TEXTURE EDGE DETECTION BY FEATURE ENCODING AND PREDICTIVE MODEL

Jyh-Charn Liu and Gouchol Pok

Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

ABSTRACT

Texture boundaries or edges are useful information for segmenting heterogeneous textures into several classes. Texture edge detection is different from the conventional edge detection that is based on the pixel-wise changes of gray level intensities, because textures are formed by patterned placement of texture elements over some regions. We propose a prediction-based texture edge detection method that includes encoding and prediction modules as its major components. The encoding module projects n-dimensional texture features onto a 1-dimensional feature map through SOFM algorithm to obtain scalar features, and the prediction module computes the predictive relationship of the scalar features with respect to their neighbors sampled from 8 directions. The variance of prediction errors is used as the measure for detection of edges. In the experiments with the micro-textures, our method has shown its effectiveness in detecting the texture edges.

1. INTRODUCTION

Texture segmentation is a fundamental task in image processing applications such as computer vision, medical imaging and image retrieval from databases [4] [6]. In segmenting heterogeneous textures, even rough information on the texture boundaries is useful and complementary to the approach based on the pixel-labeling according to the class-specific features. Most of the previous edge detection methods, however, are based on the pixel-wise changes of gray level intensities [1] [5], and thus are not appropriate for texture edge detection. Textures are formed by some patterned placement of texture elements consisting of a group of pixels. Hence, the definition of homogeneity should be extended from uniformness of gray levels to the similarity of patterns, and texture edge detection requires a mechanism which can discrimate the patterns of texture elements, not the dicontinuity of grey levels.

This paper presents a framework for detection of texture patches by detecting and representing the boundary points between different textures. The block diagram of the proposed framework is illustrated in Fig. 1. For texture edge detection, several issues need to be resolved: 1) what texture features are used, 2) how to represent texture features in a compact form , and 3) how to define the measure for discontinuity of texture patterns. In our approach, the texture features are first extracted as n-dimensional vectors by using Gabor filter banks, and then they are projected onto an 1-dimensional feature map $\mathcal{F}$, and encoded as scalars by replacing the Gabor features by the corresponding location index on $\mathcal{F}$. Then, the predictive relationship between an encoded feature and its neighbors is computed along eight directions by utilizing the function approximation property of the multilayer perceptron. The variance of eight prediction errors is used to represent the similarity of texture patterns around the given pixel. After Gaussian smoothing is applied over the variances to supress local fluctuation, the final edge map is produced by the Canny's edge-detection algorithm [1].

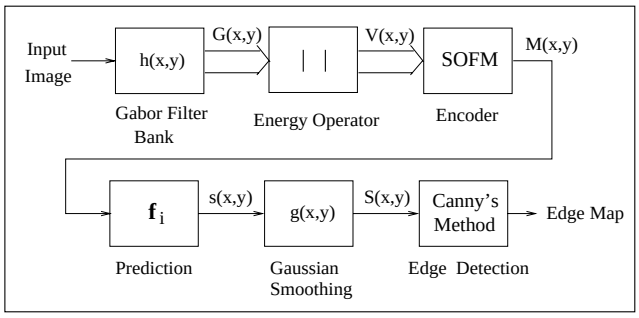

Figure 1: Texture Edge Detection System.

2. BACKGROUNDS

2.1. Gabor Filter

Gabor filters are efficient for extracting texture features based on localized spatial frequency information, which are useful for texture classification and segmentation [3] [4]. Texture features, in the Gabor filtering scheme, are obtained by convolving an image with the Gabor elementary functions. A Gabor elementary function is a 2-D Gaussian modulated by

complex sinusoids defined by,

$$h(x,y) = g(x',y') \exp[2\pi j(Ux + Vy)], \qquad (1)$$

where (x', y') denote rotated coordinates in the spatial domain, and (U, V) represents the filter location in frequency-domain with a center frequency $F = \sqrt{(U^2 + V^2)}$ and orientation $\theta = \arctan(U/V)$. $g(x', y')$ is a 2-D Gaussian function with the orientation angle ϕ and coordinates $(x', y') = (xcos\phi + ysin\phi, -xsin\phi + ycos\phi)$. It is defined by

