handwritten character recognition. *Proc. ICPR*, 1 of 4, 584–587.

Liu, C.-L., & Fujisawa, H. (2005). Classification and learning for character recognition: Comparison of methods and remaining problems. *Proceedings of the International Workshop on Neural Networks and Learning in Document Analysis and Recognition* (pp. 5–7). Seoul, Korea: IEEE Computer Society Press.

Liu, C.L., Nakashima, K., Sako, H., & Fujisawa, H. (2004). Handwritten digit recognition: Investigation of normalization and feature extraction techniques. *Pattern Recognit, 37*(2), 265–279.

Liu, H., & Motoda, H. (1998). Feature extraction, construction, and selection. *A Data Mining Perspective*. Kluwer Academic Publisher.

Liu, Y. (1998). *Negative correlation learning and evolutionary neural network ensembles* [doctoral thesis]. Canberra, Australia: University College, The University of New South Wales, Australian Defence Force Academy.

Liu, Y., & Yao, X. (1998). Negatively correlated neural networks for classification. *Proceedings of the Third International Symposium on Artificial Life and Robotics (AROBIII'98)*, Beppu, Japan, 2, 736–739.

Liu, Y., & Yao, X. (1999). Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, 29*(6), 716–725.

Liu, Y., Ram, S., & Lusch, R. (2005). A unified market segmentation method for generating Pareto optimal sets. *Proceedings of the 15th Workshop on Information Technology and Systems*, Las Vegas, Nevada.

Liu, Y., Yao, X., & Higuchi, T. (2000). Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation, 4*(4), 380–387.

Locard, E. (1936). *Traité de criminalistique*. Lyon: Payoy.

Lourenco, A., & Fred, A. (2005). Ensemble methods in the clustering of string patterns. *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 1, 143–148.

Lovell, B.C., & Chen, S. (2005). Robust face recognition for data mining. In J. Wang (ed.), *Encyclopedia of Data Warehousing and Mining,* (965–972). Hershey, PA: Idea Group.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.

Lu, Y. (1995). Machine printed character segmentation—An overview. *Pattern Recognition, 28*(1), 67–80.

Lu, Y., & Shridhar, M. (1996). Character segmentation in handwritten words—An overview. *Pattern Recognition, 29*(1), 77–96.

Łuba, T. (1995). Decomposition of multiple-valued functions. *Proceedings of the 25th International Symposium on Multiple-Valued Logic*, Bloomington, Indiana, (pp. 256–261).

Łuba, T., & Selvaraj, H. (1995). A general approach to Boolean function decomposition and its applications in FPGA-based synthesis. *VLSI Design, Special Issue on Decompositions in VLSI Design, 3*(3-4), 289–300.

Łuba, T., Selvaraj, H., Nowicka, M., & Kraśniewski, A. (1995). Balanced multilevel decomposition and its applications in FPGA-based synthesis. In G. Saucier, & A. Mignotte (Eds.), *Logic and architecture synthesis.* Chapman & Hall.

Lusted, H. S., & Knapp, R. B. (1996, October). Controlling computers with neural signals. *Scientific American,* 82-87.

Lv, H., Wang, W., Wang, C., & Zhuo, Q. (2005). Offline Chinese signature verification based on support vector machines. *Pattern Recognition Letters, 26*(15), 2390–2399.

Madasu, V. (2006). *Automatic bank check processing and authentication using signature verification* [unpublished doctoral dissertation]. Australia: University of Queensland.

Madasu, V., Hanmandlu, M., & Madasu, S. (2003). Neuro-fuzzy approaches to signature verification. *Proceedings of the 2nd National Conference on Document Analysis and Recognition.*

Madhvanath, S., Kleinberg, E., & Govindaraju, V. (1999). Holistic verification of handwritten phrases. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(12), 1344–1356.

Makhoul, J., Schwaltz, R., Lapre, C., & Bazzi, I. (1998). A script-independent methodology for optical character recognition. *Pattern Recognit, 31*(9), 1285–1294.

Malaviya, A., & Klette, R. (1996). A fuzzy syntactic method for on-line handwriting recognition. In P. Pernet, P. Wang, & A. Rosenfeld, Eds., *Advance in structural and syntactical pattern recognition*. Springer.

Mandler, J. (1992). How to build a baby: II. Conceptual primitives. *Psychological Review, 99*(4), 587–604.

Manevitz, L.M., & Yousef, M. (2001). One-class {SVM}s for document classification. *Journal of Machine Learning Research, 2*. 139–154.

Marinai, S., Gori, M., & Soda, G. (2005). Artificial neural networks for document analysis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(1), 23–35.

Martin, G.L., Rashid, M., & Pittman, J.A. (1993). Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps. *International Journal on Pattern Recognition and Artificial Intelligence, 7*(4), 831–847.

Martinez, L., Travieso, C., Alonso, J., & Ferrer, M. (2004). Parameterization of a forgery handwritten signature verification system using SVM. *Proceedings of the International Carnahan Conference on Security Technology*, 193–196.

Materka, A., & Strzelecki, M. (1998). *Texture analysis methods—A review* (Tech. Rep. No. COST B11). Brussels: Institute of Electronics, Technical University of Lodz.

Matsui, T., & Furui, S. (1994). Comparison of text-independent speaker recognition methods using VQ-distortion and discrete/continuous HMMs. *IEEE Transactions on Speech and Audio Processing, 2*, 456–459.

Matsumoto, N., Uchida, S., & Sakoe, H. (2004). Prototype setting for elastic matching-based image pattern recognition. *Proc. ICPR*, 1 of 4, 224–227.

McCready, R. (2000). Real-time face detection on a configurable hardware system. *Proceedings of the Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications*, 157–162.

McKay, R.I.B., & Abbass, H.A. (2001). Anti-correlation: A diversity promoting mechanisms in ensemble learning. *The Australian Journal of Intelligent Information Processing Systems (AJIIPS), 7*(3/4), 139–149.

Meguro, S., & Umeda, M. (1978). *An extraction of shape derivations in handwritten characters by hierarchical pattern matching* [in Japanese]. Tech. Rep. of IEICE Japan, PRL77-70.

Meneido, H., & Neto, J. (2000). Combination of acoustic models in continuous speech recognition hybrid systems. In *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China (vol. 9, pp. 1000-1029). ACM Digital Library.

Meneido, H., Neto, P., & Almeida, L. (1999). Syllable onset detection applied to the Portuguese language. In *Proceedings of Eurospeech'99* (p. 81). Budapest, Hungry.

Mercer, J. (1909). Functions of positive and negative type, and their connection with the theory of integral equations. *Transactions of the Royal Society of London, 209(A).*

Meyer-Baese, U. (2004). *Digital signal processing with field programmable gate arrays. Second edition*. Berlin: Springer Verlag.

Mighell, D., Wilkinson, T., & Goodman, J. (1989). Backpropagation and its application to handwritten signature verification. In Touretzky, D. (Ed.), *Advances in neural information processing systems I* (pp. 340–347). San Mateo, CA: Morgan Kaufmann.

Mikolajczyk, K., Schmid, C., & Zisserman, A. (2004). Human detection based on a probabilistic assembly of robust part detectors. *Proceedings of the European Conference on Computer Vision, 1*, 69–82.

Mitchell, S.J., & Silver, R.A. (2003). Shunting inhibition modulates neuronal gain during synaptic excitation. *Neuron, 38*(3), 433–445.

Mitra, S., & Banka, H. (2006). Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition, 39*(12), 2464–2477.

Mizukami, Y. (1998). A handwritten Chinese character recognition system using hierarchical displacement extraction based on directional features. *Pattern Recog Lett, 19*(7), 595–604.

Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object representation. *Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence, 19,* 696–710.

Moghaddam, B., & Yang, M.-H. (2002). Learning gender with support faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 707–711.

Mohamed, M., & Gader, P. (1996). Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. IE*EE Trans Pattern Anal Mach Intell, 18*(5), 548–554.

Mohan, A., Papageorgiou, C., & Poggio, T. (2001). Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(4), 349–361.

Moktarian, F., & Mackworth, A.K. (1992). A theory of multiscale curvature-based shape representation for planer curvature angles. *IEEE Trans Pattern Analysis Mach Intell, 14(8),* 789–805.

Montero, H., & Neto, J. (2000). Combination of acoustic models in continuous speech recognition hybrid systems. In *Proceedings of Eurospeech'99* (pp. 2099-2102). Budapest.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning, 52*(1-2), 91–118.

Moon, H., Chellappa, R., & Rosenfeld, A. (2002). Optimal edge-based shape detection. *Proceedings of the IEEE Transactions on Image Processing, 11,* 1209–1227.

Moore, A.W., & Lee, M.S. (1994). Efficient algorithms for minimizing cross validation error. *Proceedings of the Eleventh International Conference on Machine Learning,* New Brunswick, New Jersey, 190–198.

Moore, R.K. (1979). A dynamic programming algorithm for the distance between two finite areas. *IEEE Trans Pattern Anal Mach Intell, PAMI-1*(1), 86–88.

Mori, S., Suen, C.Y., & Yamamoto, K. (1992). Historical review of OCR research and development. *Proc. IEEE, 80*(7), 1029–1058.

Mori, S., Yamamoto, K., & Yasuda, M. (1984). Research on machine recognition of handprinted characters. *IEEE Trans Pattern Anal Mach Intell, PAMI-6*(4), 386–405.

Munder, S., & Gavrila, D.M. (2006). An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 28*(11), 1863–1868.

Munive, N., Vargas, A., Serridge, B., Cervantes, O., & Kirschnning, I. (1998). Entrenamiento de un reconocedor fonético de digitos para el español mexicano usando el CSLU toolkit. *Revista de Computación y Sistemas, 3*(2), 98-104.

Munson, A., & Caruana, R. (2006). *Cluster ensembles for network anomaly detection* (Tech. Rep. 2006-2047). Ithaca, NY: Cornell University, Department of Computer Science.

Murase, H., & Nayar, S.K. (1993). Learning and recognition of 3D objects from appearance. *Proceedings of the IEEE Workshop on Qualitative Vision,* 39–50.

