Color Texture Segmentation for Clothing Based on Finite Prolate Spheroidal Sequences

CHIH-CHUNG CHIEN AND LING-LING WANG[1,*]
[1]*Department of Information Communication, Asia University, Taiwan*

ABSTRACT

By means of a computer-aided fashion design system, a fashion designer can easily make changes to the various colors, textures and pattern prints of clothing design, and thus need not draw on paper a great number of drafts. One of the most difficult tasks in a computer-aided fashion design system is to separate the clothing of interest on a model from the background, so that changes can be applied to the material. There exist folds, shadows, diverse textures, etc. on the clothing which make the segmentation work difficult. In this paper, a color texture segmentation method for clothing segmentation is proposed. Color quantization is first performed to reduce the number of colors and shadow/highlight effects on the image. The color texture features are then extracted based on the finite prolate spheroidal sequences. By these features, a hierarchical coarse-to-fine segmentation method is used to separate the clothing from the backgrounds. Finally, post-processing is applied to obtain a smooth clothing boundary. Satisfactory experimental results have been achieved using the proposed approach.

Key words: finite prolate spheroidal sequences, color quantization, texture segmentation, local centroid clustering.

1. INTRODUCTION

A traditional fashion designer has to draw a great number of drafts in order to accomplish an ideal style. Howerver, due to the rapid development of computer software, a computer-aided fashion design system has become feasible. Through the computer program's cut and paste commands (functions), when colors or pattern prints of the designed clothing are to be modified, it is not necessary to draw a new draft. Only the part to be modified is cut and pasted on the computer. Therefore, much drawing work on paper can be saved, and appropriate colors or pattern prints on the designed clothing can be determined quickly according to the customer model. Better performance can thus be achieved.

In a computer-aided fashion design system, one of the most difficult tasks is to separate the desired clothing from backgrounds automatically. Few commercial image processing or image editing packages can perform the segmentation well. For some packages, the users even have to specify by hand the clothing boundary. Some other packages might provide an automatic segmentation function, but the segmentation results are unsatisfactory. There are folds, shadows, diverse textures, etc. on the clothing which make the segmentation work difficult. In this paper, designing an algorithm that can automatically separate the clothing of interest from backgrounds is our aim.

* Corresponding author. Email: ling@asia.edu.tw

Segmentation is a process of partitioning an image into several meaningful regions that are homogenous with respect to some characteristics, like colors, textures, etc. Various methods have been presented for texture and image segmentation or classification. For texture segmentation, statistical methods have been widely used to extract texture features from an image (Conners & Harlow, 1980; Paragios & Deriche, 2002; He & Chen, 2000; Reed & Hans du Buf, 1993). However, these methods can not characterize well the structural characteristics of textures. Structural methods (Reed & Hans du Buf, 1993) are good for textures which are composed of well-defined texture elements. Since many textures violate this property, structural methods are of limited utility. Space/spatial-frequency based methods (Reed & Hans du Buf, 1993; Slepian, 1978; Wilson, 1987; Wilson & Spann, 1988; Bovik, Clark & Geisler, 1990; Teuner, Pichler & Hosticka, 1995; Dunn & Higgins, 1995; Unser, 1995) have been found to be of great use in texture segmentation. For example, orientation and frequency selection methods, such as the Gabor and Wavelet transform (Wilson, 1987; Wilson & Spann, 1988; Bovik et al., 1990; Teuner et al., 1995; Dunn & Higgins, 1995; Unser, 1995; Huang, Dai & Lin, 2006), have been widely used and good segmentation results have been shown. Their main drawbacks are the complicated computation and the need for prior determination of parameters. The finite prolate spheroidal sequences (FPSS) (Slepian, 1978; Wilson, 1987; Wilson & Spann, 1988), used in this paper, was presented early in 1978 by Slepian (Slepian, 1978), but not used on image processing until 1987 by Wilson (Wilson, 1987). Using the FPSS, we can specify intervals of both the spatial and frequency domains simultaneously, and thus can characterize textures easily in both the spatial and frequency domains. That is, the local information (relationship among pixels within a texture element) and global information (relationship among texture elements) of textures can be characterized.

In this paper, a color texture segmentation method is proposed to separate or segment the clothing of interest from backgrounds. The clothing to be segmented is specified by the designer (or the user) using a seed point. The clothing where the seed point locates is the one of interest. There are mainly three stages in our segmentation method. In the first stage we quantize the input color image to reduce the number of colors on the image. As a result, both the computational cost in segmentation and the effect of shadows and highlights on the image can be reduced. Then we use the FPSS, which can characterize the textures in both the spatial and frequency domains, to extract the features of the clothing. In the second stage, a coarse-to-fine segmentation method is applied based on the extracted features. A clustering method (the local centroid clustering method (Wilson & Spann, 1988)) is performed to coarsely segment the textures, and a hierarchical refinement process is utilized to refine the texture boundaries. After segmentation, each region contains textures which are homogeneous with respect to the extracted features. A simple region growing algorithm is then performed from the given seed point to locate the clothing boundary. Finally, post-processing is applied in the third stage to smooth the clothing boundary. The experimental results indicate that our proposed approach is indeed effective, and can somewhat tolerate shadows, highlights, folds, and texture orientations on the clothing.

In the remainder of this paper, detailed descriptions of the proposed approach are given in Section 2, including an introduction to the FPSS, the feature extraction method, the color texture segmentation method and post-processing for boundary smoothing. Experimental results are presented in Section 3. Conclusions appear in the last section.

2. PROPOSED APPROACH

There are three main stages in the proposed approach to separating the clothing of interest from backgrounds. These are feature extraction, color texture segmentation and post-processing. In Section 2.1 we introduce the features used for texture segmentation. A description of the feature extraction method is also included in this section. The hierarchical coarse-to-fine color texture segmentation approach is described in Section 2.2. Finally, post-processing for smoothing the clothing boundary is given in Section 2.3.

