

Texture Segmentation Using Voronoi Polygons*

Mihran Tüceryan

Anil K. Jain

Department of Computer Science

Michigan State University

East Lansing, MI 48824-1027

June 26, 1989

Abstract

Texture segmentation is one of the early steps towards identifying surfaces and objects in an image. Textures considered here are defined in terms of primitives called tokens. In this paper we have developed a texture segmentation algorithm based on the Voronoi tessellation. The algorithm first builds the Voronoi tessellation of the tokens that make up the textured image. It then computes a feature vector for each Voronoi polygon. These feature vectors are used in a probabilistic relaxation labeling on the tokens, to identify the interior and the border regions of the textures. The algorithm has successfully segmented binary images containing textures whose primitives have identical second-order statistics and a number of gray level texture images.

1 INTRODUCTION

The natural world abounds with textured surfaces. Any realistic vision system that is expected to work successfully, therefore, must be able to handle such input. The process of identifying regions with similar texture and separating regions with different texture is one of the early steps towards identifying surfaces and objects. This process is called texture segmentation and is the major focus of this paper.

Texture segmentation has been studied for a long time using various approaches. The most popular approach performs texture analysis directly upon the gray levels in an image. This includes gray level co-occurrence matrix (GLCM) [10], autocorrelation function analysis [10], generalized co-occurrence matrices (GCM) [8], second order spatial averages [9], and two-dimensional filtering in the spatial and frequency domain [6,5,24,25,20]. Another approach operates at a symbolic level where a textured image is organized

*This research was supported in part by NSF Grant no. IRI 8705256 and Grant no. CDA 8806599.

or represented in terms of primitives. Examples of this can be seen in Julesz’s work [13,12] and in syntactic texture analysis. There has also been recent interest in texture analysis approaches based on statistical modeling such as Markov random fields (MRF) [7,17,16]. Some texture analysis methods, for example, Beck *et al.* have examined the role of spatial frequency channels (signal processing level) and perceptual grouping (symbolic level) in texture segregation [1,2].

Julesz had conjectured [14,15] that texture pairs with identical second-order statistics cannot be preattentively discriminated by humans, but he later gave counterexamples to this conjecture [13]. An important factor in the preattentive discrimination of such texture pairs appears to be certain shape features of the texture primitives. These features include closure, linearity, terminations, etc. which are called *textons* [13]. The important questions that need to be answered in this theory are whether the list of textons is complete and how to extract textons from images. One approach to this problem is to formulate a general framework for feature detection so that the detected features capture the texton information. If this is accomplished then we no longer have to list all the textons. This is the main motivation in this paper. Shape features of the Voronoi polygons are used to perform texture segmentation and they seem to capture the perceptually necessary information for this task. Furthermore, the features so extracted depend on local properties of the image.

The textures that are of interest to us are defined in terms of texture primitives which we call *tokens*. These tokens can be either dots or points (simplest case) or they can be more complex primitives such as arrows and triangles. Primitives can be extracted from the gray level images as a result of some low-level processing (Section 3.3) or they can be generated synthetically in order to control precisely the various spatial and statistical properties of the texture [13].

In this paper we propose a method of obtaining shape features from tokens at a symbolic level which will be useful for texture segmentation. This is accomplished by first computing the Voronoi tessellation of the input texture and then extracting the shape properties of the resulting Voronoi polygons. Shape features of the polygons are represented by their moments of area which are able to capture not only the first-order differences in textures (such as density differences) but also higher order statistical differences in textures (e.g., two textures with the same dot density but different orientation tendencies). The Voronoi tessellation has been used as the underlying geometric representation in other problems such as grouping of dot patterns [23]. The latter, however, did not involve the computation of textural properties.

Our texture segmentation scheme consists of the following three basic steps: (a) Define local relationships between texture primitives or tokens using a graph structure. We use the Delaunay graph (Voronoi diagram) for this purpose. (b) Identify the degree of “inconsistency” of the edges in the Delaunay graph. The edges in the interior regions of a homogeneous texture region will be “consistent” and those between two different texture regions will be “inconsistent.” The degree of inconsistency of the Delaunay edges are computed using the shape features of each Voronoi polygon (Section 3.1). (c) The initial labeling in (b) is refined using probabilistic relaxation labeling, which uses the Gestalt constraint of boundary smoothness. Algorithm 1

Input: Gray level texture image.