Murphy, K.P., Torralba, A.B., & Freeman, W.T. (2003). Graphical model for recognizing scenes and objects. *Proceedings of the Neural Information and Processing Systems,* Vancouver, Canada.

Murshed, N., Bortolozzi, F., & Sabourin, R. (1995). Off-line signature verification using fuzzy artmap neural networks. *Proceedings of the IEEE International Conference on Neural Networks,* 2179–2184.

Murshed, N.A., Sabourin, R., & Bortolozzi, F. (1997). A cognitive approach to offline signature verification. In

H. Bunke & P.S.P. Wang (Eds.), *Automatic bankcheck processing* (pp. 339–364). Singapore: World Scientific Publishing.

Murty, K.S R., & Yegnanarayana, B. (2006). Combining evidence from residual phase and MFCC features for speaker recognition. *IEEE Signal Processing Letters, 13*(1), 52–55.

Nagel, R.N., & Rosenfeld, A. (1977). Computer recognition of freehand forgeries. *IEEE Transactions on Computers, 26*(9), 895–905.

Nakagawa, M., Yanagida, T., & Nagasaki, T. (1999). An off-line character recognition method employing model-dependent pattern normalization by an elastic membrane model. *Proc. ICDAR,* 495–498.

Nakano, Y., Nakata, K., Uchikura, Y., & Nakajima, A. (1973). Improvement of Chinese character recognition using projection profiles. *Proceedings of the International Joint Conference on Pattern Recognition*, 172–178.

Nakata, K., Nakano, Y., & Uchikura, Y. (1972). Recognition of Chinese characters. *Proceedings of the Conference on Machine Perception of Patterns and Pictures*, 45–52.

Narayanan, E.K.A. (2005). *Intelligent bioinformatics: The application of artificial Intelligence Techniques to bioinformatics problems.* John Wiley & Sons.

Navernioukk, I. (2005). *Multiobjective graph clustering with variable neighbourhood descent* [master's thesis]. University of British Columbia, Canada.

Nel, E.-M., du Preez, J., & Herbst, B. (2005). Estimating the pen trajectories of static signatures using hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27,* 1733–1746.

Nemcek, W.F., & Lin, W.C. (1974). Experimental investigation of automatic signature verification. *IEEE Transactions on Systems, Man and Cybernetics, 4,* 121–126.

Newman, D.J., Hettich, S., Blake, C.L., & Merz, C.J. (1998). *UCI repository of machine learning databases.* Irvine, CA: University of California, Department of Information and Computer Science. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Nicchiotti, G., Scagliola, C., & Rimassa, S. (2000). A simple and effective cursive word segmentation method. *Proceedings of the 7th International Workshop on the Frontiers of Handwriting Recognition* (pp. 499–504).

Nikkilä, J., Törönen, P., Kaski, S., Venna, J., Castrén, E., & Wong, G. (2002). Analysis and visualization of gene expression data using self-organizing maps. *Neural Networks, Special Issue on New Developments on Self-Organizing Maps, 15*(8-9), 953–966.

Nishimura, H., Tsutsumi, M., Maruyama, M., Miyao, M., & Nakano, Y., (2001). Off-line handwritten character recognition using integrated 1D HMMs based on feature extraction filters. *Proc. ICDAR,* 417–421.

NIST. (2001). *Feret database.* Retrieved from http://www.itl.nist.gov/iad/humanid/feret/

Nowicka, M., Łuba, T., & Rawski, M. (1999). FPGA-based decomposition of Boolean functions: Algorithms and implementation. *Proceedings of the Sixth International Conference on Advanced Computer Systems,* Szczecin, Poland, 502–509.

Oh, I.S., Lee, J.S., & Suen, C.Y. (1998). Using class separation for feature analysis and combination of class-dependent features. *Proceedings of the Fourteenth International Conference on Pattern Recognition, 1,* 453–455.

Oh, I.S., Lee, J.S., & Suen, C.Y. (1999). Analysis of class separation and combination of class-dependent features for handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21,* 1089–1094.

Oliveira, L., Justino, E., Freitas, C., & Sabourin, R. (2005). The graphology applied to signature verification. *Proceedings of the 12th Conference of the International Graphonomics Society,* 286–290.

Oliver, T.F., Mohammed, S., Krishna, N.M., & Maskell, D.L. (2004). Accelerating an embedded RTOS in a SoPC platform. *Proceedings of the TENCON Conference,* 4, 415–418.

Opitz, D.W., & Shavlik, J.W. (1996). Generating accurate and diverse members of a neural-network ensemble. In D.S. Touretzky, M.C. Mozer, & M.E. Hasselmo (Eds.), *Advances in neural information processing systems*, Vol. 8 (pp. 535–541). Cambridge, MA: MIT Press.

Oppenheim, A.V., & Schafer, R.W. (2004). From frequency to quefrency: A history of the cepstrum. *Signal Processing Magazine, IEEE, 21*, 95–106.

Oren, M., Papageorgiou, C., Sinha, P., Osuna, E., & Poggio T. (1997). Pedestrian detection using wavelet templates. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 193–199.

Osborn, A.S. (1929). *Questioned documents*. New York: Boyd Printing.

Ostu, N. (1978). A thresholding selection method from graylevel histogram. *IEEE Trans Systems Man Cybernet, SMC8*, 62–66.

Oz, C. (2005). Signature recognition and verification with artificial neural network using moment invariant method. *Lecture Notes in Computer Science, 3497*, 195–202.

Ozgunduz, E., Senturk, T., & Karsligil, E. (2005). Off-line signature verification and recognition by support vector machine. *Proceedings of the European Signal Processing Conference*.

Pal, N.R., Aguan, K., Sharma, A., & Amari, S. (2007). Discovering biomarkers from gene expression data for predicting cancer subgroups using neural networks and relational fuzzy clustering. *BMC Bioinformatics, 8*(5). Retrieved January 19, 2007, from http://www.biomedcentral.com/1471-2105/8/5.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision, 38*(1), 15–33.

Park, H.-S. & Lee, S.W. (1998). A truly 2-D hidden Markov model for off-line handwritten character recognition. *Pattern Recognit, 31*(12), 1849–1864.

Paschalakis, S., & Bober, M. (2003). A low cost FPGA system for high speed face detection and tracking. *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT)*, 214–221.

Pavlidis, T. (1982). *Algorithms for graphics and image processing*. Computer Science Press.

Pekalska, E., & Duin, R. (2000). Classifiers for dissimilarity-based pattern recognition. *Proceedings of the International Conference on Pattern Recognition*, Vol. 2, 12–16.

Peled, A., & Liu, B. (1974). A new realization of digital filters. *IEEE Transactions on Acoustics, Speech and Signal Processing, 22*(6), 456–462.

Pender, D.A. (1991). *Neural networks and handwritten signature verification* [doctoral dissertation]. Stanford University.

Pentland, A., Moghaddam, B., & Starner, T. (1994). View-based and modular eigenspaces for face recognition. *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 84–91.

Perez-Hernandez, A., Sanchez, A., & Velez, J. (2004). Simplified stroke-based approach for off-line signature recognition. *Proceedings of the 2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications*, 89–94.

Perrone, M., & Cooper, L. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R.J. Mammone (Ed.), *Artificial neural networks for speech and vision* (pp. 126–142). London: Chapman & Hall.

Pervouchine, V., Leedham, C.G., & Melikhov, K. (2005). Handwritten character skeletonisation for forensic document analysis. *Proceedings of the 20th Annual ACM Symposium on Applied Computing*, (pp. 754–758). Santa Fe, New Mexico.

Pervouchine, V., Leedham, C.G., & Melikhov, K. (2005). Three-stage handwriting stroke extraction method with hidden loop recovery. *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR'2005)*. Seoul, Korea.

Peskin, B., Gillick, L., & Liberman, N. (1991). Progress in recognizing conversational telephone speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP'97) Toronto, Canada (vol. 3, pp. 1811-1814). Washington, DC: IEEE Computer Society.

Pham, D.T.D., Tran, T., Zhou, X., & Wong, S.T.C. (2005). An automated procedure for cellphase imaging identification. *Proceedings of the AI-2005 Workshop on Learning Algorithms for Pattern Recognition*, 52–59.

Pham, T.D., Tran, D.T., Zhou, X., & Wong, S.T.C. (2006). Classification of cell phases in time-lapse images by vector quantization and Markov models. In E.V. Greer, Ed., *Neural stem cell research*. New York: Nova Science.

Phillips, P.J., Grother, P., Micheals, R., Blackburn, D.M., Tabassi, E., & Bone, M. (2003). Face recognition vendor test 2002. *Proceedings of the Analysis and Modeling of Faces and Gestures*, 44.

Phung, S.L., Bouzerdoum, A., & Chai, D. (2005). Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(1), 148–154.

Plamondon, R. (Ed.). (1994). *Progress in automatic signature verification*. Singapore: Word Scientific.

Plamondon, R., & Lorette, G. (1989). Automatic signature verification and writer identification—The state of the art. *Pattern Recognition*, *22*, 107–131.

Plamondon, R., & Lorette, G. (1989). Designing automatic signature verification and writer identification—The state of the art. *Pattern Recognition*, *2*(2), 107–131.

Plamondon, R., & Srihari, S.N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 63–84.

Platt, J.C., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin DAGs for multi-class classification. *Advances in Neural Information Processing Systems*, *12*, 547–553.

Poisson, E., Gaudin, C.V., & Lallican, P.-M. (2002). Multi-modular architecture based on convolutional neural networks for online handwritten character recognition. *Proceedings of the 9th International Conference on Neural Information Processing*, Vol. 5, 2444–2448.

Pontecorvo, C., & Bouzerdoum, A. (1997). Edge detection in multiplicative noise using the shunting inhibitory cellular neural network. *Proceedings of the International Conference on Engineering Applications of Neural Networks*, Stockholm, Sweden, 281–285.

Poritz, A. (1982). Linear predictive hidden Markov models and the speech signal. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 7*, 1291–1294.