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left\{-\frac{1}{2}\left[\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right]\right\}, \qquad (2)$$

where σ_x and σ_y are scale factors which characterize the spatial extent and bandwidth of the Gaussian filter. Usually, σ_x and σ_y have a common value $\sigma = \mu/\lambda$, where λ is the center frequency and μ is some constant. In this case, the Gabor elementary function (3) takes a simple form

$$h(x,y) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{x^2 + y^2}{2\sigma^2}\right\} \exp[2\pi j(Ux + Vy)] \qquad (3)$$

By convolving an image $I(x, y)$ with Gabor elementary functions, the outputs or coefficients of Gabor filters are obtained

$$V(x,y) = |I(x,y) * h(x,y)|, \qquad (4)$$

where $|\cdot|$ denotes the energy operator. In Gabor representation of texture features, a set of filters are chosen by varying the center frequency and orientation so that they cover the whole frequency domain. The texture features are defined by mn-dimensional vectors defined in Eq. (11).

2.2. Kohonen's Self-Organizing Feature Map

The Self-Organizing Feature Map (SOFM) algorithm proposed by Kohonen [9] is a computational model of self-organizing process based on competition between neurons. The SOFM provides a nonlinear projection of high dimensional input patterns onto a low dimensional output space, called a feature map, in such a way that topologically-ordered relations are preserved, i.e., similar input patterns are mapped to the neighboring units in the feature map. This implies that the probability distribution of input patterns is represented as locations in the feature map, and this propery is useful for processing vector-valued features in the transformed scalar space.

In the SOFM model, the network consists of one input unit, and output units on a 1- or 2-dimensional lattice. Let $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ be input vectors, and the weight of output unit i be represented by $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]$. When an input $\mathbf{x}$ is presented to the network, the winning unit is determined by

$$c(\mathbf{x}) = \arg\min_i ||\mathbf{x} - \mathbf{w}_i||, \; i = 1, 2, \ldots, N, \qquad (5)$$

where $c(\mathbf{x})$ represents the index of the winning unit for the input x and $||\cdot||$ denotes the Euclidean norm. Depending on the application, either the weight vector $\mathbf{w}_c$ or the index c could be used. In our work, n-dimensional inputs, which are Gabor features, are encoded as scalars with the indices.

Let $\Lambda_{c(\mathbf{x})}(t)$ denote the neighborhood kernel of the winning unit $c(\mathbf{x})$, which is a function of time, and it specifies the spatial extent of the neighborhood over the lattice at the time t. Usually $\Lambda_{c(\mathbf{x})}(t)$ is so chosen that it initially covers a wide region and then shrinks to zero in the radius with the time t. Once the winning unit and its neighbors are determined, the weights are updated as follows

$$\mathbf{w}_i(t+1) = \begin{cases} \eta(t)[\mathbf{x} - \mathbf{w}_i(t)] & \text{if } i \in \Lambda_{c(\mathbf{x})}(t), \\ \mathbf{w}_i(t) & \text{otherwise} \end{cases} \qquad (6)$$

where $\eta(t)$ is the learning rate at time t. By Eq. (6), the weight vector $\mathbf{w}_i$ of the winning unit $c(\mathbf{x})$ moves to the input vector $\mathbf{x}$, and the topological ordering property emerges from this process.

2.3. Function Approximation by Multilayer Perceptrons

Let $I(x, y)$ be a random variable representing the attribute of a pixel (x, y) in an image. In modeling the relationship of $I(x, y)$ with its neighbors, $\mathcal{N}_{xy} = \{I(x-i, y-j) \mid 0 \leq i, j \leq d\} - \{I(x, y)\}$, we assume that a function $f(\cdot)$ underlies the set of data pairs $\{I(x, y), \mathcal{N}_{xy}\}$,

$$I(x,y) = f(I_1, I_2, \ldots, I_n) + \epsilon, \qquad (7)$$

where $I_j \in \mathcal{N}_{xy}$, and ϵ is a random variable with zero-mean. This relationship can be modeled by a regression function which interpolates the regression surface $E[I(x, y)|\mathcal{N}_{xy}]$ from the observed data.