2.1 Feature Extraction Based on FPSS

A. FPSS

Before introducing the FPSS, we first define two operators: the truncation operator and bandlimiting operator. The truncation operator, $\mathbf{T}_{n1,\,n2}$, is defined as

$$\mathbf{T}_{n1,\,n2}\mathbf{M} = \begin{cases} \langle \mathbf{M} \rangle_{k\ell} & \text{if } n1 \le k \le n2 \\ 0 & \text{otherwise} \end{cases}, \quad 0 \le n1 \le n2 < n, 0 \le \ell < q \tag{1}$$

where $\mathbf{M}$ denotes an $n \times q$ matrix and $\langle \mathbf{M} \rangle_{k\ell}$ represents the entry in row k and column ℓ of matrix $\mathbf{M}$. $\mathbf{T}_{n1,\,n2}$ can also be considered as an $n \times n$ matrix:

$$\langle \mathbf{T}_{n1,\,n2} \rangle_{k\ell} = \begin{cases} 1 & \text{if } k = \ell \text{ and } n1 \le k \le n2 \\ 0 & \text{otherwise} \end{cases}, \quad 0 \le n1 \le n2 < n, 0 \le \ell < n\,. \tag{2}$$

The output matrix $\mathbf{T}_{n1,n2}\mathbf{M}$ obtained from applying the truncation operator $\mathbf{T}_{n1,\,n2}$ to the matrix $\mathbf{M}$ is called a truncated matrix. A vector $\mathbf{v}$ is called index limited if it satisfies

$$\mathbf{T}_{n1,\,n2}\mathbf{v} = \mathbf{v}\,. \tag{3}$$

The bandlimiting operator $\mathbf{B}_{m1,\,m2}$ is defined as

$$\mathbf{B}_{m1,\,m2} = \mathbf{F}^{*}\mathbf{T}_{m1,\,m2}\mathbf{F} \tag{4}$$

where **F** is the discrete Fourier transform (DFT) matrix (Gonzalez & Woods, 2007), which is defined as

$$\langle \mathbf{F} \rangle_{k\ell} = \frac{1}{\sqrt{n}} \exp(\frac{-j2\pi k\ell}{n}), \quad 0 \le k, \ell < n \tag{5}$$

and $\mathbf{F}^*$, the conjugate of **F**, is the inverse DFT (IDFT) matrix. $\mathbf{B}_{m1,\,m2}$ can also be regarded as an $n \times n$ matrix. A vector **u** is called bandlimited if the following equation is satisfied:

$$\mathbf{B}_{m1,\,m2}\mathbf{u} = \mathbf{u} \,. \tag{6}$$

As the name, truncation operator, indicates, when a truncation operator $\mathbf{T}_{n1,\,n2}$ is applied to a matrix the values of the elements within the specified range (from $n1$ to $n2$) in the matrix are unchanged, but the values of elements outside the range are set to zero. The region formed by the unchanged elements in this matrix is called the truncated region. Since the truncation operation is performed directly on a matrix, it is a spatial-domain operation. The bandlimiting operator is very much like the truncation operator. When applying the bandlimiting operation to a matrix (see Equation (4)), we first transform the matrix from the spatial domain to the frequency domain using the DFT matrix **F**, then perform the truncation operation on the transformed matrix, and finally, transform the truncated matrix from the frequency domain to the spatial domain using the IDFT matrix $\mathbf{F}^*$. Note that the bandlimiting operation is a frequency-domain operation whereas the truncation operation is a spatial-domain one.

We frequently need to specify the interval of time and the interval of frequency in analyzing a signal. We can specify the interval of time by a truncation operator with two parameters $n1$ and $n2$, and specify the interval of frequency by a bandlimiting operator with two parameters $m1$ and $m2$. However, can we use a single operator which can specify both intervals at the same time? As Equations (3) and (6) show, an index limited vector is an eigenvector of $\mathbf{T}_{n1,\,n2}$ and a bandlimited vector is an eigenvector of $\mathbf{B}_{m1,\,m2}$. It is clear that $\mathbf{T}_{n1,\,n2}$ and $\mathbf{B}_{m1,\,m2}$ are both Hermitian (Hogben, 2007), and that in general $\mathbf{T}_{n1,\,n2}\mathbf{B}_{m1,\,m2} \neq \mathbf{B}_{m1,\,m2}\mathbf{T}_{n1,\,n2}$. Furthermore, it can be shown that no vector exists which is an eigenvector of both $\mathbf{T}_{n1,\,n2}$ and $\mathbf{B}_{m1,\,m2}$ (Hogben, 2007). Thus, the answer of the above question is 'no'. That is, we can not find such an operator to specify both intervals at the same time. The question is now changed for an alternative solution, "Can we find an index limited vector which can approximate the bandlimited vector with the smallest loss of energy, and vice versa?" The answer is 'yes' (Wilson, 1987), and it is the index limited eigenvector $\mathbf{e}_0$ corresponding to the largest eigenvalue θ_0 of the operator $\mathbf{T}_{n1,n2}\mathbf{B}_{m1,m2}$:

$$\mathbf{T}_{n1,\,n2}\mathbf{B}_{m1,\,m2}\mathbf{e}_k = \theta_k \mathbf{e}_k \;, \quad 0 \le k < n \;\text{ and } \theta_0 \ge \theta_1 \ge \cdots \ge \theta_{n-1} \tag{7}$$

or the bandlimited eigenvector $\mathbf{g}_0$ corresponding to the largest eigenvalue θ_0 of the operator $\mathbf{B}_{m1,\,m2}\mathbf{T}_{n1,\,n2}$:

$$\mathbf{B}_{m1,m2}\mathbf{T}_{n1,n2}\mathbf{g}_k = \theta_k \mathbf{g}_k \ , \ \ 0 \le k < n \ \text{ and } \theta_0 \ge \theta_1 \ge \cdots \ge \theta_{n-1} . \tag{8}$$