1. Extract tokens.
2. Construct the Voronoi tessellation.
3. Compute initial BREAK probabilities for Delaunay edges based on moments of area of polygons. (See Section 3.1).
4. Perform a probabilistic relaxation labeling to identify interior and border tokens. $P_i^{(k)}(\lambda)$ is the probability of label λ for the i^{th} token at the k^{th} iteration. (See Section 3.2).
5. The final label on token i is λ if $P_i^{(k)}(\lambda) = \max_{\lambda'} P_i^{(k)}(\lambda')$

Output: Output labeling identifies interior regions and borders of regions.

Algorithm 1: The overall algorithm for the texture segmentation.

shows the steps of the texture segmentation algorithm.

Section 2 describes the Voronoi tessellation and the computation of the textural features from it. Section 3 describes our segmentation algorithm and the probabilistic relaxation labeling scheme. Section 4 gives experimental results and Section 5 makes some concluding remarks.

2 VORONOI TESSELLATION

Our texture segmentation algorithm uses the Voronoi tessellation to establish local relationships among the tokens. The features of Voronoi cells are then used to group the tokens belonging to the same texture. In this section we give a definition of the Voronoi tessellation and the texture features we compute from it.

Let \mathbf{S} denote a set of three or more points in the Euclidean plane. Assume that these points are not all collinear, and that no four points are cocircular. Consider an arbitrary pair of points P and Q belonging to \mathbf{S} . The bisector of the line joining P and Q is the locus of points equidistant from both P and Q and divides the plane into two halves. The half plane H_P^Q (H_Q^P) is the locus of points closer to P (Q) than to Q (P). For any given point P , a set of such half planes is obtained for various choices of Q . The intersection

$\bigcap_{Q \in \mathbf{S}, Q \neq P} H_P^Q$ defines a polygonal region consisting of points in the Euclidean plane closer to P than to any other point. Such a region is called the *Voronoi polygon* [26] associated with the point P . The set of complete

polygons for all points in \mathbf{S} is called the *Voronoi diagram* of \mathbf{S} [22]. The Voronoi diagram together with the incomplete polygons in the convex hull of \mathbf{S} define a *Voronoi tessellation* of the entire plane. Two points $P, Q \in \mathbf{S}$ are said to be *Voronoi neighbors* if the Voronoi polygons of P and Q share a common edge. The

Feature	Computation
f_1	m_{00}
f_2	$\sqrt{\bar{x}^2 + \bar{y}^2}$
f_3	$\tan^{-1}(\bar{y}/\bar{x})$
f_4	$\left[\frac{[(m_{20}-m_{02})^2 + 4m_{11}^2]^{\frac{1}{2}}}{[(m_{20}-m_{02})^2 + 4m_{11}^2]^{\frac{1}{2}} + m_{20} + m_{02}} \right]^{\frac{1}{2}}$
f_5	$\tan^{-1}(2m_{11}/(m_{20} - m_{02}))$

Table 1: The computation of texture features from moments.

dual representation of the Voronoi tessellation is the *Delaunay graph* which is obtained by connecting all the pairs of points which are Voronoi neighbors as defined above.

In the following discussion, first we consider textures defined in terms of dots and then generalize our approach to arbitrary tokens. In order to segment the textured image, we compute local features in the dot pattern and then group dots with similar features to construct uniform texture regions. This also helps in identifying borders separating such uniform regions. Moments of area of the Voronoi polygons serve as a useful set of features that reflect both the spatial distribution and shapes of the tokens in the textured image. The $(p + q)^{th}$ order moments of area of a closed region R with respect to a point or dot with coordinates (x_0, y_0) are defined as [11]:

$$m_{pq} = \iint_{(x,y) \in R} (x - x_0)^p (y - y_0)^q dx dy \quad (1)$$

where $p + q = 0, 1, 2, \dots$. These moments are computed with respect to the coordinates of the dot to which the polygon belongs, so that they are comparable for each polygon across the dot pattern. The details of efficient computation of these moments for polygonal regions can be found in [27]. The lower order moments (small values of p and q) have well defined geometric interpretations. For example, m_{00} is the area of a polygon, m_{10}/m_{00} and m_{01}/m_{00} give the displacement of the polygon centroid with respect to (x_0, y_0) in the x and y directions, respectively. The m_{20} , m_{11} , and m_{02} can be used to derive the amount of elongation of a polygon, and the orientation of its major axis. The higher order moments give even more detailed shape characteristics of the polygons such as symmetry, etc.

