SCANNED DOCUMENT ENHANCEMENT BASED ON FAST TEXT DETECTION

Yue Wang[b], Jobin J Mathew[b], Eli Saber[ab], David Larson[c], Peter Bauer[c], George Kerby[c], Jerry Wagner[c]

[a]Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology
[b]Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology,
Rochester, NY, USA 14623;
[c]Hewlett Packard Company – Color & Imaging, Boise, ID, USA 83714;

ABSTRACT

In this paper, we propose a Fast Text Detection (FTD) algorithm to automatically identify the text and line art pixels in scanned monochromatic and color documents for enhancing text in printed documents. To simulate the scanning operation, the input image is converted from the RGB space to a Luminance-Chrominance space and is then divided into individual strips. This is followed by a low pass filtering operation on the input strips for removing high frequency noise and to reduce processing time. An edge detection scheme is then applied to the blurred strips to generate corresponding edge-strips. The edge-strips are then subjected to a morphological operation and an edge-based adaptive thresholding simultaneously and the resulting two output strips are merged together to obtain the final candidate text plane. In the text enhancement operation, the original image is first converted into LCH space and the pixels corresponding to the detected text pixels are enhanced via a clipping operation in the L and C channel. This algorithm is highly efficient in terms of memory usage and processing speed and is thus suited to run effectively in low-cost embedded devices.

Index Terms— Text Detection, strip, enhancement, scanners, edge based

1. INTRODUCTION

Image layout analysis is a fundamental principle in many applications related to processing document images. For example, extracting text pixels from images is used to convert scanned documents into electronic files, also known as optical character recognition (OCR). The extracted text plane also provides an opportunity to enhance the image quality in the printing process along with other applications such as document tagging, keyword search, etc.

Khedekar *et al.* [1] presents a top-down, projection-profile based algorithm to separate text blocks from image blocks. However, this approach lacks the ability to handle non-rectangular shaped text blocks, sparse images and horizontal lines.

Erkilinc et al. [2] proposed a multiple-step analysis algorithm to classify a scanned document into different regions such as text, images, etc. This algorithm uses several image processing techniques such as the wavelet transform, Markov random field based segmentation and K-means clustering. Although this methodology averages an 89% accuracy, the computational cost is relatively high.

In [3, 4, 5], several texture-based approaches are used for detecting and separating text pixels from images. These approaches generally fall into two categories: pixel-based and block-based. The papers [6, 7] have proposed SVM classifier and Neural network training scheme to segment text from video frames.

Although a great deal of text detection algorithms have been developed, it is still a challenge to perform text recognition in real-time when hardware resources are limited. For instance, in a real-time embedded scanning device, memory resources are always limited, which means only a part of image could be stored at a time. Thus, an analysis on a full document page [2] may not be a practical option in these scenarios. Another constraining factor is the increasing complexity of computation under limited resources which is directly linked to the speed/execution time.

Prakash et al. [8] proposed a method based on Discrete Cosine Transform (DCT) to localize text in digital videos. Pietikainen et al. [9] provided a texture-based approach based on the edge information. Both these approaches have shown a reasonable performance in identifying the text while maintaining a reasonable computational complexity. However, the disadvantage of these methods is that, some high-frequency regions (images) in the document are identified as text. For the purpose of enhancement, such false-positive regions might lead to some visible defects in enhanced output.

This paper proposes a strip-based methodology to identify text and line art pixels from monochrome or color documents which are scanned at 300 DPI. The generated output text plane will be used to enhance the text quality in the printing process. Due to the low memory usage and fast

processing time, this algorithm can be easily be integrated onto a real-time embedded system such as those found in all-in-one scanners/printers. After testing the algorithm on a low-cost embedded device, the processing speed was found to be around 30 to 40 pages per minute.

The algorithm details will be described in section 2. Results and enhanced output images are discussed and presented in section 3. Finally, conclusion are drawn based on these results in section 4.

2. METHODOLOGY

The primary objective of the Fast Text Detection (FTD) algorithm is to identify text and line art pixels from scanned documents for the purpose of text enhancement. Considering the scanning operation in real-time, the lines/rows of a page are read by sensors one at a time and hardware can only store several lines of the input page. Therefore, the algorithm must be strip-based and causal in nature where, the system can process each strip individually and independent of other strips. The flowchart of the proposed FTD algorithm is shown in Figure 1.