The eigenvectors $\mathbf{e}_k$, $0 \le k < n$, form the FPSS. The relation between $\mathbf{e}_k$ and $\mathbf{g}_k$ (Wilson, 1987) is

$$\mathbf{g}_k = \theta_k^{-1/2}\mathbf{B}_{m1,\,m2}\mathbf{e}_k \ , \ 0 \le k < n . \tag{9}$$

The eigenvector $\mathbf{e}_k$ is index limited and can approximate $\mathbf{g}_k$ with the minimum loss; likewise, the bandlimited vector $\mathbf{g}_k$ can approximate the $\mathbf{e}_k$. Figure 1 shows the spatial response and the frequency response of $\mathbf{e}_0$ and $\mathbf{g}_0$. Using the FPSS, we can specify both intervals in the spatial and frequency domains. The operator $\mathbf{T}_{n1n2}$ is mainly concerned with an interval in the spatial domain, but the operator $\mathbf{B}_{m1m2}$ an interval in the frequency domain.

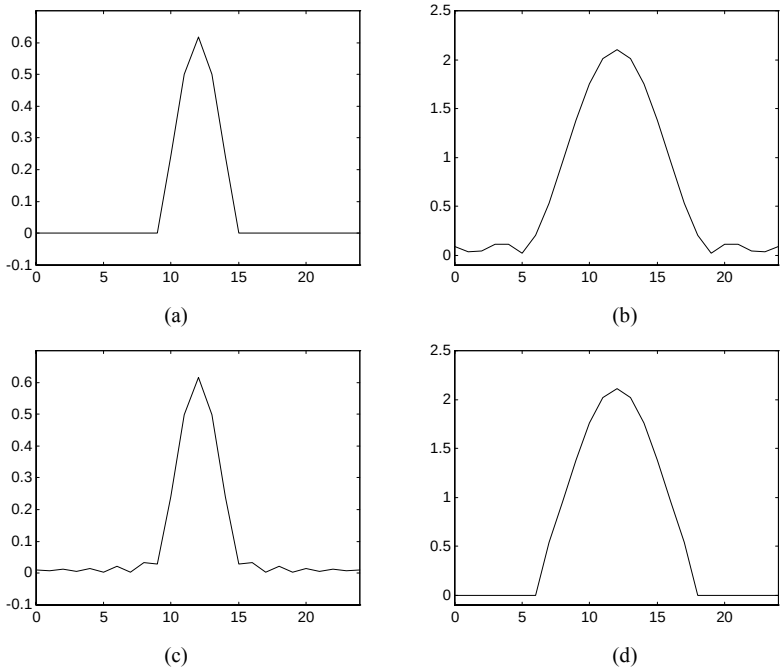


Figure 1. Spatial and frequency responses of $\mathbf{e}_0$ and $\mathbf{g}_0$ with (n=25, $n1$=10, $n2$=14, $m1$=7, $m2$=17): (a) spatial response of $\mathbf{e}_0$; (b) frequency response of $\mathbf{e}_0$; (c) spatial response of $\mathbf{g}_0$; (d) frequency response of $\mathbf{g}_0$.

B. 2-D FPSS

The FPSS introduced above is of one-dimensional (1-D) form. To process two-dimensional (2-D) images, we extend it to the 2-D form. The FPSS of a 2-D

form is just the combination of two 1-D FPSS's respectively in the horizontal (x) and vertical (y) directions of the Cartesian coordinate system. The truncation and bandlimiting operators of 2-D form are therefore the Kronecker products (Hogben, 2007) of those of 1-D form. Thus, the 2-D form of Equation (7) is

$$\mathbf{TBe}_k = \mathbf{T}_x \otimes \mathbf{T}_y \mathbf{B}_x \otimes \mathbf{B}_y \mathbf{e}_{xk} \otimes \mathbf{e}_{yk} = \theta_{xk}\theta_{yk}\mathbf{e}_{xk} \otimes \mathbf{e}_{yk}, \quad 0 \le k < n \tag{10}$$

where $\mathbf{T}$ and $\mathbf{B}$ are both $n \times n$ matrices, denoting respectively the 2-D truncation operator and the bandlimiting operator. $\mathbf{T}_x$ and $\mathbf{B}_x$ are respectively the 1-D truncation operator and bandlimiting operator in the x direction, and $\mathbf{T}_y$ and $\mathbf{B}_y$ are respectively the 1-D truncation operator and bandlimiting operator in the y direction; $\mathbf{e}_k$ are the eigenvectors (2-D FPSS) of $\mathbf{TB}$; $\mathbf{e}_{xk}$ and θ_{xk} are respectively the eigenvectors (1-D FPSS) and eigenvalues of $\mathbf{T}_x\mathbf{B}_x$ with parameters ($xn1$, $xn2$, $xm1$, $xm2$), and $\mathbf{e}_{yk}$ and θ_{yk} are respectively the eigenvectors (1-D FPSS) and eigenvalues of $\mathbf{T}_y\mathbf{B}_y$ with parameters ($yn1$, $yn2$, $ym1$, $ym2$).

Given the parameters ($xn1$, $xn2$, $xm1$, $xm2$) and ($yn1$, $yn2$, $ym1$, $ym2$), we can use two 1-D FPSS's $\mathbf{e}_{xk}$ and $\mathbf{e}_{yk}$ to obtain the 2-D FPSS $\mathbf{e}_k$. With these parameters, the truncated region in the spatial domain and that in the frequency domain are both rectangular. The regions truncated to rectangular shapes are of Cartesian separable form (Wilson & Spann, 1988). Note that the 2-D truncation operator $\mathbf{T}$ with parameters ($xn1$, $xn2$, $yn1$, $yn2$) can be treated as a mask matrix, where the values are all 1's within the truncated region but 0's outside the region. Hence, the truncation operation $\mathbf{TM}$ becomes the entry-by-entry product of $\mathbf{T}$ and $\mathbf{M}$ (Hogben, 2007):