Prescott, S.A., & Koninck, Y.D. (2003). Gain control of firing rate by shunting inhibition: Roles of synaptic noise and dendritic saturation. *Proceedings of the National Academy of Sciences of the United States of America, 100*(4), 2076–2081.

Proesmans, M., Gool, L.V., Pauwels, E., & Osterlinck, A. (1994). *Determination of optical flow and its discontinuities using non-linear diffusion.* Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden, 2, 295–304.

Qi, Y., & Hunt, B. (1994). Signature verification using global and grid features. *Pattern Recognition*, *27*(12), 1621–1629.

Quek, C., & Zhou, R.W. (2002). Antiforgery: A novel pseudo-outer product based fuzzy neural network driven signature verification system. *Pattern Recognition Letters, 23*(14), 1795–1816.

Quénot, G.M. (1992). The orthogonal algorithm for optical flow detection using dynamic programming. *Proc. ICASSP*, III-249–252.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE, 77*(2), 257–286.

Rabiner, L. R., & Levinson, S. E. (1990). Isolated and connected word recognition-theory and selected applica-

tions. In A. Waibel & K. Lee (Eds.), *Readings in speech recognition* (pp. 115-153). New York: Morgan Kaufman Publishers.

Rabiner, L., & Biing-Hwang, J. (1993). *Fundamentals of speech recognition.* Englewood Cliffs, NJ: Prentice Hall.

Rabiner, L.R. (1989). A tutorial on hidden Markov model and selected applications in speech recognition. *Proceedings of the IEEE, 77,* 257-286.

Raileanu, L.E., & Stoffel, K. (2004). Theoretical comparison between the Gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence, 1*(41), 77–93.

Ramesh, V., & Murty, M. (1999). Off-line signature verification using genetically optimized weighted features. *Pattern Recognition, 32,* 217–233.

Randen, T., & Husøy, J.H. (1999). Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(4), 291-310.

Rao, R.M., & Bopardikar, A.S. (1998). *Wavelet transform: Introduction to theory and applications.* Addison-Wesley.

Rawski, M., Jóźwiak, L., & Łuba T. (1999). Efficient input support selection for sub-functions in functional decomposition based on information relationship measures. *Proceedings of the EUROMICRO'99 Conference,* Milan, Italy.

Rawski, M., Jóźwiak, L., & Łuba, T. (1999). The influence of the number of values in sub-functions on the effectiveness and efficiency of the functional decomposition. *Proceedings of the EUROMICRO'99 Conference,* Milan, Italy.

Rawski, M., Jóźwiak, L., & Łuba T. (2001). Functional decomposition with an efficient input support selection for sub-functions based on information relationship measures. *Journal of Systems Architecture, 47,* 137–155.

Rawski, M., Selvaraj, H., & Morawiecki, P. (2004). Efficient method of input variable partitioning in functional decomposition based on evolutionary algorithms. *Proceedings of the DSD 2004 Euromicro Symposium on Digital System Design, Architectures, Methods and Tools,* Rennes, France, (pp. 136–143).

Rawski, M., Tomaszewicz, P., & Łuba, T. (2004). Logic synthesis importance in FPGA-based designing of information and signal processing systems. *Proceedings of the International Conference on Signal and Electronics Systems,* Poznań, Poland, (pp. 425–428).

Rawski, M., Tomaszewicz, P., Selvaraj, H., & Łuba, T. (2005). Efficient implementation of digital filters with use of advanced synthesis methods targeted FPGA architectures. *Proceedings of the Eighth Euromicro Conference on DIGITAL SYSTEM DESIGN,* Portugal, (pp. 460–466).

Redert, A., Hendriks, E., & Biemond, J. (1999). Correspondence estimation in image pairs. *IEEE SP Mag, 16*(3), 29–46.

Revow, M., Williams, C.K.I., & Hinton, G.E. (1996). Using generative models for handwritten digit recognition. *IEEE Trans Pattern Anal Mach Intell, 18*(6), 592–606.

Reynolds, D.A., & Rose, R.C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing, 3,* 72–83.

Riedmiller, M. (1994). Advanced supervised learning in multilayer perceptrons—From backpropagation to adaptive learning algorithms. *Neural Networks, 5,* 265–278.

Rigoll, G., & Kosmala, A. (1998). A systematic comparison between on-line and off-line methods for signature verification with hidden Markov models. *Proceedings of the International Conference on Pattern Recognition,* Vol. 2, 1755–1757.

Rioul, O., & Vetterli, M. (1991). Wavelets and signal processing. *IEEE Signal Processing Magazine,* 14–38.

Ripon, K.S.N., Tsang, C., & Kwong, S. (2006). Multi-objective data clustering using variable-length real jumping genes genetic algorithm and local search method.

Proceedings of the International Joint Conference on Neural Networks, 3609–3616.

Robertson, E.W. (1991). *Fundamentals of document examination*. IL: Nelson Hall.

Rocha, J., & Pravlidis, T. (1994). A shape analysis model with application to a character recognition system. *IEEE Trans Pattern Anal Mach Intell, 16*(4), 393–404.

Ronee, M.A., Uchida, S., & Sakoe, H. (2001). Handwritten character recognition using piecewise linear two-dimensional warping. *Proc. ICDAR*, 39–43.

Rosen, B.E. (1996). Ensemble learning using decorrelated neural networks. *Connection Science—Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, *8*(3/4), 373–384.

Rosenfeld, A. (1973). Arcs and curves in digital pictures, *JACM, 20*, 81–87.

Rowley, H.A. (1999). *Neural network-based face detection* [unpublished doctoral dissertation]. Carnegie Mellon University.

Rowley, H.A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(1), 23–38.

Rowley, H.A., Baluja, S., & Kanade, T. (1998). Rotation invariant neural network-based face detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 38–44.

Ruiz, R., Aguilar-Ruiz, J.S., & Riquelme, J.C. (2002). SOAP: Efficient feature selection of numeric attributes. *Proceedings of the Iberoamerican Conference on Artificial Intelligence, IBERAMIA'02*, 233–242.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representation by error propagation. In D.E. Rumelhart, & J.L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*, Vol. 1 (pp. 318–362). Cambridge, MA: MIT Press.

Sabourin, R., & Drouhard, J. (1992). Off-line signature verification using directional pdf and neural networks. *Proceedings of the International Conference on Pattern Recognition*, 321–325.

Sabourin, R., & Genest, G. (1994). An extended shadow code approach for off-line signature verification: Part I—Evaluation of the bar mask definition. *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, 450–460.

Sabourin, R., & Genest, G. (1995). An extended-shadow-code based approach for off-line signature verification: Part II—Evaluation of several multi-classifier combination strategies. *Proceedings of the Third IAPR Conference on Document Analysis and Recognition*, 197–201.

Sabourin, R., & Plamondon, R. (1986). Preprocessing of handwritten signatures form image gradient analysis. *Proceedings of the 8th International Conference on Pattern Recognition*, 576–579.

Sabourin, R., Cheriet, M., & Genest, G. (1993). An extended shadow-code based approach for off-line signature verification. *Proceedings of the International Conference on Document Analysis and Recognition*, 1–5.

Sabourin, R., Drouhard, J., & Wah, E. (1997). Shape matrices as a mixed shape factor for off-line signature verification. *Proceedings of the International Conference on Document Analysis and Recognition*, Vol. 2, 661–665.

Sabourin, R., Genest, G., & Prêteux, F. (1996). Pattern spectrum as a local shape factor for off-line signature verification. *Proceedings of the 13th International Conference on Pattern Recognition*, C43–C48.

Sabourin, R., Genest, G., & Prêteux, F. (1997). Off-line signature verification by local granulometric size distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(9), 976–988.

Sabourin, R., Genest, G., & Prêteux, F.J. (1997). Off-line signature verification by local granulometric size distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(9), 976–988.

Sabourin, R., Plamondon, R., & Beaumier, L. (1994). Structural interpretation of handwritten signature im-

ages. *International Journal of Pattern Recognition and Artificial Intelligence*, *8*(3), 709–748.

Sadri, M.S., et al. (2004). An FPGA based fast face detector. *Proceedings of the Global Signal Processing Expo and Conference*.

Saito, T., Yamada, H., & Yamamoto, K. (1982). An analysis of handprinted Chinese characters by directional pattern matching approach [in Japanese]. *IEICE Trans, J65-D*(5), 550–557.

Sakaue, K., Amano, A., & Yokoya, N. (1999). Optimization approaches in computer vision and image processing. *IEICE Trans Inf & Syst, E82-D*(3), 534–547.

Sakoe, H. (1974). *Handwritten character recognition by rubber string matching method* [in Japanese]. Tech. Rep. of IEICE Japan, PRL74-20.

Sakoe, H., Ali, M.M., & Katayama, Y. (1994). One dimensional-two dimensional dynamic programming algorithm for character recognition. *IEICE Trans Information & Systems, E77-D*(9), 1047–1054.

Sanderson, C., & Paliwal, K.K. (2002). Fast feature extraction method for robust face verification. *Electronics Letters, 38*(25), 1648–1650.

Sanderson, C., Bengio, S., & Gao, Y. (2006). On transforming statistical models for non-frontal face verification. *Pattern Recognition*, *39*(2), 288–302.

Sansone, C., & Vento, M. (2000). Signature verification: Increasing performance by a multi-stage system. *Pattern Analysis and Applications*, *3*, 169–181.

Sant'Ana, R., Coelho, R., & Alcaim, A. (2006). Text-independent speaker recognition based on the Hurst parameter and the multidimensional fractional Brownian motion model. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(3), 931–940.

Santos, C., Justino, E., Bortolozzi, F., & Sabourin, R.(2004). An off-line signature verification method based on the questioned document expert's approach and a neural network classifier. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 498–502.

Sasao, T., Iguchi, Y., & Suzuki, T. (2005). On LUT cascade realizations of FIR filters. *Proceedings of the Eighth Euromicro Conference on DIGITAL SYSTEM DESIGN*, Portugal, (pp. 467–474).