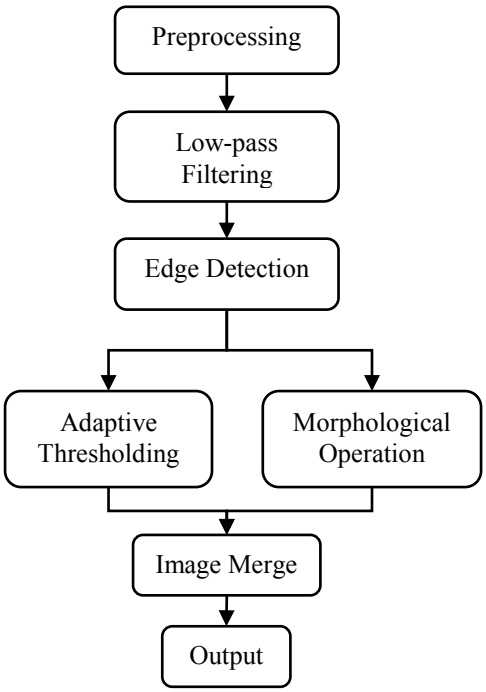


Fig. 1 Flowchart of FTD algorithm

2.1. Preprocessing

To simulate the scanning operation, input images needs to be converted from RGB to LCH color space. Since the edge information is primarily present in the Luminance plane, only the L channel is used in the FTD algorithm, while chrominance channels are not.

The Luminance plane is then divided into individual strips of specific size. The size of each strip is selected carefully in such a way that, it is neither too small to convey any relevant text information nor too large to surpass the hardware buffer size. For the purpose of identifying text up to 24pts in 300dpi documents, we select and extract 128 lines for each strip from the full page document. Also, to avoid any errors caused by boundary alignment, each strip includes 10 lines from the previous and following strips respectively. For the top lines of the first strip and the bottom lines of the last strip a replicate padding is applied.

2.2. Low-pass Filtering

To reduce the computational complexity and eliminate high-frequency noise, the input strip is down sampled by a factor of 2 followed by applying a 3 by 3 average filter to it. Implementing a 2-D filtering by the method of convolution or Fourier transform might lead to an extra computational cost. Thus, to optimize this filtering process, we perform average filtering *via* the integral image technique.

Integral image, also known as summed area table is a highly effective and fast way to calculate the sum of values in a rectangular data structure which was introduced by Viola and Jones in [9]. The value at any location *(x, y)* in the integral image is defined according to the follow equation:

$$I(x,y) = \sum_{x' \le x, y' \le y} i(x', y') , \tag{1}$$

where *I(x, y)* is the integral image and *i(x',y')* is the original image. Once the integral image is generated, the sum of a rectangular region with upper left corner (x_1, y_1) and lower right corner (x_2, y_2) is calculated via the following equation:

$$\sum_{x=x_1}^{x_2} \sum_{y=y_1}^{y_2} i(x,y) = I(x_2, y_2) - I(x_2, y_1 - 1) - I(x_1 - 1, y_2) + I(x_1 - 1, y_1 - 1) \tag{2}$$

Finally, the average filtered image is obtained by dividing the sum of each rectangular region by its size. While the 2-D convolution method is complex and requires multiple passes, the integral image method is efficient and requires only one pass over the original image.

2.3 Edge Detection

To achieve better hardware performance, we avoid using any edge detection operators such as Sobel or Prewitt. In the FTD algorithm, edge detection is implemented by combining the horizontal and vertical gradient components together, thus generating a binary edge map.

2.4 Morphological Operation

Text pixels in the edge map could be considered as small regions which are closed by boundaries. To enlarge these regions, an image closing operation with a 5 by 5 structuring element size is applied to the edge map. The output of the morphological operation might include regions which are non-text or false-positives. These false-positive regions are eliminated in the next step, when the morphological output is merged with the output from adaptive thresholding module.

2.5 Edge-Based Adaptive Thresholding

An edge-based adaptive thresholding technique (EBAT) is used in conjunction with the morphological operation module, which is intended to increase the accuracy of detected text plane.

Since majority of the text information in a document can be associated with edges, the pixel intensities corresponding to these edges is first identified and then used to estimate the intensity of text pixels in this specific region. Therefore, the basic concept of EBAT is to calculate the average intensity of text pixels in a region via an edge map followed by setting the average intensity value as the local threshold. This operation is described by the followed equation:

$$F = \frac{1}{N} \sum_{(x,y)\in E} I(x,y) \tag{3}$$

where E represents the edge pixels, $I(x, y)$ represents the intensity value at the location (x, y), and N is the number of edge pixels present in this region.