In the supervised learning approach, an approximation $\hat{f}(\cdot)$ to $f(\cdot)$ is obtained by iteratively modifying the system parameters in reponse to differences between the targets $I(x, y)$ and the outputs $\hat{f}(\cdot)$. By the *universal approximation theorem* ([8]), the multilayer perceptron networks can approximate any continuous multivariate function by superposition of sigmoidal basis functions $\phi(v) = 1/(1 + \exp(-v))$. The approximate $\hat{f}(\cdot)$ is represented by

$$\hat{f}(I_1, I_2, \ldots, I_n) = \sum_{j=1}^{M} v_j \phi\left(\sum_{k=1}^{n} w_{jk} I_k - w_{j0}\right), \qquad (8)$$

where w_{jk} and v_j are connection weights, and n and M are dimension of the input and output space, respectively [7].

3. TEXTURE EDGE DETECTION

3.1. Predictive Relationship

In order to detect the discontinuty of texture patterns, one needs to define a measure according to which textures are discriminated. We used the prediction errors in defining the measure of texture homogeneity. Given the attribute $M(x,y)$ of a pixel (x,y), if the neighbors from eight directions, as illustrated in Fig. 2, can predict accurately or consistently the value $M(x,y)$, then it is very probable that the textures around (x,y) belong to the same class. Here, "consistency" means that the eight prediction errors have similar values. When prediction errors are large but similar values, the source of errors might be attributed to the encoding scheme, not to the difference of texture patterns. Thus the variance of prediction errors is a better indicator of texture homogeneity than the mean of prediction errors.

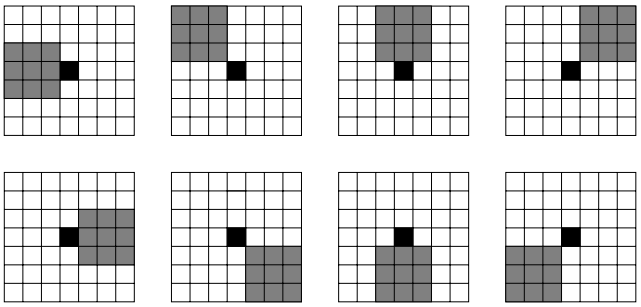

Figure 2: Eight sets of neighbors.

3.2. The Proposed Method

Refering to Fig. 1, the texture edge detection algorithm is summarized as follows.

Step 1. The input texture image $I(x,y)$ is convolved with each Gabor filter $h_{\lambda\theta}(x,y)$ from the filter bank $\{h_{\lambda_i\theta_j} \mid i = 1,\ldots,n,\ j = 1,\ldots,m\}$ to obtain the output

$$G_{\lambda\theta}(x,y) = h_{\lambda\theta}(x,y) * I(x,y), \tag{9}$$

where * denotes 2-dimensional convolution.

Step 2. The energy of each filter output is computed

$$V_{\lambda\theta}(x,y) = \mid G_{\lambda\theta}(x,y) \mid \ = \ \mid h_{\lambda\theta}(x,y) * I(x,y) \mid , \tag{10}$$

and thus mn-dimensional vectors $\mathbf{V}(x,y)$ are obtained

$$\mathbf{V} = [V_{\lambda_1\theta_1},\ldots,V_{\lambda_1\theta_m},\ldots,V_{\lambda_n\theta_1},\ldots,V_{\lambda_n\theta_m}], \tag{11}$$

where λ_i and θ_j represent center frequency and filter orientation, repectively.