$$\mathbf{TM} = \begin{bmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots & \vdots \\ 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \cdots & 0 \\ 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \cdots & 0 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 1 & 1 & \cdots & 1 & \cdots & 0 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} .\times \begin{bmatrix} m_{0,0} & \cdots & m_{0,xn1} & m_{0,..} & m_{0,..} & \cdots & m_{0,xn2} & \cdots & m_{0,q} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots & \vdots \\ m_{yn1,0} & \cdots & m_{yn1,xn1} & m_{yn1,..} & m_{yn1,..} & \cdots & m_{yn1,xn2} & \cdots & m_{yn1,q} \\ m_{..,0} & \cdots & m_{..,xn1} & m_{..,..} & m_{..,..} & \cdots & m_{..,xn2} & \cdots & m_{..,q} \\ m_{..,0} & \cdots & m_{..,xn1} & m_{..,..} & m_{..,..} & \cdots & m_{..,xn2} & \cdots & m_{..,q} \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{yn2,0} & \cdots & m_{yn2,xn1} & m_{yn2,..} & m_{yn2,..} & \cdots & m_{yn2,xn2} & \cdots & m_{yn2,q} \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{p,0} & \cdots & m_{p,xn1} & m_{p,..} & m_{p,..} & \cdots & m_{p,xn2} & \cdots & m_{p,q} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots & \vdots \\ 0 & \cdots & m_{yn1,xn1} & m_{yn1,..} & m_{yn1,..} & \cdots & m_{yn1,xn2} & \cdots & 0 \\ 0 & \cdots & m_{..,xn1} & m_{..,..} & m_{..,..} & \cdots & m_{..,xn2} & \cdots & 0 \\ 0 & \cdots & m_{..,xn1} & m_{..,..} & m_{..,..} & \cdots & m_{..,xn2} & \cdots & 0 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & m_{yn2,xn1} & m_{yn2,..} & m_{yn2,..} & \cdots & m_{yn2,xn2} & \cdots & 0 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \tag{11}$$

where ".×" is the entry-by-entry product operator. By Equations (10) and (11), we can derive the general form of the 2-D FPSS (Wilson, 1987) as follows:

$$\mathbf{TBe}_k = \mathbf{T}_s\mathbf{F}^* \otimes \mathbf{F}^*\mathbf{T}_f\mathbf{F} \otimes \mathbf{Fe}_k = \theta_k \mathbf{e}_k \tag{12}$$

where $\mathbf{T}_s$ and $\mathbf{T}_f$ denote truncation operators in the spatial domain and the frequency domain, respectively. In Equation (10), we use a 1-D FPSS in the x direction and a 1-D FPSS in the y direction to derive the 2-D FPSS. While in Equation (12), we obtain the 2-D FPSS through the specification of a truncated region in the spatial domain ($\mathbf{T}_s$) and that in the frequency domain ($\mathbf{T}_f$). The truncated regions of the spatial domain and the frequency domain can be of arbitrary shape. A rectangular-shape truncated region is of Cartesian separable form (Wilson & Spann, 1988). A circular-shape or wedge-shape truncated region is of polar separable form (Wilson & Spann, 1988). Different shapes of truncated region can be obtained by using different mask matrices.

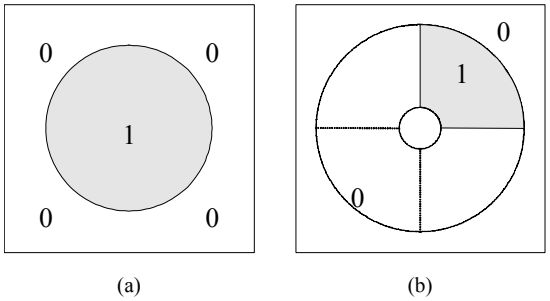


Figure 2. Different shapes can be used in the mask matrix: (a) circular; (b) wedge.

C. Color Texture Features

Selection of texture features is important for color image segmentation. Without good features, no matter how good the segmentation scheme is, the result will not be satisfactory. Hence, many approaches have given emphasis to selecting or extracting promising features for texture segmentation.

In this paper, the FPSS is used to extract the texture features of the clothing. Using the FPSS, we can characterize textures in both the spatial and frequency domains. Hence, the local and global information of textures can be obtained. That is, we can use the FPSS to characterize the relationship among pixels within a texture element (local information) and that among texture elements (global information). Before extracting texture features, we first transform the input image from the RGB (red, green, blue) space to the *HSI* (hue, saturation, intensity) space. Next, we apply the circular hue histogram approach (Li & Tseng, 1995) to quantize the hue image such that both the number of colors and the influence of shadows/highlights on the image can be reduced. Figure 3 shows four color model images, and Figure 4 shows their quantized results.

Figure 3. Color images of four models.

Figure 4. Quantized results of images in Figure 3: (a) quantized to 7 colors; (b) quantized to 7 colors; (c) quantized to 8 colors; (d) quantized to 6 colors.

The texture features will be obtained by convoluting the FPSS with the quantized image. Let the truncated region of a FPSS in the spatial domain be S, and that in the frequency domain be Ω. Assume that the areas of S and Ω are $A(S)$ and $A(\Omega)$, respectively and they satisfy

$$A(S)A(\Omega) = MN \tag{13}$$

where M and N are the width and height of the image, respectively. The first eigenvector $\mathbf{e}_0$ of the FPSS is an effective basis for the whole spatial and frequency domains (Wilson, 1987; Wilson & Spann, 1988). Consequently, if we choose the parameters of the FPSS such that Equation (13) holds, the texture features generated from the first eigenvector of each FPSS will be promising for segmentation.

We can truncate the frequency domain to a set of disjoint regions, which constitute a tessellation. Different FPSS's can be generated when we use different tessellations. In the spatial domain, the truncated region is located in the center, and its size is the same as that of the convolution window. In the frequency domain, there are two commonly used tessellations, the Cartesian separable tessellation and the polar separable tessellation (Wilson & Spann, 1988), as shown in Figure 5. The Cartesian separable tessellation is computationally easy, and it has been shown that using the Cartesian separable tessellation is as good as using the polar separable tessellation for texture segmentation (Wilson & Spann, 1988).