To handle the background variations across the whole page, each individual input strip is divided into small blocks. A local threshold is calculated for each block using equation (3), which is used to threshold and convert these blocks to binary blocks. Considering the cases with light color text on dark color background, the total percentage of text pixels present in each thresholded block is calculated. We assume that text pixels are always the minority in a block, and if the percentage of text pixels in the thresholded block exceeds 50 percent, a NOT operation is applied to the thresholding block.

EBAT operation can effectively distinguish foreground from background (non-text). The foreground primarily represents text and line art pixels, but it may also contain features and high frequency regions from non-text regions.

2.6 Image merge

The final step of the FTD algorithm is to merge the output maps obtained from morphological and EBAT modules and thus generate the final text plane. The morphological plane and the EBAT plane constitutes the detected text regions along with some false positives. Since the false positives in these two binary text maps will be present in different locations, an AND operation between the two text planes can yield a final text plane with minimal false positives.

3. RESULTS & ENHANCEMENT EXPERIMENT

3.1 FTD Results

The proposed FTD algorithm is mainly used for document enhancement in scanning/printing and the primary goal in terms of false positives is to 'fail gracefully'. This concept stems from the fact that the false detection of a group of pixels in the document can result in a poorly enhanced output image. To this effect, emphasis was placed on minimizing the enhancement artifacts caused due to false classification.

(a) (b)

Fig. 2: (a) Original images, (b) FTD output text planes

The results of the proposed FTD algorithm on two scanned documents are shown in Figure.2. To meet the requirement of 'failing gracefully', our strategy is to include text pixels in its wholeness and thus avoiding any partially detected text lines/pixels. Meanwhile, the FTD algorithm labels strong edges and high frequency details from non-text

regions as false-positives. During the enhancement procedure, those regions will be enhanced as text as well. Since they represent edge components and line-art features, the visible flaws that will show up in the enhanced images will be minimal

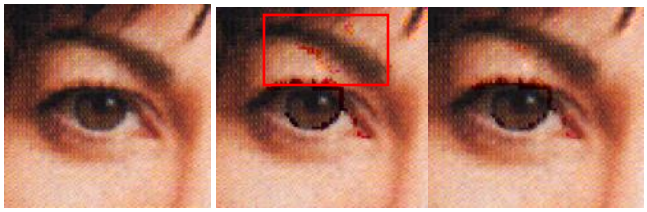
The images utilized for testing are obtained from a scanned document database provided by Hewlett Packard. We picked 14 images from this database as a subset for benchmarking and the ground truths were generated manually. This subset includes different documents from simple mono-text-documents to color-text documents containing images and varying backgrounds. The performance of the algorithm was determined by comparing the number of text pixels in the detected image with that of the corresponding ground truth. According the test, the proposed FTD algorithm has accuracies in the high 90s. The average accuracy of the algorithm on both color and mono images is 96.95%. And the false-positive rate is no higher than 4%. Numerical test results of the images in Figure 2 are shown in Table 1.

Image Name	False Positive (%)	Misses (%)	Hits (%)
Scotch	1.61	0.43	99.56
Boat	3.07	1.53	98.46

Table 1: Test results for FTD

3.2 Enhancement Experiment

The accuracies of the FTD algorithm indicates that it meets the requirement of "failing gracefully" and is highly effective for document enhancement. The image enhancement operation is accomplished in two steps: first, the source image is converted from RGB to CIE LCH color space. After that, in the luminance and chromatic channels, linear clipping operation is applied to the pixels identified as text pixels. Finally, the image in CIE LCH space is converted back to RGB color space to render the enhanced image.



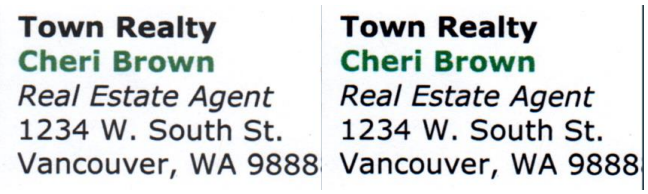
(a) (b) (c)

Fig. 3: (a) Original Image, (b) Aggressive Enhancement, (c) Milder Enhancement

To evaluate the enhancement quality, enhancements at two levels with different clipping values are applied to the original image. According to the results shown in Figure 3, with aggressive enhancement, visible artifacts mostly show up in the false positive regions. With milder enhancement, these artifacts become barely noticeable. Another example is shown in Figure 4 to illustrate that milder enhancement is still effective depending on the content of the document being enhanced.