Let (\bar{x}, \bar{y}) be the coordinates of the polygon centroid, where $\bar{x} = m_{10}/m_{00}$ and $\bar{y} = m_{01}/m_{00}$. We have used the features derived from moments of area shown in Table 1. f_2 gives the magnitude of the vector from the dot to the polygon centroid, f_3 gives its direction, f_4 gives the overall elongation of the polygon ($f_4 = 0$ for a circle), and f_5 gives the orientation of its major axis. The feature vector for a dot then is $\langle f_1, f_2, f_3, f_4, f_5 \rangle$.

How do we compute the Voronoi tessellation when the texture primitives are arbitrary shaped tokens as in Figures 3–5? We have chosen to regard a token as consisting of dots. We compute a generalized

Delaunay graph to define the neighborhood relationship among the tokens and compute the shape feature for the token in terms of the features of its constituent dots' polygons. Let \mathbf{T} be the set of tokens. The generalized Delaunay graph for the tokens is defined as follows: two tokens $t_i, t_j \in \mathbf{T}$ are neighbors or have an edge between them if there exist two dots $p_k \in t_i$ and $p_l \in t_j$ such that p_k and p_l are neighbors in the Delaunay graph for the dot pattern making up the tokens. The feature vector for the token is computed based on the features of its constituent dots. The simplest such feature for a token is the average of the feature for the dots. Note that this is an *ad hoc* measure, but it appears reasonable for the texture examples we have used. Let \overline{f}_k be the average of the k^{th} feature. Then we define the feature vector for the token as $\langle \overline{f}_1, \overline{f}_2, \overline{f}_3, \overline{f}_4, \overline{f}_5 \rangle$.

3 SEGMENTATION ALGORITHM

In this section we discuss the main steps of the segmentation algorithm. We first describe computation of the probability that a Delaunay edge e_{ij} , connecting two tokens i and j , is broken (i.e., whether the edge is inconsistent). Next we describe the probabilistic relaxation labeling to disambiguate and identify the borders of the regions.

3.1 Initial Break Probabilities on Delaunay Edges

Let e_{ij} be the Delaunay edge connecting tokens i and j . Let N_i be the set of neighboring tokens (immediate neighbors and neighbors of neighbors) for token i . Similarly, define N_j for token j . If the Delaunay edge e_{ij} lies within a single textured region, then the textural properties of the two sets of neighbors N_i and N_j should be similar. If e_{ij} connects tokens belonging to two differently textured regions, then the two sets of neighbors N_i and N_j will have different properties. The probability $P_{ij}(NB)$ that the edge e_{ij} is not broken is defined in terms of the similarity of the features of sets N_i and N_j .

Let $\mathbf{F}^k = (f_1^k, \dots, f_5^k)$ be the feature vector associated with token k . These are the derived features discussed in Section 2. We use the Kolmogorov-Smirnov (KS) statistic [19] to compare the set of features associated with the two neighborhoods N_i and N_j . Let $S_k(f_l^k)$ be the unbiased estimator of the cumulative distribution function of the l^{th} feature associated with N_k . If the feature values f_l^k are sorted, then $S_k(f_l^k)$ is the function giving the fraction of the values to the left of f_l^k . The KS statistic D_{ij}^l for comparing the similarity of the two neighborhoods N_i and N_j based on the l^{th} feature is defined as

$$D_{ij}^l = \max_{-\infty < x_l < \infty} |S_i(x_l) - S_j(x_l)|. \quad (2)$$

D_{ij}^l would be close to zero for two neighborhoods which are within a single texture. We use $P_{ij}(NB) = 1 - \max_l \{D_{ij}^l\}$. It is possible that some of the moments of area (and thus the derived features) will have similar values even if the two neighbor sets belong to different textures. For example, two textures with the same average dot density, but with different orientation trends, will have comparable zeroth- and first-order

moments but different second-order moments. We assume the worst case and pick the feature with the largest difference in the two cumulative distribution functions to compute $P_{ij}(NB)$ for the Delaunay edge e_{ij} . A high value of $P_{ij}(NB)$ means that the two tokens i and j belong to the same texture and thus the Delaunay edge e_{ij} should not be broken. We also define the probability $P_{ij}(B)$ that the edge e_{ij} is broken as $P_{ij}(B) = 1 - P_{ij}(NB)$.