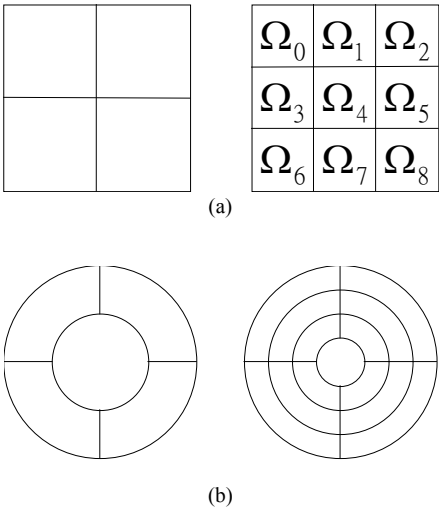


Figure 5. Tessellations used in the frequency domain: (a) Cartesian separable tessellation; (b) polar separable tessellation.

In the paper, the 9-region Cartesian separable tessellation is used. Let the 9 regions of the tessellation be denoted as Ω_0, Ω_1, …, Ω_8 (the width and length of the 9 regions are all n). Given a quantized image $\mathbf{I}$ and the set of $n^2\times1$ eigenvectors $\mathbf{e}_0(\Omega_i)$ of the FPSS's, $0 \le i \le 8$, we define the texture feature vector $\mathbf{F}(x, y)$ for each point (x, y) in the image as follows:

$$\mathbf{F}(x, y) = (f_0(x, y), f_1(x, y), \cdots, f_8(x, y))$$

with

$$f_i(x, y) = \left|\mathbf{I}(x, y) * \mathbf{fpss}_i\right|, \qquad 0 \le i \le 8$$

and

$$\left\langle \mathbf{fpss}_i \right\rangle_{k\ell} = \left\langle \mathbf{e}_0(\Omega_i) \right\rangle_{k\times n+\ell} \tag{14}$$

where * is the convolution operator. These feature vectors will be used in the segmentation in the next section.

2.2 Color Texture Segmentation

The texture feature vectors $\mathbf{F}(x, y)$ are used as features here for texture segmentation. The segmentation method used in this paper is a multi-dimensional hierarchical coarse-to-fine approach. In the method, we first build a hierarchical structure based on the feature vectors in a bottom-up manner. Next, we segment the top level (the coarsest resolution level) of the structure by a local centroid clustering algorithm (Wilson & Spann, 1988). Finally, using the coarse segmentation results, we proceed down the hierarchical structure to refine the boundaries of texture regions level by level until the bottom level (the finest resolution level) is reached. This coarse-to-fine segmentation process can reduce the noise on the image. Furthermore, finding texture boundaries using the level-by-level segmentation method can achieve a lower computational cost.

A. Building a hierarchical structure

The hierarchical structure that we build for segmentation is the so-called quadtree (Wilson & Spann, 1988). Each feature vector $\mathbf{F}(x, y)$ extracted from the FPSS is considered as a point in the bottom level of the quadtree. Each level, except the bottom level, of the quadtree is constructed from its lower level. The value of each point in the ℓ-th level is computed from the values of four points in the (ℓ-1)-th level in an averaging manner. Assume that the bottom level $2^n\times2^n$ points. Let the value of a point located at (x, y) in the ℓ-th level of the quadtree be denoted as $\mathbf{q}(x, y, \ell)$. Then

$$\mathbf{q}(x, y, \ell) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1}\mathbf{q}(2x+u, 2y+v, \ell-1), \quad 0 \le x, y < 2^{n-\ell}. \tag{15}$$

Note that $\mathbf{q}(x, y, 0) = \mathbf{F}(x, y)$. The size of each level (i.e., the number of points in each level) is one-fourth of that of its lower level. When a level of 16×16 points is obtained, we terminate the quadtree construction process. That is, the top level is 16×16 points. If the top level is of large size, then a lot of the cost of computing will be spent in the segmentation of the top level. However, if it is of small size, then the texture boundaries on the top level will be too blurred to segment. The size of the top level is obtained by experience. The noise is reduced in the higher level after the averaging operation is performed, but the boundaries of texture regions are blurred at the same time. Therefore, segmenting the top level of the quadtree, we obtain a coarse segmentation result. Proceeding down the quadtree, we can refine the segmentation results to locate the correct texture boundaries.

B. Segmenting the top level of the structure

The segmentation work starts from the top level of the quadtree. The segmentation result of the top level will have a great influence on the entire segmentation process. A misclassified point $\mathbf{q}(x, y)$ in the top level will result in the misclassification of four points (which are used to generate $\mathbf{q}(x, y)$ in quadtree construction) in its lower level. Furthermore, when segmenting downwards, the number of misclassified points grows four times when its lower level is processed. For example, if a point in the top level which is 16×16 is misclassified, then 256 points will be misclassified in the bottom level which is 256×256. Consequently, segmentation of the top level is of importance. At the top level two steps need to be performed:

(1) Local centroid clustering: segmenting the texture image.
(2) Insignificant regions removal: removing regions whose number of points is small.