Step 3. A 1-dimensional feature map $\mathcal{F}$ over the vectors $\{\mathbf{V}(x,y)\}$ is generated by the Kohonen's SOFM algorithm. As illustrated in the previous section, the probability distribution of $\{\mathbf{V}(\mathbf{x},\mathbf{y})\}$ is preserved on $\mathcal{F}$. For each pixel (x,y), the scalar index $M(x,y)$ of the reference vector closest to $\{\mathbf{V}(\mathbf{x},\mathbf{y})\}$ is assigned

$$M(x,y) = \arg\min_i \ ||\mathbf{V}(x,y) - \mathbf{w}_i|| \ \text{ for all } \mathbf{w}_i \in \mathcal{F}. \tag{12}$$

In this way we transformed the vector image to a scalar image.

Step 4. For each $M(x,y)$, its eight sets of neighbors are defined by $\{\mathcal{N}^i_{xy}\}^8_{i=1}$, where i is a direction index as shown in Fig. 2. The neighbors from the direction i are arranged as a vector $[M^i_1,\ldots,M^i_p]$, $M^i_j \in \mathcal{N}^i_{xy}$, and multilayer perceptrons are trained to obtain the approximation functions $\{f^i\}^8_{i=1}$ with the relations,

$$M(x,y) = f^i(M^i_1,\ldots,M^i_p) + \epsilon^i, \ \text{for } i = 1,\ldots,8. \tag{13}$$

The predictive relationship between $M(x,y)$ and its neighbors $\mathcal{N}^i_{xy}$ is estimated by the prediction error

$$e^i(x,y) = M(x,y) - f^i(M^i_1,\ldots,M^i_p) \tag{14}$$

The sample variance of the prediction errors

$$s^2(x,y) = \frac{\sum^8_{i=1}(e^i(x,y) - \mu(x,y))^2}{8}, \tag{15}$$

where $\mu(x,y)$ is the mean of $\{e^i(x,y)\}^8_{i=1}$, are assigned to the pixel (x,y).

Step 5. Over the sample variances $s^2(x,y)$, the Gaussian smoothing is performed to remove local fluctuation effect, and *variance image* $S(x,y)$ is obtained. This step is similar to noise supression by lowpass filtering.

Step 6. Following the Canny's edge-detection method, grdient operator is applied to the output of step 5, and by thresholding the gradient magnitude defined by

$$F = \left[\left(\frac{\partial S(x,y)}{\partial x}\right)^2 + \left(\frac{\partial S(x,y)}{\partial y}\right)^2\right]^{1/2}, \tag{16}$$

the edge map is produced.

4. EXPERIEMENTS

The proposed approach was tested on micro-texture images composed of heterogeneous textures such as fabric, water, and sand textures [10]. As texture features, 24 coefficients of Gabor filters are obtained with four scale factors ($\sigma = 2, 4, 6,$ and 8), and six orientation angles ($\theta = 0°, 30°, 60°,$

90°, 120°, and 150°). The 24-dimensional vectors are encoded with the corresponding indices of the feature map of size 256, and then eight 9-15-1 multilayer perceptrons are trained to predict the center value from the neighbors along the eight directions as illustrated in Fig. 2. The detection results, after Gaussian smoothing with $\sigma = 5.0$, are shown in Fig. 4 with various thresholding values.

5. REFERENCES

[1] J. Canny, "A Computational Approach to Edge Detection", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 8, no. 6, 1986, pp 679-6

[2] J. S. Chen and G. Medioni, "Detection, Localization, and Estimation of Edges," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 2, 1989, pp 191-198.

[3] J. Daugman, "Uncertainty Relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters," *J. Opt. Soc. Amer. A*, vol. 2, July 1985, pp 1160-1169.

[4] D. Dunn and W. Higgins, "Optimal Gabor Filters for Texture Segmentation," *IEEE Trans. Image Proc.*, vol. 4, no. 7, 1995, pp 947-964.

[5] P. H. Gregson, "Using Angular Dispersion of Gradient Direction for Detecting Edge Ribbons", *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 15, no. 7, 1993, pp 682-696.