3.2 Identification of Textured Regions by Probabilistic Relaxation Labeling

The probabilities $P_{ij}(NB)$ computed in the previous section are a good starting point for the region segmentation, but they rely only on local information and do not enforce Gestalt constraints such as border smoothness. To smooth local irregularities and enforce border smoothness, we formulate a probabilistic relaxation labeling scheme.

Probabilistic relaxation labeling is a technique of parallel constraint propagation for obtaining locally consistent labels for a set of objects [21,28]. It consists of a set of objects $\mathbf{T} = \{t_i, i = 1, 2, \dots, n\}$ to be labeled and a set $\mathbf{\Lambda} = \{\lambda_i, i = 1, 2, \dots, m\}$ of labels. Let $P_i(\lambda)$ be the probability that the label λ is ‘‘correct’’ for the i^{th} object, such that $\sum_{\lambda} P_i(\lambda) = 1$.

The probability $P_i^{(k+1)}(\lambda)$ of label λ on object i at iteration $(k + 1)$ is obtained by using the following rule:

$$P_i^{(k+1)}(\lambda) = \frac{P_i^{(k)}(\lambda)[1 + q_i^{(k)}(\lambda)]}{\sum_{\lambda'} P_i^{(k)}(\lambda)[1 + q_i^{(k)}(\lambda')]} \quad (3)$$

where

$$q_i^{(k)}(\lambda) = \sum_j w_{ij} \left[\max_{\lambda'} \{r_{ij}(\lambda, \lambda') P_j^{(k)}(\lambda')\} + \min_{\lambda'} \{r_{ij}(\lambda, \lambda') P_j^{(k)}(\lambda')\} \right]. \quad (4)$$

In Equation 4, $r_{ij}(\lambda, \lambda')$ is the compatibility of the labels λ and λ' on objects i and j and w_{ij} is the coefficient which determines the contribution of the j^{th} object towards the change in probability for the i^{th} object. The r_{ij} values incorporate the border smoothness constraint. The quantity $q_i^{(k)}(\lambda)$ is in the range $[-1,1]$, and it determines whether $P_i^{(k+1)}(\lambda)$ is increased, decreased, or remains unchanged after the probabilities are normalized. The iteration continues until either (i) all the probabilities in Equation (3) converge, or (ii) the number of iterations exceeds a prespecified limit.

We now look at the specific formulation of the relaxation labeling for our segmentation problem.

- (a) The set of objects to be labeled are the n tokens in the texture image, $\mathbf{T} = \{t_i, i = 1, 2, \dots, n\}$.
- (b) The set of labels $\mathbf{\Lambda} = \{I, E, N, W, S\}$, indicate whether the token is an interior token (I), or it lies on the border of a region with direction 0 degrees (E), 90 degrees (N), etc. The sense of the direction is chosen according to the right hand rule; that is, when facing in the direction of the border label, the interior of the region lies on the right hand side. In this algorithm we have decided to use four quantized directions for the border labels. One can increase the resolution by using a larger number of border directions.

- (c) The initial label probabilities are computed from the initial $P_{ij}(NB)$ computed before. For example,

$$P_i^{(0)}(I) = \frac{1}{|N_i|} \sum_{j \in N_i} P_{ij}(NB) \quad (5)$$

This is based on the expectation that in the interior of a homogeneous textured region most Delaunay edges will have high values of $P_{ij}(NB)$. The initial probability for a border label $b_k \in \{E, N, W, S\}$ is computed by

$$P_i^{(0)}(b_k) = \frac{(\frac{P_{NB}}{W_{NB}} + \frac{P_B}{W_B})}{2} \quad (6)$$

where

$$P_{NB} = \sum_j P_{ij}(NB) \frac{2(\frac{\pi}{2} - \theta)}{\pi} \quad (7)$$

$$W_{NB} = \sum_j \frac{2(\frac{\pi}{2} - \theta)}{\pi} \quad (8)$$

$$P_B = \sum_j P_{ij}(B) \frac{2\theta}{\pi} \quad (9)$$

$$W_B = \sum_j \frac{2\theta}{\pi} \quad (10)$$

In the above expressions, the terms P_{NB}/W_{NB} and P_B/W_B are weighted averages of the $P_{ij}(NB)$ and $P_{ij}(B)$ probabilities, respectively, around the token i . The weights depend on the angle θ that the Delaunay edge e_{ij} makes with the border label b_k (see Figure 1(a)). Edges are expected to be linked along the border label direction (thus the weight $2(\frac{\pi}{2} - \theta)/\pi$ for the NB label) and they are expected to be broken away from the border label direction (and hence the weight $2\theta/\pi$ for the B label). The angle θ can take on values in the range $[0, \frac{\pi}{2}]$. The initial probabilities for those dots whose Voronoi polygons are incomplete are computed differently, because in these cases we know *a priori* that the dot is not an interior dot. This computation takes advantage of the fact that the interior regions lie on the non-empty side of the border. In this case, we assign $P_i^{(0)}(I) = 0$, and proceed with the above analysis for the probabilities of border labels.