The local centroid clustering is an iterative process (Wilson & Spann, 1988). The feature set used here is the set of features $\mathbf{q}(x, y, L)$ (L denotes the top level of the quadtree). For each point $\mathbf{q}$ in the feature set, we compute the local center LC_{q} from its neighboring points $\mathbf{p}$ in a widow W_R of a specified radius R as follows:

$$LC_{\mathbf{q}} = \frac{\sum_{\mathbf{p}\in W_R} \mathbf{p}}{\sum_{\mathbf{p}\in W_R} 1}. \tag{16}$$

Each point will be moved to a new local center or stay unchanged if its location is the same as that of the local center. Then a new iteration is performed again using the new locations of all points. The process terminates when all points stay unmoved. The points which occupy the same location form a class. In the paper, weights are added to Equation (16) for computing the local center as follows:

$$LC_{\mathbf{q}} = \frac{\sum_{\text{all}\,\mathbf{p}} w_{\mathbf{qp}}\mathbf{p}}{\sum_{\text{all}\,\mathbf{p}} w_{\mathbf{qp}}}$$

with

$$w_{\mathbf{qp}} = \frac{1}{\exp\left(d_{\mathbf{qp}} \big/ 10\right)} \tag{17}$$

where d_{qp} is the Euclidean distance from **q** to **p**. By Equation (17), we need not to specify the window of **q** but use all the points in the feature space to compute its local center. Its advantage is that no a priori information on the number of classes or the class centers is required.

The purpose of this paper is to separate the clothing of interest from backgrounds. We assume that the clothing to be segmented occupies a significant percentage in the image. There may be buttons, pockets or folds on the clothing whose textures are different from the clothing texture. They are usually grouped into classes with small areas in the top level. Consequently, we remove these insignificant classes or regions and reassign them such that the whole clothing region can be obtained.

The insignificant regions are removed using the following steps (Schroeter & Bigün, 1995). We first find all regions whose number of points is small. Second, for each point **q**(x, y, L) in these regions, determine its neighboring classes C_k ($0 \le k \le 7$) in its eight directions (see Figure 6). If the class of **q**($x+u$, $y+v$, L), $-1 \le u, v \le 1$, is different from that of **q**(x, y, L), then the class of **q**($x+u$, $y+v$, L) is stored as a neighboring class of **q**(x, y, L). Otherwise, we consider the next point **q**($x+2u$, $y+2v$, L) in the same direction and compare the classes of **q**(x, y, L) and **q**($x+2u$, $y+2v$, L). If they are different, the class of **q**($x+2u$, $y+2v$, L) is stored as a neighboring class of **q**(x, y, L). If they are the same, we consider the next point **q**($x+3u$, $y+3v$, L) in the same direction. The process is repeated until a neighboring class of **q**(x, y, L) is found. After we obtain eight neighboring classes for the point **q**(x, y, L), we reassign **q**(x, y, L) to one of the neighboring classes whose center has the minimum Euclidean distance (Gonzalez & Woods, 2007) to **q**(x, y, L). For example, in Figure 6, the eight neighboring classes of **q** are (C_0, C_1, …, C_7) = (1, 3, 3, 3, 3, 2, 2, 2). Then we reassign **q** to class 1, assuming the distance from the center of class 1 to **q** is smaller than those from class 2 and class 3.

After removing insignificant regions, segmentation of the top level is completed.

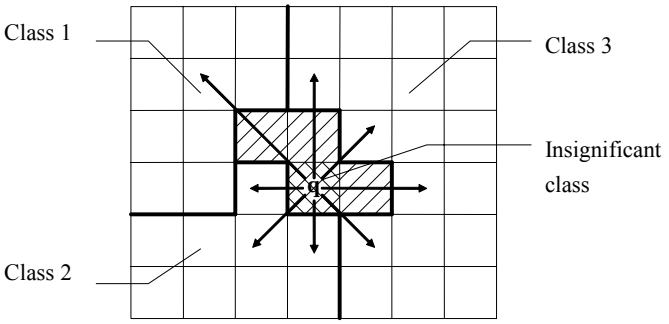


Figure 6. Eight directions of the point q.

C. Segmenting downwards from the top level of the structure

The class boundaries obtained in the top level are blurred or rough. Proceeding downwards the quadtree level by level from the top, we can refine the boundaries. The following steps are used to compute the texture boundaries at the ℓ-th level of the quadtree:

(1) Initialize the set of boundary points, $B(\ell)$, in the ℓ-th level to be an empty set.
(2) Apply the following rule to each point $\mathbf{q}(x, y, \ell)$ in the ℓ-th level for finer classification:

if the classes of $\mathbf{q}(\frac{x}{2}+u, \frac{y}{2}+v, \ell+1)$ are the same for all integers u and v, $-1 \le u, v \le 1$,
then assign $\mathbf{q}(x, y, \ell)$ to the class of $\mathbf{q}(x/2, y/2, \ell+1)$;
else set $B(\ell) = B(\ell) \cup (x, y)$.

(3) For each point $\mathbf{q} \in B(\ell)$, assign it to one of the classes of its corresponding 9 points in the $(\ell+1)$-th level (see Figure 7) whose center has the minimum distance to $\mathbf{q}$.

The above steps are repeated for each level until the bottom level is reached.

In quadtree construction, a point $\mathbf{q}(\frac{x}{2}, \frac{y}{2})$ in the $(\ell+1)$-th level is obtained from four points ($\mathbf{q}(x, y)$, $\mathbf{q}(x, y+1)$, $\mathbf{q}(x+1, y)$, $\mathbf{q}(x+1, y+1)$) in the ℓ-th level, as shown in Figure 7. Therefore, the classes of these four points in the ℓ-th level are determined by the classes of $\mathbf{q}(\frac{x}{2}, \frac{y}{2})$ and its 8 neighboring points in the $(\ell+1)$-th level. If the 9 points in the $(\ell+1)$-th level are of the same class, then all the four points in the ℓ-th level are assigned to this class. Otherwise, each of these four points is reassigned to one of the classes of the 9 points in the $(\ell+1)$-th level whose

center has the minimum distance to it. Thus, the boundary location in the ℓ-th level is more accurate than that in the (ℓ+1)-th level. By performing the refinement process level by level from top to bottom, the boundaries can be located with greater and greater precision. In the final result, we obtain the class of each point on the image. We then apply a simple region growing algorithm (Gonzalez & Woods, 2007) starting from a given seed point to locate the whole clothing boundary. Figure 8 shows the segmentation results of the model images in Figure 4.