[6] J. F. Haddon and J. F. Boyce, "Image Segmentation by Unifying Region and Boundary Information," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 12, no. 10, 1990, pp 929-948.

[7] S. Haykin, Neural Networks : A Comprehensive Foundation, NewYork: Macmillan, 1994.

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward network are universal approximators," *Neural Networks*, vol. 2, 1989, pp 359-366.

[9] T. Kohonen, "Self-Organization and Associative Memory," 3rd ed. New York: Springer, 1989.

[10] Collection of Microtextures, **http://www-dbv.cs.uni-bonn.de/image/browse**.

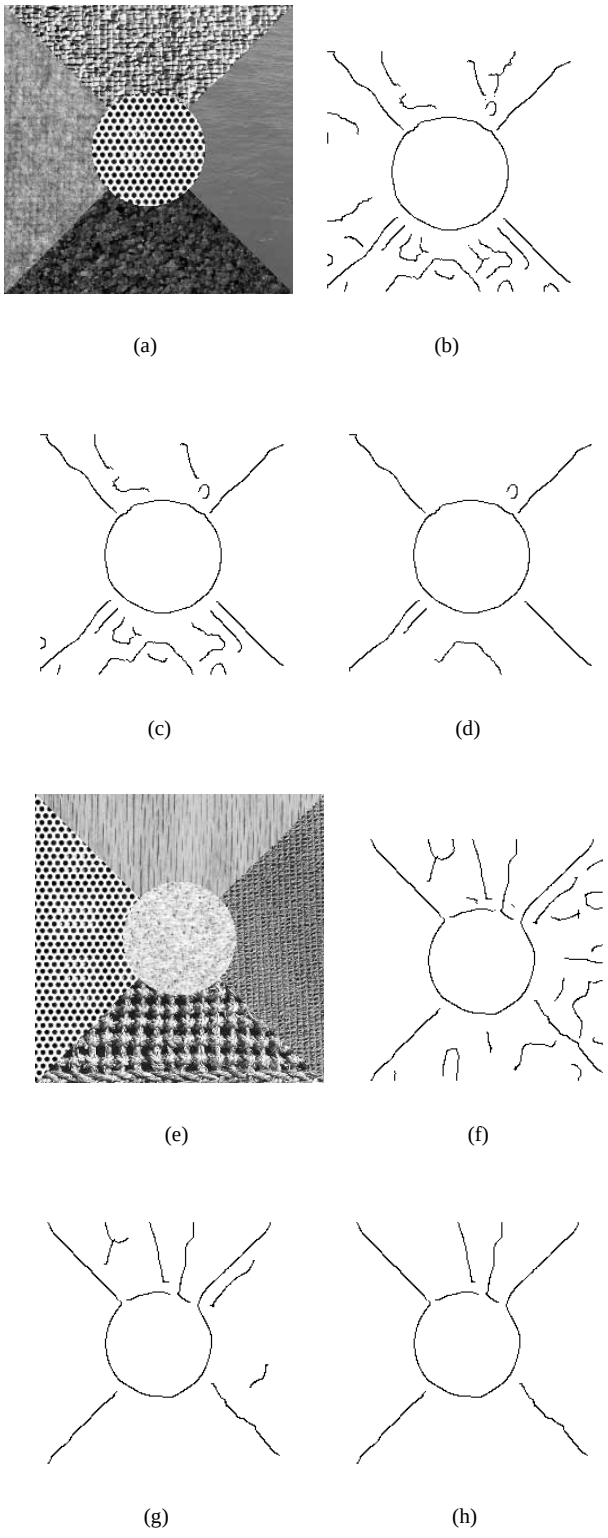

Figure 3: (a) Test image 1, and texture edge detection results by thresholding at (b) 0.7, (c) 0.75, (c) 0.8. (e) Test image 2, and results by thresholding at (f) 0.6, (g) 0.7, (h) 0.75.