Schambach, M.-P. (2005). Fast script word recognition with very large vocabulary. *Proceedings of the 8th International Conference on Document Analysis and Recognition* (pp. 9–13). Seoul, Korea: IEEE Computer Society Press.

Schapire, R.E. (1990). The strength of weak learnability. *Machine Learning, 5*(2), 197–227.

Scholkopf, B., Bartlett, P., Smola, A., & Williamson, R. (1999). Shrinking the tube: A new support vector regression algorithm. *Advances in Neural Information Processing Systems, 11*, 330–336.

Scholkopf, B., Platt, J., Taylor, J., Smola, A., & Williamson, R. (2001). Estimating the support of a high dimensional distribution. *Neural Computation*, *13*, 1443–1471.

Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., & Williamson, R.C. (1999). *Estimating the support of a high-dimensional distribution* (Tech. Rep MSR-TR-99-87). Redmond: Microsoft Research.

Scholl, C. (2001). *Functional decomposition with application to FPGA synthesis.* Kluwer Academic Publishers.

Senior, A.W. (1994). *Off-line cursive handwriting recognition using recurrent neural networks* [unpublished doctoral dissertation]. Cambridge, England: University of Cambridge.

Senol, C., & Yildirim, T. (2005). Signature verification using conic section function neural network. *Proceedings of the 20th International Symposium Computer and Information Sciences*, 524–532.

Shakhnarovich, G., Viola, P.A., & Moghaddam, B. (2002). A unified learning framework for real time face detection and classification. *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, 14–21.

Sharkey, A. (1999). Combining artificial neural nets: Ensemble and modular multi-net systems. In *Multi-net systems* (pp. 1–30). Springer-Verlag.

Shashua, A., Gdalyahu, Y., & Hayon, G. (2004). Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. *Proceedings of the IEEE Intelligent Vehicles Symposium*, 1–6.

Shi, D., Gunn, S.R., & Damper, R.I. (2003). Handwritten Chinese radical recognition using nonlinear active shape models. *IEEE Trans Pattern Anal Mach Intell, 25*(2), 277–280.

Shi, M., Fujisawa, Y., Wakabayashi, T., & Kimura, F. (2002). Handwritten numeral recognition using gradient and curvature of gray scale image. *Patt Recog, 35*, 2051–2059.

Shih, F.Y., & Wong, W.-T. (1995). A new safe-point thinning algorithm based on the mid-crack code tracing. *IEEE Transactions on Systems, Man and Cybernetics, 25*(2), 370–378.

Shiku, O., Nakamura, A., Kuroda, H., & Miyahara, S. (2000). A method for handwritten Japanese word recognition based on holistic strategy [in Japanese]. *Trans Inf Process Soc Japan, 41*(4), 1086–1095.

Shilton, A. (2006). *Design and training of support vector machines* [doctoral dissertation]. Melbourne: The University of Melbourne.

Shilton, A., & Palaniswami, M. (2004). A modified nu-SV method for simplified regression. *Proceedings of the International Conference on Intelligent Sensing and Information Processing*, 422–427.

Shilton, A., Palaniswami, M., Ralph, D., & Tsoi, A.C. (2005). Incremental training of support vector machines. *IEEE Transactions on Neural Networks, 16*(1), 114–131.

Simard, P., Le Cun, Y., & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems, 5*, 50–58.

Simard, P.Y., Steinkraus, D., & Platt, J.C. (2003). Best practices for convolutional neural networks applied to visual documents analysis. *Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2*, 958–962.

Singh, S., & Amin, A. (1999). Neural network recognition of hand printed characters. *Neural Computing & Applications, 8*(1), 67–76.

Smola, A. (1996). *Regression estimation with support vector learning machines* [master's thesis]. Technische Universitat Munschen.

Smola, A., & Scholkopf, B. (1998). *A tutorial on support vector regression* (Tech. Rep. NeuroCOLT2 Technical Report Series, NC2-TR-1998-030). London: University of London, Royal Holloway College.

Smola, A., Scholkopf, B., & Muller, K. (1998). Convex cost functions for support vector regression. *Proceedings of the 8th ICANN, Perspectives in Neural Computing*, 99–104.

Smola, A., Scholkopf, B., & Muller, K. (1998). General cost functions for support vector regression. *Proceedings of the Ninth Australian Conference on Neural Networks*, 79–83.

Solihin, Y. (1997). *A toolset of image processing algorithms for forensic document examination* [unpublished master's thesis]. Nanyang Technological University, School of Applied Science.

Sonka, M., Hlavac, V., & Boyle, R. (1993). *Image processing, analysis and machine vision*. Cambridge: Chapman & Hall Computing.

Sorlie, T., et al. (2001). Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98, 10869–10874.

Souto, M.C.P., Araujo, D.S.A., & Silva, B.L.C. (2006). Cluster ensemble for gene expression microarray data: Accuracy and diversity. *Proceedings of the IEEE International Joint Conference on Neural Networks*, 16, 2174–2180.

Specht, D.F. (1990). Probabilistic neural networks. *Neural Networks, 3,* 109-118.

Spellman, P., et al. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell, 9,* 3273–3297.

Srihari, S., Xu, A., & Kalera, M. (2004). Learning strategies and classification methods for off-line signature verification. *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, 161–166.

Srihari, S.N. (1993). Recognition of handwritten and machine-printed text for postal address interpretation. *Pattern Recognition Letters, 14*(4), 291–302.

Srihari, S.N. (2006). Automatic handwriting recognition. *Encyclopedia of Language & Linguistics, 2nd Edition*. Elsevier.

Srihari, S.N., Cha, S.-H., & Lee, S. (2001). Establishing handwriting individuality using pattern recognition techniques. *Proceedings of the 6th International Conference on Document Analysis and Recognition (ICDAR'2001),* (pp. 1195–1204). Seattle, Washington.

Srihari, S.N., Cha, S.-H., Arora, H., & Lee, S. (2002). Individuality of handwriting. *Journal of Forensic Sciences, 47*(4), 1–17.

Srihari, S.N., Tomai, C.I., Zhang, B., & Lee, S. (2003). Individuality of numerals. *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'2003),* (pp. 1096–1100). Edinburgh, UK.

Stauffer, C., & Grimson, W. (1999). *Adaptive background mixture models for real-time tracking.* Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ft. Collins, Colorado, 2, 246–252.

Steinherz, T., Rivlin, E., & Intrator, N. (1999). Offline cursive script word recognition—A survey. *International Journal of Document Analysis and Recognition, 2*, 90–110.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles—A knowledge reuse framework for combining partitions. *Journal of Machine Learning Research, 3*, 583–617.

Suárez-Guerra, S. (2005). ¿100% de reconocimiento de voz? Unpublished.

Suen, C.Y., & Tan, J. (2005). Analysis of errors of handwritten digits made by a multitude of classifiers. *Pattern Recognition Letters, 26*(3), 369–379.

Suen, C.Y., & Wang, P.S.P. (Eds.). (1994). *Thinning methodologies for pattern recognition* (Vol. 8). Singapore: World Scientific.

Suen, C.Y., Legault, R., Nadal, C., Cheriet, M., & Lam, L. (1993). Building a new generation of handwriting recognition systems. *Pattern Recognition Letters, 14*(4), 305–315.

Suen, C.Y., Xu, Q., & Lam, L. (1999). Automatic recognition of handwritten data on cheques—Fact or fiction? *Pattern Recognition Letters, 20,* 1287–1295.

Sugimura, M., Iiguni, Y., & Adachi, N. (1997). A 2-dimensional dynamic programming for image matching with Zernike moments [in Japanese]. *Trans IEICE, J80-D-II*(1), 101–108.

Sun, Z., Yuan, X., Bebis, G., & Louis, S. (2002). Neural-network-based gender classification using genetic eigenfeature extraction. *Proceedings of the International Joint Conference on Neural Networks, 3*, 2433–2438.

Sung, K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence, 20*(1), 31–59.

Suters, M., & Yan, H. (1994). Connected handwritten digit separation using external boundary curvature. *Journal of Electronic Imaging, 3*(3), 251–256.

Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least squares support vector machines.* New Jersey: World Scientific Publishing.

Sydral, A., Bennet, R., & Greenspan, S. (1995). *Applied speech technology.* New York: CRC.

Tai-Chi, L., Zeien, R., Roach, A., & Robinson, P. (2006). DES decoding using FPGA and custom instructions.

Proceedings of the International Conference on Information Technology: New Generations, 575–577.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on System, Man and Cybernetics, 15*, 116–132.

Tamayo, P., et al. (1999). Interpreting patterns of gene expression with selforganizing maps: Methods and application to hematopoietic differentiation. *Proceedings of the National Academy of Sciences*, 96, 2907–2912.

Tanaka, E. (1985). A two dimensional context-dependent similarity measure. *Trans IECE Japan, E-68*(10), 667–673.

Tanay, A., Steinfeld, I., Kupiec, M., & Shamir, R. (2005). Integrative analysis of genome-wide experiments in the context of a large high-throughput data compendium. *Molecular Systems Biology, 1.*

Tang, Y.Y., Tao, Y., & Lam, E.C.M. (2002). New method for feature extraction based on fractal behavior. *Pattern Recognition, 35*(5), 1071–1081.

Tax, D. (2001). *One-class classification* [unpublished doctoral dissertation]. TU Delft.

Tebelskis, J. (1995). *Speech recognition using neural networks.* Unpublished doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Theocharides, T., Link, G., Vijaykrishnan, N., Irwin, M.J., & Wolf, W. (2004). Embedded hardware face detection. *Proceedings of the 17th International Conference on VLSI Design*, 133–138.

Tivive, F.H.C., & Bouzerdoum, A. (2006). Application of SICoNNets to handwritten digit recognition. *International Journal of Computational Intelligence and Applications, 6*(1), 45–59.

Tollenaere, T. (1990). SuperSAB: Fast adaptive BP with good scaling properties. *Neural Networks, 3*, 561–573.

Topchy, A., Jain, A., & Punch, W. (2004). A mixture model for clustering ensembles. *Proceedings of the SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, 331–338.

Topchy, A., Jain, A., & Punch, W. (2005). Clustering ensembles: Models of consensus and weak partitions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(12), 1866–1881.