- (d) The Gestalt constraint of border smoothness is enforced through the proper use of compatibility coefficients. These compatibilities between pairs of neighboring tokens are computed based upon the local evidence for the token being in the interior or lying on the border and the enforcement of local border smoothness to obtain more perceptually meaningful segments. In the following formulas, $P_i(b_k)$ is the probability of token i having border label $b_k \in \{E, N, W, S\}$, θ_{ij} is the angle formed by the edge from token i to token j with the x-axis, and α_k is the angle of the border label b_k with the x-axis (see Figure 1(b)). The compatibilities $r_{ij}(\lambda, \lambda')$ between two tokens i and j with labels $\lambda, \lambda' \in \{I, E, N, W, S\}$ are computed as follows:

$$r_{ij}(I, I) = \frac{P_i(I) + P_j(I)}{2} \quad (11)$$

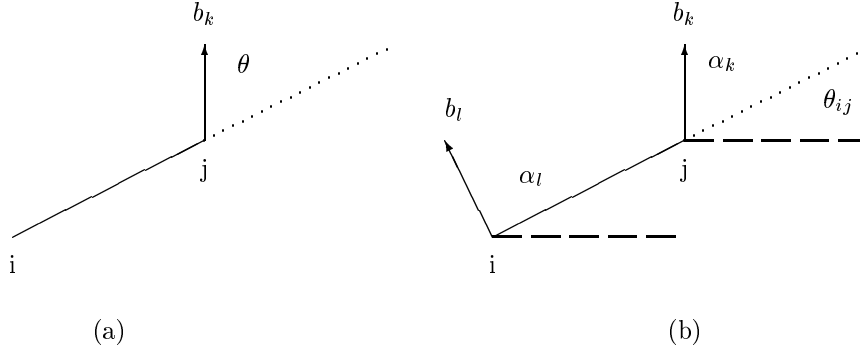


Figure 1: (a) The definitions of angles in the interior-border compatibilities. (b) The definitions of angles in the border-border compatibilities.

$$r_{ij}(I, b_k) = \left(\frac{1 + \sin(\theta_{ij} - \alpha_k)}{2} \right) \frac{P_i(I) + P_j(b_k)}{2} \quad (12)$$

$$r_{ij}(b_k, I) = \left(\frac{1 - \sin(\theta_{ij} - \alpha_k)}{2} \right) \frac{P_j(I) + P_i(b_k)}{2} \quad (13)$$

$$r_{ij}(b_k, b_l) = \left(\frac{P_i(b_k) + P_j(b_l)}{2} \right) S_{angle} \quad (14)$$

where

$$S_{angle} = \frac{1}{2} \left(1 + \left(\frac{\cos 2(\alpha_k - \theta_{ij}) + \cos 2(\alpha_l - \theta_{ij})}{2} \cos(\alpha_k - \alpha_l) \right) \right) \quad (15)$$

In these expressions, P_i and P_j provide the local evidence for the particular labels whose compatibilities are computed. The $\sin(\cdot)$ terms in Equations (12) and (13) enforce the right hand rule for the border direction. That is, the support provided by the $\sin(\cdot)$ term is positive if the interior labeled dot is on the right hand side of the border direction. S_{angle} in Equation (14) enforces two conditions: (i) When the border tokens are along the contour of a single region, they must be aligned and pointing in the same direction ($\alpha_k - \theta_{ij} = \alpha_l - \theta_{ij} = \alpha_k - \alpha_l = 0$). (ii) When the border tokens are along the contours of two separate regions, they should be pointing in opposite directions (e.g., $\theta_{ij} = 0$, $\alpha_k = \pi/2$, $\alpha_l = -\pi/2$). This is done in order to keep the interior of each region on the right side of each border label. The space between two such border tokens is the crack between two texture regions.