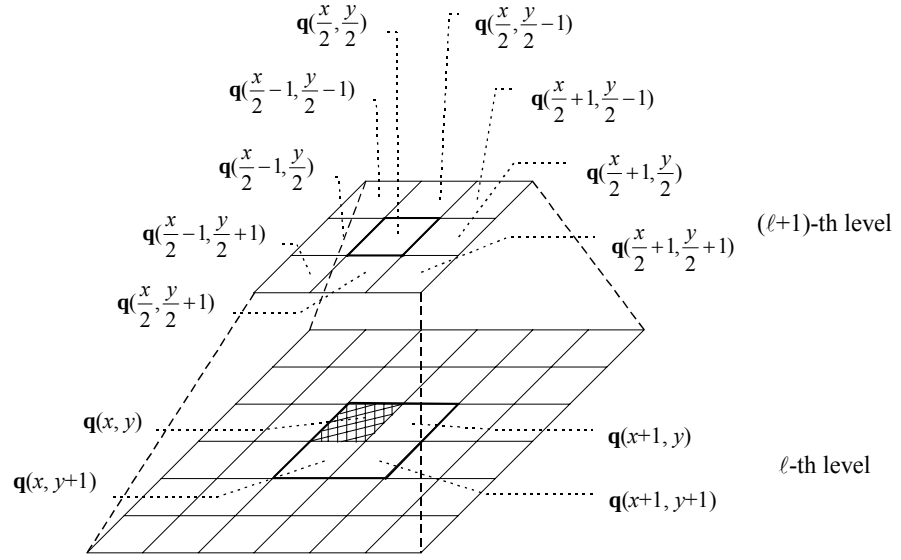


Figure 7. The corresponding 9 points in the (ℓ+1)-th level for the point q(x, y) in the ℓ-th level.

2.3 Post-Processing

The extracted clothing boundary obtained in the previous stage is frequently jagged. Hence, post-processing is necessary to obtain a smooth clothing boundary. There are two steps in post-processing, including morphological filtering (Gonzalez & Woods, 2007) and Gaussian smoothing (Lin, Wang & Yang, 1996).

The morphological filtering process consists of the opening and closing operations. The opening operation can eliminate the protrusions, while the closing operation can fill the gaps. We use the filter on the clothing region to remove small protrusions and gaps on the clothing boundary, and thus smooth the boundary. A circular structuring element with radius 3 is used in the morphological filter in our experiments. The filtering results of the images in Figure 8 are shown in Figure 9.

(a) (b)

(c) (d)

Figure 8. Segmentation results of images in Figure 4 without post-processing.

(a) (b)

(c) (d)

Figure 9. Morphological filtering results of images in Figure 8.

After morphological filtering, the clothing boundary is still a little jagged though the protrusions and gaps are removed. We use the Gaussian smoothing algorithm on the clothing boundary to smooth the boundary. The Gaussian smoothing algorithm uses the Gaussian function to convolute an input 1-D signal for smoothing the signal. In the algorithm, the spread parameter (or the standard deviation) of the Gaussian function is automatically determined by an iterative process (Lin et al., 1996). To apply the algorithm, we represent the boundary as a sequence of points with their (x, y) coordinates: (x_0, y_0), (x_1, y_1), …, (x_{n-1}, y_{n-1}), where n is the number of boundary points. The x and y coordinates of the points form two 1-D signals $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$, respectively. Thus we apply the automatic Gaussian smoothing algorithm on these two 1-D signals to smooth the signals. Finally, the clothing boundary can be polygonally approximated based on the two smoothed 1-D signals (Lin et al., 1996). The smoothing results of the images in Figure 9 are shown in Figure 10.

Figure 10. Gaussian smoothing results of images in Figure 9.

3. EXPERIMENTAL RESULTS

The proposed approach has been implemented in C language on a Pentium-100 PC with an UMAX digital color scanner and the MATLAB mathematical tool package. Most model images are obtained from magazines, such as LADY BOUTIQUE, JUNIE, and NON-NO.

In the experiments, the truncated region in the spatial domain is 11×11 points. Each of the 9 truncated regions in the frequency domain is 23×23 points and the whole frequency domain is 69×69 points. Some experimental results are shown in Figures 11 through 14. Figure 11 shows the segmentation results of four samples of non-textured clothing. The shadows and highlights on the clothing are evident. Figure 12 shows the segmentation results of four samples of textured clothing. In these images, shadows, highlights and folds are apparent on the clothing. Figure 13 shows the segmentation results of four samples of textured clothing with larger pattern prints. Figure 14 shows the segmentation results of four samples of textured clothing with serious folds. It takes about two minutes in total for each 256×256 image to separate the clothing of interest from the background.

To evaluate the experimental results quantitatively, an error function is defined. Let P be the clothing region whose boundary is found by the proposed method, and H be that whose boundary is specified by hand. The error function $e(P, H)$ is defined as

$$e(P,H) = \frac{A(P \cup H) - A(P \cap H)}{A(H)}$$

where $A(\cdot)$ denotes the area. The value of the error function for each segmentation result is given in Figures 11 through 14. The average value of the error function in Figures 11 through 14 is about 2.96%.

(a) $e(P, H)$ = 1.42% (b) $e(P, H)$ = 3.45%

(c) $e(P, H)$ = 3.42% (d) $e(P, H)$ = 0.63%

Figure 11. Segmentation results of non-textured clothing.