Tortorella, F. (2005). A roc-based reject rule for dichotomizers. *Pattern Recognition Letters, 26*, 167–180.

Trier, Ø.D., Jain, A.K., & Taxt, T. (1996). Feature extraction methods for character recognition—A survey. *Pattern Recognit, 29*(4), 641–662.

Tsuji, T., Bu, N., Fukuda, O., & Kaneko, M. (2003). A recurrent log-linearized Gaussian mixture network. *IEEE Transactions on Neural Networks, 14*, 304-316

Tsuji, T., Fukuda, O., Ichinobe, H., & Kaneko, M. (1999). A log-linearized Gaussian mixture network and its application to EEG pattern classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 29*, 60-72.

Tsuji, T., Fukuda, O., Kaneko, M., & Ito, K. (2000). Pattern classification of time-series EMG signals using neural networks. *International Journal of Adaptive Control and Signal Processing, 14*, 829-848.

Tsuji, T., Ichinobe, H., Ito, K., & Nagamachi, M. (1993). Discrimination of forearm motions from EMG signals by error back propagation typed neural network using entropy (in Japanese). *Transactions of the Society of Instrument and Control Engineers, 29*, 1213-1220.

Tsukumo, J. (1992). Handprinted Kanji character recognition based on flexible template matching. *Proc. ICPR*, 483–486.

Tsutsui, H., Masuzaki, T., Izumi, T., Onoye, T., & Nakamura, Y. (2002). High speed JPEG2000 encoder by configurable processor. *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, 1, 45–50.

Tuceryan, M., & Jain, A.K. (1998). Texture analysis. In C.H. Chen, L.F. Pau, & P.S.P. Wang (Eds.), *The handbook of pattern recognition and computer vision* (2nd ed.) (pp. 207–248). Singapore: World Scientific.

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience, 3*(1), 71–86.

Turk, M.A., & Pentland, A.P. (1991). Face recognition using eigenfaces. *Computer Vision and Pattern Recognition*, 586–591.

Turner, K., & Gosh, J. (1996). Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition, 29*(2), 341–348.

Turner, K., & Gosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science, 8*(3-4), 385–403.

Uchida, S., & Sakoe, H. (1998). A monotonic and continuous two-dimensional warping based on dynamic programming. *Proc. ICPR*, 1 of 2, 521–524.

Uchida, S., & Sakoe, H. (1999). Handwritten character recognition using monotonic and continuous two-dimensional warping. *Proc. ICDAR*, 499–502.

Uchida, S., & Sakoe, H. (2000). An approximation algorithm for two-dimensional warping. *IEICE Trans Info & Syst, E83-D*(1), 109–111.

Uchida, S., & Sakoe, H. (2000). Piecewise linear two-dimensional warping. *Proc ICPR, 3*, 538–541.

Uchida, S., & Sakoe, H. (2003). Eigen-deformations for elastic matching based handwritten character recognition. *Pattern Recognit, 36*(9), 2031–2040.

Uchida, S., & Sakoe, H. (2003). Handwritten character recognition using elastic matching based on a class-dependent deformation model. *Proc. ICDAR, 1*, 163–167.

Uchida, S., & Sakoe, H. (2005). A survey of elastic matching techniques for handwritten character recognition. *IEICE Transactions on Information & Systems, E88-D*(8), 1781–1790.

Ueda, N., & Nakano, R. (1996). Generalization error of ensemble estimators. *Proceedings of International Conference on Neural Networks*, 90–95.

United States vs. Starzecpyzel. (1995). 880 F. Supp. 1027, 1046 (S.D.N.Y.)

Vapnik, V. (1995). *Statistical learning theory.* New York: Springer-Verlag.

Vapnik, V. (1998). *Statistical learning theory.* New York: Wiley.

Vapnik, V. (2000). *The nature of statistical learning theory.* Berlin, Germany: Springer-Verlag Press.

Vapnik, V. (Ed.). (1999). *The nature of statistical learning theory.* 2nd edition. New York: Springer-Verlag.

Vapnik, V., Golowich, S., & Smola, A. (1997). Support vector methods for function approximation, regression estimation, and signal processing. *Advances in neural information processing systems, 9*, 281–187.

Veeneman, D. E. (1988). Speech signal analysis. In C. H. Chen (Ed) *Signal processing handbook.* New York: Marcel Dekker, Inc.

Veith, D., & Abry, P. (1994). A wavelet-based joint estimator of the parameters of long-range dependence. *IEEE Transactions on Information Theory, 45*(3), 878–897.

Vélez, J., Sánchez, A., & Moreno, A. (2003). Robust off-line signature verification using compression networks and positional cuttings. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 627–636.

Verma, B. (2003). A contour code feature based segmentation for handwriting recognition. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 1203–1207). Edinburgh, UK: IEEE Computer Society Press.

Verma, B., Blumenstein, M., & Ghosh, M. (2004). A novel approach for structural feature extraction: Contour vs. direction. *Pattern Recognition Letters, 25*(9), 975–988.

Verma, B., Blumenstein, M., & Kulkarni, S. (1998). Recent achievements in offline handwriting recognition. *Proceedings of the 2nd International Conference on Computational Intelligence and Multimedia Applications* (pp. 27–33). Melbourne, Australia: World Scientific Publishing Company.

Verma, B., Gader, P., & Chen, W. (2001). Fusion of multiple handwritten word recognition techniques. *Pattern Recognition Letters*, *22*(9), 991–998.

Vert, J.–P. (2001). Introduction to support vector machines and applications to computational biology. *Proceedings of the Seminar on SVM in Bioinformatics.*

Viard-Gaudin, C., Lallican, P.-M., & Knerr, S. (2005). Recognition-directed recovering of temporal information from handwriting images. *Pattern Recognition Letters*, *26*(16), 2537–2548.

Vida, I., Bartos, M., & Jonas, P. (2006). Shunting inhibition improves robustness of gamma oscillations in hippocampal interneuron networks by homogenizing firing rates. *Neuron*, *49*(1), 107–117.

Villing, R., Timoney, J., Ward, T., & Costello, J. (2004). Automatic blind syllable segmentation for continuous speech. In *Proceedings of the Irish Signals and Systems Conference 2004.* Belfast, UK (pp. 41-46). IET Digital Library.

Vinciarelli, A. (2002). A survey on off-line cursive word recognition. *Pattern Recognition*, *35*(7), 1433–1446.

Vinciarelli, A., Bengio, S., & Bunke, H. (2003). Offline recognition of large vocabulary cursive handwritten text. *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 1101–1107). Edinburgh, UK: IEEE Computer Society Press.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1, 511–518.

Viola, P., Jones, M., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. *Proceedings of the International Conference on Computer Vision*, *2*, 734–741.

Vuskovic, M., & Du, S. (2002). Classification of prehensile EMG patterns with simplified fuzzy ARTMAP networks. *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2539-2545.

Wakahara, T. (1994). Shape matching using LAT and its application to handwritten numeral recognition. *IEEE Trans Pattern Anal Mach Intell*, *16*(6), 618–629.

Wakahara, T., & Odaka, K. (1998). Adaptive normalization of handwritten characters using global/local affine transformation. *IEEE Trans Pattern Anal Mach Intell*, *20*(12), 1332–1341.

Wakahara, T., Kimura, Y., & Tomono, A. (2001). Affine-invariant recognition of gray-scale characters using global affine transformation correlation. *IEEE Trans Pattern Anal Mach Intell*, *23*(4), 384–395.

Wang, C.X., Qi, Y., Yu, D., & Xu, S. (1990). A fast algorithm for boundary tracing of binary image with neighborhood coding. *Proceedings of the International Conference on Signal Processing*, 1083–1085.

Wang, J., & Wang, J. (2005). Speaker recognition using features derived from fractional Fourier transform. *Proceedings of the Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, (pp. 95–100).

Wang, J.T.L., Zaki, M.J., Toivonen, H.T.T., & Shasha, D.E. (Eds.). (2003). *Data mining in bioinformatics: Advanced information and knowledge processing.* Springer.

Wang, L.P. (Ed.). (2005). *Support vector machines: Theory and applications.* New York: Springer.

Wang, L.P., & Fu, X. J. (2005). *Data mining with computational intelligence*. Berlin: Springer.

Wang, L.P., Chu, F., & Xie, W. (2007). Accurate cancer classification using expressions of very few genes. *IEEE Transactions on Computational Biology and Bioinformatics*, *1*(1), 40-53.

Wang, L.P., Chu, F., & Xie, W. (Accepted 2006). Accurate cancer classification using expressions of very few genes. *IEEE Transactions on Bioinformatics and Computational Biology.*

Wang, W., Brakensiek, A., Kosmala, A., & Rigoll, G. (2001). Multi-branch and two-pass HMM modeling approaches for off-line cursive handwriting recognition. *Proc. ICDAR*, 231–235.

Wang, X., Ding, X., & Liu, C. (2005). Gabor filters-based feature extraction for character recognition. *Pattern Recognition, 38*(3), 369–379.

Wayman, J., Jain, A., Maltoni, D., & Maio, D. (Eds.). (2005). *Biometric systems, technology, design and performance evaluation*. New York: Springer.

Weber, K. (2000). Multiple timescale feature combination towards robust speech recognition. In *Konferenz zur Verarbeitung natürlicher Sprache KOVENS2000*. Ilmenau, Germany.

Webster, R., & Nakagawa, M. (1997). An on-line/off-line compatible character recognition method based on a dynamic model. *IEICE Trans Inf & Syst, E80-D*(6), 672–683.

Wei, M., & Bigdeli, A. (2004). Implementation of a real-time automated face recognition system for portable devices. *Proceedings of the IEEE International Symposium on Communications and Information Technology, 1*, 89–92.

Weiss, S.M., & Kulikowski, C.A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems.* Morgan Kaufmann.

Wen, Y., Lu, Y., & Shi, P. (2007). Handwritten bangla numeral recognition system and its application to postal automation. *Pattern Recognition, 40*(1), 99–107.