3.3 Hardware Performance

To test the performance, the FTD algorithm is simulated in the C programming language on a PC environment and also on a BeagleBoard, the latter of which simulates running on a low-cost embedded device. The BeagleBoard has an AM335x 1GHz ARM Cortex-A8 processer and a 512MB DDR3 RAM. During the test, the FTD algorithm processes 300 dpi full page documents at 35-40 pages per minute (PPM). Considering other tasks running in parallel, the proposed algorithm has the ability to handle no less than 10 pages per minute in a real product.



(a) (b)

Fig. 4: Milder Text Enhancement Result. (a) Original Image, (b) Enhanced Image.

4. CONCLUSION AND FUTURE WORK

The proposed FTD algorithm has the ability to identify text and line art pixels in documents scanned at 300 DPI and provide text planes for document enhancement. Based on our tests, few of the strong edges in the original documents are classified as text, but this does not contribute to any enhancement errors since the false-positive regions are generally unnoticeable after the enhancement. Additionally, the case of black and color text detection is tackled simultaneously in the same algorithm, with both cases having accuracies in the high 90s. Furthermore, this algorithm is robust for non-Latin languages, such as Chinese characters and Japanese Kanjis.

Currently, the proposed algorithm still has some difficulties to detect text on low contrast or strong background documents. It also suffers from few missed detections with thick strokes and large fonts. These shortcomings could be improved by using some hierarchical analysis and more sophisticated edge detection schemes.

5. ACKNOWLEDGEMENT

This research was supported by Hewlett Packard Company in collaboration with the Department of Electrical and Microelectronic Engineering at the Rochester Institute of Technology.

6. REFERENCES

[1] S. Khedekar, V. Ramanaprasad, S. Setlur and V. Govindaraju, "*Text-Image Separation in Devanagari Documents,*" In Proc.7th International Conference on Document Analysis and Recognition, ICDAR'03, pp. 1265-1269, 2003.

[2] M.S. Erkilinc, M. Jaber, E. Saber, P. Bauer and D. Depalov, "*Text, photo, and line extraction in scanned Documents,*" Journal of Electronic Imaging, Vol. 21(3), pp. 1265-1269, July 2012.

[3] D.F. Dunn and N.E. Mathew, "*Extracting color halftones from printed documents using texture analysis,*" Pattern Recognition, 33(3), pp. 445-463, 2000.

[4] A.K. Jain and Y. Zhong. *"Page segmentation using texture analysis"*. Pattern Recognition, 29(5):743–770, 1996.

[5] M. Murgu´ia. *"Document segmentation using texture variance and low resolution images"*. In Proc. of 1998 IEEE Southwest Symposium on Image Analysis and Interpretation,Tucson, Arizona, USA, pages 164-167, 1998.

[6] K. Y. Jeong, K. Jung, E. Y. Kim and H. J. Kim, *"Neural network-based text location for news video indexing"*. In Proc. ICIP, pp 319-323, 1999.

[7] K.I.Kim, C.S.Shin, M.H.Park, and H.J.Kim. *"Support Vector Machine – based text detection in digital video"*. In proc.IEEE signal Processing Society Workshop, pages 634-641, 2000.

[8] S. Prakash and M. RaviShankar. *"Multi-Oriented Video Text Detection and Extraction using DCT feature extraction and Projection based rotation calculation"*. Advances in Computing, Communications and Informatics (ICACCI), pages 714-718, 2013

[9] M. Pietikainen, O. Okun. *"Edge-based method for text detection from complex document images"*. In Proc. 6th international conference on Document Analysis and Recognition (ICDAR), pages 286-291, 2001.

[10] P. Viola and M. Jones. *"Robust Real-time Object Detection"*. Second International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing and Sampling. Vancouver, Canada, July 13, 2001.

[11] R.C. Gonzalez, and R.E. Woods, *Digital Image Processing,* Pearson Prentice Hall, 2008

[12] K.N. King, *C Programming, A modern Approach,* W.W. Norton, 2008