(a) $e(P, H) = 1.33\%$ (b) $e(P, H) = 2.39\%$

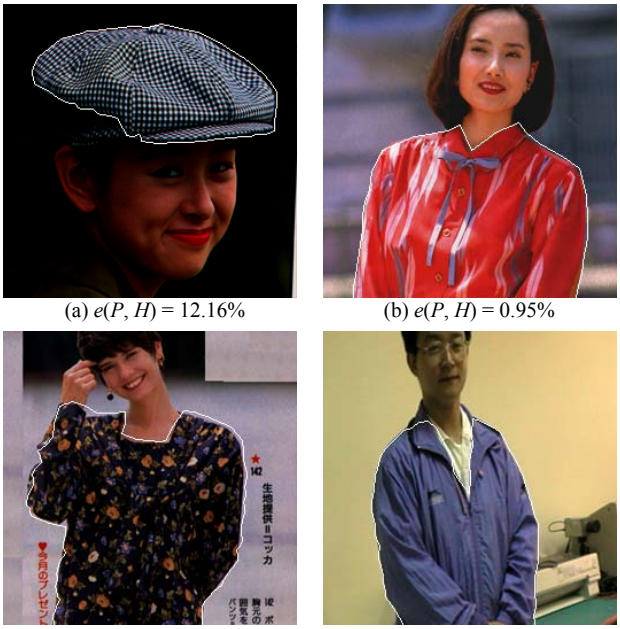
(c) $e(P, H) = 3.67\%$ (d) $e(P, H) = 4.14\%$

Figure 12. Segmentation results of textured clothing.

(a) $e(P, H) = 3.28\%$ (b) $e(P, H) = 1.50\%$

(c) $e(P, H) = 1.77\%$ (d) $e(P, H) = 2.96\%$

Figure 13. Segmentation results of clothing with larger pattern prints.



(a) $e(P, H) = 12.16\%$ (b) $e(P, H) = 0.95\%$

(c) $e(P, H) = 2.70\%$ (d) $e(P, H) = 1.57\%$

Figure 14. Segmentation results of clothing with serious folds.

From these figures, some defects can be found in the segmentation results which still remain to be solved. First, when the clothing to be segmented and the background have similar hue attributes, some points on the background or on the clothing will be misclassified. Second, when there are serious shadows or highlights on the clothing, the points on it may also be misclassified. Finally, if there is a hole which is surrounded by the clothing region, it may be considered as an insignificant region and be reassigned to the class of the clothing. Figure 12(d) shows an example.

4. CONCLUSIONS

In this paper, we have proposed a color texture segmentation method for segmenting clothing of interest from a background. The texture features extracted based on the FPSS is promising for segmentation. Texture characteristics in both the spatial and frequency domains can be captured adequately by the FPSS. The hierarchical coarse-to-fine segmentation process shows its ability with noise reduction and boundary information preserving. The proposed approach is somewhat tolerant of shadows, highlights, folds and orientation variations on the textures and satisfactory experimental results are achieved.

Further research may be directed to the following topics. The first is to use more color information rather than only the hue attribute of an image to improve the quantization effect. The second is to find efficient algorithms for convolution operations and for solving the eigensystem problem (computing the eigenvectors and eigenvalues of a matrix) to improve the computing speed. And the third is to achieve better performance by resolving the serious shadow/highlight problem.

REFERENCES

Bovik, A. C., Clark, M., & Geisler, W. S. (1990). Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(1), 55-73.

Conners, R. W., & Harlow, C. A. (1980). A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*(3), 204-222.

Dunn, D., & Higgins, W. E. (1995). Optimal Gabor filters for texture segmentation. *IEEE Transactions on Image Processing*, *4*(7), 947-964.

Gonzalez, R. C., & Woods, R. E. (2007). *Digital Image Processing*. New Jersey, USA: Prentice Hall.

He, H., & Chen, Y. Q. (2000). Unsupervised texture segmentation using resonance algorithm for natural scenes. *Pattern Recognition Letters*, *21*, 741-757.

Huang, P. W., Dai, S. K., & Lin, P. L. (2006). Texture image retrieval and image segmentation using composite sub-band gradient vectors. *Journal of Visual Communication and Image Representation*, *17*, 947-957.

Hogben, L. (2007). *Handbook of Linear Algebra*. Florida, USA: Chapman.

Li, Y. F., & Tseng, D. C. (1995). Circular histogram thresholding for color image segmentation. *proceedings of the International Conference on Document Analysis and Recognition*, Montreal, Canada.

Lin, H. C., Wang, L. L., & Yang, S. N. (1996). Automatic determination of the spread parameter in Gaussian smoothing. *Pattern Recognition Letters*, *17*(12), 1247-1252.

Paragios, N., & Deriche, R. (2002). Geodesic active regions and level set methods for supervised texture segmentation. *International Journal of Computer Vision*, *46*(3), 223-247.

Reed, T. R., & Hans du Buf, J. M. (1993). A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image Understanding*, *57*(3), 359-372.

Schroeter, P., & Bigün, J. (1995). Hierarchical image segmentation by multi-dimensional clustering and orientation-adaptive boundary refinement. *Pattern Recognition*, *28*(5), 695-709.

Slepian, D. (1978). Prolate spheroidal wave functions, Fourier analysis, and uncertainty - V: the discrete case. *The Bell System Technical Journal*, *57*(5), 1371-1430.

Teuner, A., Pichler, O., & Hosticka, B. J. (1995). Unsupervised texture segmentation of images using tuned matched Gabor filters. *IEEE Transactions on Image Processing*, *4*(6), 863-870.

Unser, M. (1995). Texture classification and segmentation using Wavelet frames. *IEEE Transactions on Image Processing*, *4*(11), 1549-1560.

Wilson, R. (1987). Finite prolate spheroidal sequences and their application I: generation and properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *9*(6), 787-795.

Wilson, R., & Spann, M. (1988). Finite prolate spheroidal sequences and their application II: image feature description and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *10*(2), 193-203.

Ling-Ling Wang received a B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Chiao Tung University, Hsinchu, Taiwan, ROC in 1984, 1986, and 1990, respectively.

From 1986 to 1987, she was an associate engineer in the System Software Department of ERSO, ITRI, Hsinchu, Taiwan. From 1991 to 1997, she was an associate professor of Computer Science at National Tsing Hua University, Taiwan. Currently, she is a professor of Information Communication at Asia University, Taichung, Taiwan. Her current research is in image processing, pattern recognition, computer vision, and artificial intelligence.