Wen-Ming, Z., Shao-Fa, L., & Xian-Gui, Z. (2004). A hybrid scheme for off-line Chinese signature verification. *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, Vol. 2, 1402–1405.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for SVMs. *Advances in Neural Information Processing Systems, 13*, 668–674.

Whichello, A., & Yan, H. (1996). Lining broken character borders with variable sized masks to improve recognition. *Patt Recog, 29*(8), 1429–1435.

Wilkinson, T., & Goodman, J. (1990). Slope histogram detection of forged handwritten signatures. *Proceedings of the SPIE—International Society for Optical Engineering*, 293–304.

Williams, C. S. (1986). *Designing digital filters.* Englewood Cliffs, NJ: Prentice Hall.

Wiskott, L., & von der Malsburg, C. (1996). Face recognition by dynamic link matching. In J. Sirosh, R. Miikkulainen, & Y. Choe (Eds.), *Lateral interactions in the cortex: Structure and function*. Austin, TX: UTCS Neural Networks Research Group.

Wiskott, L., Fellous, J.M., Kuiger N., & von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions, 19*(7), 775–779.

Wu, B., Ai, H., & Huang, C. (2003). LUT-based adaboost for gender classification. *Proceedings of the Fourth International Conference on Audio- and Video-Based Biometric Person Authentication*, 104–110.

Wu, C.-M., Liu, P., & Chang, W.-C. (1995). Unconstrained-endpoint dynamic space-warping algorithm with experiments in binary English character images. *Int J Electronics, 78*(1), 55–66.

Wu, S. (1998). *Incorporating information from syllable-length time scales into automatic speech recognition.* Unpublished doctoral dissertation, Berkeley University.

Wu, S., Shire, M., Greenberg, S., & Morgan, N. (1997). Integrating syllable boundary information into automatic speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Munich, Germany (vol. 1, pp. 987-990). Washington, DC: IEEE Computer Society.

Xiao, X., & Leedham, G. (2000). Knowledge-based cursive script segmentation. *Pattern Recognition Letters, 21*(10), 945–954.

Xiao, X., & Leedham, G. (2002). Signature verification using a modified Bayesian network. *Pattern Recognition, 35*(5), 983–995.

Xiujuan, C., Shiguang, S., & Wen, G. (2003). Pose normalization for robust face recognition based on statistical affine transformation. *Proceedings of the Information,*

Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, 3, 1413–1417.

Xu, Q., Lam, L., & Suen, C.Y. (2003). Automatic segmentation and recognition system for handwritten dates on Canadian bank cheques. In M. Fairhurst, & A. Downton (Eds.), *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (pp. 704–709). Edinburgh, UK: IEEE Computer Society Press.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, *16*(3), 645–678.

Xuhua, Y., Furuhashi, T., Obata, K., & Uchikawa, Y. (1996). Selection of features for signature verification using the genetic algorithm. *Computers & Industrial Engineering*, *30*(4), 1037–1045.

Yamada, H. (1984). Contour DP matching method and its application to handprinted Chinese character recognition. *Proc. ICPR*, 389–392, 1 of 2.

Yamada, H., Saito, T., & Mori, S. (1981). An improvement of correlation method—Locally maximized correlation [in Japanese]. *Trans IEICE, J64-D*(10), 970–976.

Yamada, H., Yamamoto, K., & Saito, T. (1990). A nonlinear normalization method for handprinted Kanji character recognition—Line density equalization. *Pattern Recognit, 23*(9), 1023–1029.

Yamamoto, K., & Rosenfeld, A. (1982). Recognition of hand-printed KANJI characters by a relaxation method. *Proc. ICPR*, 1 of 2, 395–398.

Yan, H. (1993). Prototype optimization of a nearest neighbor classifier using a multi-layer neural network. *Patt Recog, 26*, 317–324.

Yan, H. (1994). Handwritten digit recognition using optimized prototypes. *Patt Lett, 15*, 207–211.

Yang, J., Parekh, R., & Honavar, V. (1997). *Distal: An inter-pattern distance-based constructive learning algorithm* (Technical Report No. ISU-CS-TR 97-06). Iowa State University, Department of Computer Science.

Yang, M.-H., Kriegman, D. J., and Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 34–58.

Yang, Y., & Yan, H. (2000). An adaptive logical method for binarization of degraded document images. *Pattern Recognition, 33*(5), 787–807.

Yanikoglu, B., & Sandon, P.A. (1998). Segmentation of off-line cursive handwriting using linear programming. *Pattern Recognition, 31*(12), 1825–1833.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), 1423–1447.

Yarrow, J.C., et al. (2003). Phenotypic screening of small molecule libraries by high throughput cell imaging. *Comb Chem High Throughput Screen, 6*, 279–286.

Yasuda, M., Yamamoto, K., & Yamada, H. (1997). Effect of the perturbed correlation method for optical character recognition. *Pattern Recognit, 30*(8), 1315–1320.

Ye, X., Hou, W., & Feng, W. (2005). Off-line handwritten signature verification with inflections feature. *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Vol. 2, 787–792.

Yen, C., Kuo, S.S., & Lee, C.-H. (1999). Minimum error rate training for PHMM-based text recognition. *IEEE Trans Image Proc*, *8*(8), 1120–1124.

Yeoh, E.J., et al. (2002). Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, *1*(2), 133–143.

Yilmaz, A., & Gokmen, M. (2000). Eigenhill vs. eigenface and eigenedge. *Proceedings of the 15th International Conference on Pattern Recognition*, 2, 827–830.

Yongsheng, G., & Leung, M.K.H. (2002). Face recognition using line edge map. *Pattern Analysis and Machine Intelligence, IEEE Transactions, 24*(6), 764–779.

You, X., Fang, B., He, Z., & Tang, Y. (2005). Similarity measurement for off-line signature verification. *Lecture Notes in Computer Science: Advances in Intelligent Computing, 3644*, 272–281.

Yu, D., & Yan, H. (1997). An efficient algorithm for smoothing, linearization and detection of structure feature points of binary image contours, *Patt Recog, 30*(1), 57–69.

Yu, D., & Yan, H. (1998). Separation of single-touching handwritten numeral strings based on structural features. *Patt Recog, 31*(12), 1835–1847.

Yu, D., Hu, J., & Yan, H. (1999). Separation of single- and double-touching handwritten numeral strings based on structural analysis. *Optical Engineering, 38*(3), 514–522.

Zhang, B., Srihari, S.N., & Lee, S. (2003). Individuality of handwritten characters. *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'2003)*, (pp. 1086–1090). Edinburgh, UK.

Zhang, G.D. (2000). Neural network for classification: A survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 30*, 451-462.

Zhang, T.Y., & Suen, C.Y. (1984). A fast parallel algorithm for thinning digital patterns. *Comm. ACM, 27*(3), 236–239.

Zhao, T., & Nevatia, R. (2004). *Tracking multiple humans in crowded environments.* Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, 2, 406–413.

Zheng, N., & Ching, P.C. (2004). Using Haar transformed vocal source information for automatic speaker recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 1*, 77–80.

Zhu, Q., Avidan, S., Yeh, M.C., & Cheng, K.T. (2006). Fast human detection using a cascade of histograms of oriented gradients. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2*, 1491–1498.

Zitová, B., & Flusser, J. (2003). Image registration methods: A survey. *Image and Vision Computing, 21*, 977–1000.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zivkovic, Z. (2004). *Improved adaptive Gaussian mixture model for background subtraction.* Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 2, 28–31.

About the Contributors

Brijesh Verma is associate professor in the School of Computing Sciences at Central Queensland University. His research interests include pattern recognition and computational intelligence. He has authored/co-authored 11 books (one textbook, 10 edited books), five book chapters, and 101 papers in refereed international journals and conference proceedings. He has supervised 29 research students, including seven PhD students in the areas of pattern recognition and computational intelligence. He has received (chief investigator/co-chief investigator) 11 research grants, including ARC and industry linkage grants. He is editor-in-chief of the *International Journal of Computational Intelligence and Applications* (*IJCIA*) published by World Scientific, and an associate editor of *IEEE Transaction on Biomedicine in Information Technology.* He is serving or has served on organizing, program, and advisory committees of more than 35 national and international conferences. He is a senior member of IEEE, IEEE Computational Intelligence Society, and a member of the International Neural Network Society (INNS). He is a chair of IEEE Computational Intelligence Queensland Chapter.

Michael Blumenstein is senior lecturer and head of the School of Information and Communication Technology, Griffith University, Gold Coast Campus. His research interests include the areas of pattern recognition and computational intelligence. His research spans such topics as automated handwriting recognition, the analysis of video imagery for automatic object detection (surveillance), and numerous applications of artificial intelligence to the fields of engineering, environmental science, and coastal management. Dr Blumenstein has published more than 50 papers in peer-reviewed conferences and journals in his areas of expertise, and has been invited to serve on numerous national/international conference committees in the fields of pattern recognition and computational intelligence.

* * *

Luana Bezerra Batista has been a PhD student of the École de Technologie Supérieure, Université du Québec in Montréal, since 2006. In 2005, she was research assistant in a collaborative project between HP-Brazil and Federal University of Campina Grande in the area of digital photography. In 2004, she worked for a company in Brazil developing neural network software for financial applications and data mining. She received a BSc in 2002 from the Federal University of Paraíba, and in 2004, an MSc from the Federal University of Campina Grande, both in computer science.

Abbas Bigdeli has more than six years experience in consultancy, scientific research, and technology leadership in the areas of digital signal and image processing, computer architecture, and information security. Bigdeli is a member of the IEEE and has published more than 40 papers in journals, book

chapters, and international conferences. He has three patents in the areas of information security and computer vision. He has been acting as a technical reviewer for several journals, including the *Journal of Microprocessors and Microsystems, European Signal Processing Association, Australian* and *Journal of Research and Practice in IT,* as well as for conferences such as FPL, IEEE VLSI, and EUSIPCO.

Abdesselam Bouzerdoum is professor of vomputer rngineering and associate dean of research (faculty of informatics), University of Wollongong, Australia. He received an MSc and PhD, both in electrical engineering, from the University of Washington, Seattle. In 1991, he joined Adelaide University as a research associate and became a faculty member in June 1992. In 1998, he joined Edith Cowan University, Western Australia, as an associate professor. In 2004, he was appointed professor of computer engineering and head of the School of Electrical, Computer, and Telecommunications Engineering at the University of Wollongong. He held several visiting professor appointments at the Institut Galilée, University of Paris-13 in 2004, 2005, and 2007; and at the Hong Kong University of Science and Technology in 2007. He has published more than 200 technical articles and graduated 15 PhD and six research masters students. Prof. Bouzerdoum has received several fellowships and distinguished awards; among them are the Vice Chancellor's Distinguished Researcher Award in 1998 and 1999, Awards for Excellence in Research Leadership and Excellence in Postgraduate Supervision, and the Chester Sall Award for best paper in *IEEE Trans. on Consumer Electronics* in 2004. In 2001, he was awarded a Distinguished Researcher (Chercheur de Haut Niveau) Fellowship from the French Ministry of Research. He served as Chair of the IEEE WA Section Signal Processing Chapter in 2004, and was Chair of the IEEE SA Section NN RIG from 1995 to 1997. From 1999 to 2006, he served as Associate Editor of the IEEE *Transactions on Systems, Man and Cybernetics.* Currently, he is serving as Associate Editor of the *International Journal of Computational Intelligence and Applications* and is a member of the governing board of the Asia Pacific Neural Network Assembly.

Shaokang Chen received a BE in automatic control engineering from the South China University of Technology in 1999. In 2005, he received a PhD in electrical engineering from the University of Queensland (UQ), Australia. He has worked as a researcher in UQ and NICTA since 2006. Shaokang has done some research in real-time image processing, face recognition, and medical image processing. He currently focuses on robust real-time face recognition in NICTA Queensland.

André C. Ponce de Leon F. de Carvalho received a BSc and MSc in computer science from the Universidade Federal de Pernambuco, Brazil. He received a PhD in electronic engineering from the University of Kent, UK. Prof. André de Carvalho is full professor at the Department of Computer Science, University of São Paulo, Brazil. He has published around 50 journal and 200 conference refereed papers. His main interests are machine learning, data mining, bioinformatics, evolutionary computation, bio-inspired computing, and hybrid intelligent systems.

Marcilio C.P. de Souto received a BSc in computer science from the Federal University of Rio Grande do Norte, Brazil, in 1992; an MSc in computer science from the Federal University of Pernambuco, Brazil, in 1995; and a PhD in electrical engineering (artificial neural networks) from Imperial College, University of London, U.K., in 1999. He worked as a visiting professor at the Center of Informatics of the Federal University of Pernambuco for three years, and in 2003, he spent one year as an advanced researcher at the Institute of Mathematics and Computing at the University of São Paulo at São Carlos,

Brazil. Currently, he works as an assistant professor in the Department of Informatics and Applied Mathematics at the Federal University of Rio Grande do Norte. His main research topics involve computability of artificial neural networks and machine learning techniques for bioinformatics.

Peter Duell is a PhD student in the School of Computer Science at the University of Birmingham, UK. His research interests include automatic design of neural network ensembles, evolution of ensembles, role of diversity, structures, and combination schemes in ensembles. He is specifically interested in the evolution of good ensembles for a range of various problems without the need for time-consuming and complex parameter setting. He has published a number of papers in conferences such as CEC'06 and ESANN'06.

Katti Faceli received a BSc in computer science in 1998, an MSc in computer science in 2001, and a PhD in computer science, all from the University of Sao Paulo, Brazil. Currently, she is a post-doc in the Department of Computer Science at the University of São Paulo, Brazil. Her main research interests include machine learning, hybrid intelligent systems, cluster analysis, ensembles, feature selection, and bioinformatics.

Bogdan J. Falkowski received an MSEE and PhD in electrical and computer engineering from Warsaw University of Technology, Warsaw, Poland, and Portland State University, Portland, Oregon, in 1978 and 1991, respectively. His industrial experience includes research and development positions at several companies. Since 1992, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently an associate professor. His research interests include very large-scale integrated systems and design, switching circuits, testing, digital signal and image processing, and design of algorithms. He has published three book chapters and more than 300 refereed journal and conference papers in this area. Dr. Falkowski received the Hartree Premium Award that recognizes the outstanding paper published in 2001 in IEE Proceedings on Computers and Digital Techniques in 2002. He is listed in the Marquis *Who's Who in Science and Engineering.*

Eric Granger joined the École de Technologie Supérieure (ETS) as an assistant professor in January 2004, where he has since been developing applied research activities in the areas of machine leaning, artificial neural networks, patterns recognition, signal processing, and microelectronic systems integration. Prior to joining ETS, he worked for three years in R&D at Mitel Networks in Ottawa, where he designed algorithms and ASIC circuits to support cryptographic and voice compression functions for Internet protocol platforms. From January 1999 to March 2001, Dr. Granger was a defense scientist in the electronic warfare (EW) section at DRDC Ottawa, and investigated machine leaning and neural network models for the classification of complex radar signals in radar electronic support measures (ESM) systems. Dr. Granger obtained a PhD in electrical engineering from the École Polytechnique de Montréal in 2001. His application domains include military surveillance (recognition of radar emitters in ESM systems), biometric identification and verification (recognition of faces in video and signatures), and intrusion detection in computer and network security.

Prithwijit Guha graduated from Jadavpur University, India, in 1999, with a BE in electrical engineering. Subsequently, he completed his MTech in signal processing (EE) from IIT Kanpur with a dissertation on automated visual inspection of steel surface defects in 2001. He worked as a research

associate in the BlueC project at BIWI, ETH Zurich for one year (2001–2002). He is currently pursuing a PhD from IIT Kanpur in the Department of Electrical Engineering. His research interests include computer vision and machine learning, with applications in surveillance and robotics.

Hui-Xing Jia received a BS in electronic engineering in 2003 from Beijing Jiaotong University. He is currently a PhD candidate in the Department of Electronic Engineering at Tsinghua University. His research interests include pattern recognition, computer vision, and intelligent vehicle. He is now working on object detection from image and video streams.

Graham Leedham is currently professor of computer engineering and information technology at the University of New South Wales in Singapore. He has more than 22 years of post-PhD university research, teaching, and administration experience at Essex University (1984–1994), Nanyang Technological University (1994–2006), and University of New South Wales in Singapore since 2006. His research is primarily concerned with the definition and real-time embedded implementation of image processing, computer vision, and pattern recognition algorithms. He also has research interests in human-computer interaction and computer animation. His work covers both software and hardware implementations of algorithms, encompassing his teaching interest in embedded real-time systems. Much of his early research activities were associated with document image processing, including online and off-line handwriting analysis and recognition for shorthand, postal addresses, and handwritten gestures. This research also extended to other image processing and pattern recognition areas such as medical and satellite image processing. His recent research is focused on the use of computers and novel sensors in forensic analysis, biometrics, and intelligent surveillance for security and media applications. He has published more than 150 peer-reviewed research articles in international journals, conferences, and edited books, and has co-edited two research books on image processing and analysis of handwriting.

Brian C. Lovell was born in Brisbane, Australia, in 1960. He received a BE in electrical engineering in 1982, a BSc in computer science in 1983, and a PhD in signal processing in 1991, all from the University of Queensland (UQ). Professor Lovell is research leader in National ICT Australia, and research director of the Intelligent Real-Time Imaging and Sensing Research group in the School of ITEE, UQ. He was president of the Australian Pattern Recognition Society from 1995 to 2005, senior member of the IEEE, fellow of the World Innovation Forum, fellow of the IEAust, and voting member for Australia on the governing board of the International Association for Pattern Recognition since 1998. Professor Lovell was technical co-chair of ICPR2006 in Hong Kong (computer vision and image analysis), and is program co-chair of ICPR2008 in Tampa, Florida. His research interests are currently focused on optimal image segmentation, real-time video analysis, and face recognition.

Tadeusz Łuba received an MS in electronics engineering from the Warsaw University of Technology, Poland, in 1971. He received a PhD and DSc from the same university in 1980 and 1988, respectively. In 1972, he joined the Institute of Telecommunications of the Warsaw University of Technology, where he has been a professor since 1993. Currently, he is the head of Telecommunications Fundamentals Department with the Institute of Telecommunications at WUT. His research interests include switching theory, CAD tools for logic synthesis and optimization, compilers for programmable logic devices, logic synthesis, and testing of digital circuits. Prof. Łuba has authored numerous publications in the area of logic synthesis. He is a member of program committees of several international and national conferences, and a member of the board of directors of EUROMICRO.

Vamsi Krishna Madasu is a research fellow in the School of Engineering Systems at Queensland University of Technology in Brisbane, Australia. He is working in the field of smart systems, in which he is developing new image processing and computer vision technologies for diverse applications. Vamsi has a PhD in electrical engineering from the University of Queensland, Australia, and a Bachelor's of technology with distinction in electronics & communication engineering from JNTU, India. His main research focus has been biometrics-based identity verification with applications to user authentication and bank check processing. Vamsi is a member of IEEE, APRS, and IEEE Computer Society.

Patrick Maupin joined Defence Research and Development Canada (Valcartier) in 2001, where he conducts research and experimentation on pattern recognition techniques applied to situation analysis as well as on the formalization of the situation analysis processes.

Amitabha Mukerjee (Ph.D., University of Rochester, 1986) is a professor in computer science and engineering at IIT Kanpur. He works in the areas of visual surveillance, cognitive learning from perception, and interfaces between vision and language. He has more than 100 papers in refereed journals and conferences and one book. He holds two patents in the area of interactive devices for hands-on learning. He is currently on the editorial board of the journal *Spatial Cognition and Computing* (Kluwer) and a number of conferences.

Marimuthu Palaniswami received a BE (Hons) from the University of Madras; an ME from the Indian Institute of Science, India; an MEngSc from the University of Melbourne; and a PhD from the University of Newcastle, Australia, before rejoining the University of Melbourne. He has been serving the University of Melbourne for more than 16 years. He has published more than 180 refereed papers. His research interests include SVMs, sensors and sensor networks, machine learning, neural network, pattern recognition, signal processing, and control. He was given a Foreign Specialist Award by the Ministry of Education, Japan, in recognition of his contributions to the field of machine learning. He served as associate editor for journals/transactions, including *IEEE Transactions on Neural Networks* and *Computational Intelligence for Finance*. He is also the subject editor for the *International Journal on Distributed Sensor Networks.*

Vladimir Pervouchine was born in Orenburg, Russia, and received his BSc and MSc from Moscow Institute of Physics and Technology (MIPT), Russia, in 2001 and 2003. respectively. In 2002, he joined the graduate student exchange between MIPT, Russia, and Nanyang Technological University (NTU), Singapore, and in 2003 converted to a PhD program in NTU. He received a PhD from the School of Computer Engineering, NTU (Singapore) in 2006. From 2005 to 2007 he worked as a postdoctoral research fellow for Forensics and Security Lab, NTU. In 2007, he joined the University of New South Wales (Singapore campus), Division of Engineering, Science, and Technology, as an assistant professor of computer science and engineering. His research interests are in image and signal processing, pattern recognition, and computer vision.

Tuan Pham received a PhD in civil engineering from the University of New South Wales in 1995. His research interests are bioinformatics, biomedical informatics, image analysis, geostatistics, pattern recognition, and fuzzy-set algorithms. He has published two books and more than 120 peer-review

journal and conference papers in the fields of engineering numerical analysis, soft computing, pattern recognition, image processing, speaker recognition, bioinformatics, and computational biology. Dr. Pham is a senior member of the IEEE and Director of the Bioinformatics Applications Research Center at James Cook University.

Mariusz Rawski received an MSc in electronics engineering from the Warsaw University of Technology, Poland, in 1995. He received a PhD from the same university in 2000. Currently, he is assistant professor in the Department of Electronics and Information Technology at the Warsaw University of Technology. His research interests include logic synthesis of digital circuits, CAD tools for logic synthesis and optimization, and compilers for programmable logic devices. Dr. Rawski has authored numerous publications in the area of logic synthesis.

Dominique Rivard received a BEng degree in electrical engineering from the École de Technologie Supérieure in 2004, and is currently in the doctoral program. He is engaged in research on the application of pattern recognition to off-line verification of handwritten signatures.

R. Sabourin joined the Physics Department of Montreal University in 1977. There he was responsible for the design, experimentation, and development of scientific instrumentation for the Mont Mégantic Astronomical Observatory. His main contribution was the design and implementation of a microprocessor-based fine tracking system combined with a low-light level CCD detector. In 1983, he joined the staff of the École de Technologie Supérieure, Université du Québec, in Montréal, where he cofounded the Department of Automated Manufacturing Engineering, and is currently full professor and teaches pattern recognition, evolutionary algorithms, neural networks, and fuzzy systems. Dr. Sabourin is the author and co-author of more than 200 scientific publications, including journals and conference proceedings. His research interests are in the areas of handwriting recognition and signature verification for banking and postal applications.

Henry Selvaraj earned a Master's and PhD from the Warsaw University of Technology in 1986 and 1994, respectively. From 1994 to 1998, he was a member of the faculty of computing and information technology, Monash University, Melbourne, Australia. Since 1998, he has been a faculty member in the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas. His research interests are logic synthesis, programmable logic devices, digital signal processing, and systems engineering. He has published more than 80 scientific papers in refereed journals and proceedings.

Ting Shan joined NICTA Queensland Lab as a researcher in the Safeguarding Australia Program-Smart Sensors Project in January 2007, after he completed a PhD at the University of Queensland and NICTA. Shan's research interest is in the field of computer vision and pattern recognition, particularly in face detection, pose variation compensation, multiview face recognition, still-image, and video-based face recognition. He has received two patents in the areas of face recognition, and has published 10 papers in book chapters and international conferences.

Alistair Shilton received a combined BSc/BEng from the University of Melbourne, Australia, in 2000, specializing in physics, applied mathematics, and electronic engineering. He completed a PhD from the University of Melbourne in 2006 for his thesis titled *Design and Training of Support Vector Machines.*

He is currently working as a researcher at the University of Melbourne. His research interests include machine learning (specializing in support vector machine theory), sensor networks, and localization.

Fok Hing Chi Tivive received a BEng with first-class honors from Edith Cowan University, Australia, in 2001, and a PhD from the University of Wollongong, Australia, in 2006. He is currently a research fellow at the University of Wollongong. His general research interests are in the areas of image and video processing, neural networks and machine learning, and pattern recognition.

Seiichi Uchida received a BE, ME, and Dr.Eng from Kyushu University in 1990, 1992, and 1999, respectively. From 1992 to 1996, he joined SECOM Co., Ltd., Tokyo, Japan, where he worked on speech processing. Since 2002, he has been an associate professor at faculty of information science and electrical engineering, Kyushu University. His research interests include pattern analysis and speech processing. Dr. Uchida is a member of IEEE, IEICE, IPSJ, and ITE.

K.S. Venkatesh graduated from BMS Engineering College under Bangalore University in 1987. Subsequently, he completed an MTech in communication systems with a dissertation on decision-making over incomplete orders in 1989 from IIT Kanpur, and his PhD from the same institute in 1995 with a thesis titled *Signals and Systems on Sets.* He joined IIT Guwahati (India) and served there from 1995 to 1999, then moved to IIT Kanpur, India, where he is currently employed as an assistant professor. His interests include image and video processing, computer vision with applications in robotics, surveillance, metrology, human computer interfacing, biomathematical modeling, and abstract system theory.

Lipo Wang received a BS from the National University of Defense Technology, China, and a PhD from Louisiana State University, Baton Rouge. Currently, he is associate professor at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Previously, he was a tenured faculty member in computing at Deakin University, Australia. His research interests are in theory of computational intelligence and its applications to optimization and data mining. He is author or co-author of more than 50 journal publications and 60 conference presentations; he holds a U.S. patent and is co-editor of two books and five conference proceedings. He is an associate editor/editorial board member of *Soft Computing* and *Knowledge and Information Systems.* Dr. Wang is an associate editor/editorial board member of *IEEE Transactions on Neural Networks.* He has been or is general chair for four international conferences and has served or is serving on numerous conference committees. He is president of the Asia-Pacific Neural Network Assembly.

Hong Yan received a PhD from Yale University. He has been professor of imaging science at the University of Sydney and is currently professor of computer engineering at City University of Hong Kong. His research interests include image processing, pattern recognition, and bioinformatics. He was elected a fellow of the Institute of Electrical and Electronic Engineers (IEEE) for contributions to image recognition techniques and applications, and a fellow of the International Association for Pattern Recognition (IAPR) for contributions to document image analysis. He is also a fellow of the Institution of Engineers, Australia (IEAust), and a member of the International Society for Computational Biology (ISCB).

Xin Yao is professor of computer science at the University of Birmingham, where he directs the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA). He is an IEEE fellow, a distinguished lecturer of the IEEE Computational Intelligence Society, and a distinguished visiting professor of the University of Science and Technology of China. He is the editor-in-chief of *IEEE Transactions on Evolutionary Computation*, an associate editor or an editorial board member of 10 other international journals, the editor of the World Scientific book series on Advances in Natural Computation, and a guest editor of several journal special issues. He won the prestigious IEEE Donald G. Fink prize paper award in 2001 and several other best paper awards. He has been invited to present 50 keynote or plenary speeches at conferences worldwide. He has more than 200 research publications. His research interests include evolutionary computation, simulated annealing, global optimization, neural network ensembles, data mining, computational time complexity of evolutionary algorithms, complex adaptive systems, and real-world applications. His work has been supported by AWM, EPSRC, EU, Royal Society, National Natural Science Foundation of China, Chinese Academy of Sciences, Honda, Marconi, BT, Thales, Severn Trent Water, and others. He has developed a number of novel algorithms for evolutionary artificial neural networks (EANNs) and artificial neural networks (ANNs) ensembles that generalize well. He was among the first in the world to study computational time complexity of evolutionary algorithms (EAs) with population size greater than one. Some fundamental results have been proved, which show how population size can influence the time complexity of EAs for given problems. He proved two important theorems on evolutionary stable strategies in the $N (N>2)$ player iterated prisoner's dilemma with and without noise. He proposed fast evolutionary programming (FEP) and improved FEP (IFEP) for unconstrained optimization and stochastic ranking for unconstraint handling. He has filed three EU patents from his joint work with Marconi and Honda. He started a novel M.Sc. in the Natural Computation program at the University of Birmingham in 2000.

Donggang Yu received a PhD in faculty of information, communication technologies from Swinburne University of Technology, Melbourne, Australia. His research interests are image processing, pattern recognition, and bioinformatics. He is doing research in the Bioinformatics Applications Research Center at James Cook University.

Yu-Jin Zhang is a professor in the Department of Electronic Engineering at Tsinghua University. His research interests are mainly in the area of image engineering, which includes image processing, image analysis, and image understanding, as well as their applications. He has published extensively on image engineering and related areas, with more than 250 research papers, two monographs (*Image Segmentation* and *Content-Based Visual Information Retrieval* [Science Press]), and two edited books (*Advances in Image* and *Video Segmentation and Semantic-Based Visual Information Retrieval* [IRM Press]). His homepage is http://www.ee.tsinghua.edu.cn/~zhangyujin/.

Haishan Zhong received a BEng and MEng from the University of Electronics, Science, and Technology (China) in 2003 and 2005, respectively. From 2005 to 2007, Zhong was doing MEng by research in the School of Computer Engineering, Nanyang Technological University (Singapore). Currently, she works as an R&D engineer for Panasonic Singapore Laboratories. Her interests are in speech and signal processing.

Nina Zhou received a BS from Wuhan University of Science and Technology in 2000, and an MS from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003. She is currently a third-year PhD candidate at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. She received the Singapore Government Scholarship from July 2004 to July 2007. Her current research areas include data mining, feature selection, classification, and bioinformatics.

Index

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W