This relaxation labeling algorithm is run on the tokens to obtain the final labeling of the interior regions and the borders of the texture. In running the relaxation labeling algorithm, we update our compatibility coefficients after a certain number of iterations (5 iterations in this case). This updating reflects the effect of

changing probabilities of the labels on the compatibilities of neighboring tokens. The relaxation algorithm is run for 100 iterations in total. In most instances the label probabilities converge to their final values in fewer iterations. However, for certain ambiguous cases, it takes longer for the values to settle.

3.3 Gray Level Texture Images

In order to apply our texture segmentation algorithm to gray level images, we need to first extract tokens from images. Some researchers have developed complex methods for extracting such texture elements from gray level images [3,25]. In this paper, we have used a simple algorithm to extract tokens from texture images to demonstrate that our texture segmentation algorithm can be applied not only to synthetic dot patterns but to gray level images as well.

We extract tokens from a gray level image as follows:

1. Apply Laplacian of a Gaussian (LoG or $\nabla^2 G$) filter to the image. For computational efficiency, we approximated the $\nabla^2 G$ filter with a difference of Gaussians (DoG) filter. The size of the DoG filter is determined by the sizes of the two Gaussian filters. We used $\sigma_1 = 1$ for the first Gaussian and $\sigma_2 = 1.6\sigma_1$ for the second one. According to Marr, this is the ratio at which a DoG filter best approximates the corresponding $\nabla^2 G$ filter [18, page 63].
2. Select those pixels that lie on a local peak of the filtered image. A pixel in the filtered image is said to be on a local peak if its magnitude is larger than six or more of its eight nearest neighbors. This definition allows us to identify those pixels that lie along a ridge. This results in a binary image (e.g. Figure 4(c)).
3. Perform a connected component analysis on the binary image using eight nearest neighbors. Each connected component defines a texture primitive (token). We have selected a fairly simple method of token extraction. Several other methods of token extraction have recently been proposed [3,25].

After the texture tokens are identified using this method, we apply the algorithm described in Sections 3.1 and 3.2 to segment the texture image.

4 EXPERIMENTAL RESULTS

We have tested our segmentation algorithm on three classes of texture patterns. The first class consists of textures which are made of dots, an example of which is seen in Figure 2(a). These textures differ in the regularity of the dot placement, the direction of the dot density, and the average dot density. In some of the regular textures we have added small amounts of perturbations to the positions of the dots. The second class of textures have primitives made up of dots. Figure 3(a) gives a popular example of such textures seen in the literature [13]. In this example, the tokens which are used to construct the textures have identical

second-order statistics. The third class of texture patterns were taken from the texture album of Brodatz [4]. Several gray level images of textures were put together in pairs to test our segmentation algorithm. Figures 4(a)–5(a) are examples of such texture pairs. These are 128x256 images (each Brodatz texture is 128x128) and they have a range of 256 gray levels.

Our segmentation algorithm can successfully segment the textured regions of the first class (Figure 2(a)). Figure 2(b) shows the probabilities $P_{ij}(NB)$ for the Delaunay edges for the textures in Figure 2(a). The probabilities are shown as gray levels such that the lighter edges have higher probabilities of being broken. As can be seen from Figures 2(a) and 2(b), the initial estimates of $P_{ij}^{(0)}(NB)$ are quite useful in identifying the interior regions (where most edges are not broken) and the borders between textured regions (where most Delaunay edges are weak). The final segmentation after running the relaxation labeling is shown in Figure 2(c). In the segmentation results the tokens are represented as dots at their centroids for graphical representation. Note that the borders in Figure 2(c) are thick and fuzzy due to the local nonuniformity of the shapes of the Voronoi polygons going from one region to the next.

Our algorithm also successfully segments the “corner-closure” texture pattern (Figure 3(a)). In Figure 3(b), we are able to successfully label the interiors of the two textures and identify the correct location of the border. There are some textures with identical second-order statistics that our algorithm is unable to segment, but these textures are not preattentively discriminable by humans, either. We have also run the segmentation algorithm on patterns that consist of textures made up of L’s and crosses, L’s and T’s, etc. [1]. In each of these cases, the algorithm finds the borders and the interiors of the textures properly.

Finally, the algorithm successfully segments most of the texture pairs we used from the Brodatz texture album [4]. Figures 4(a)–5(a) give the input gray level images. Figures 4(b)–5(b) give the result of convolving these images with the DoG filter ($\sigma_1 = 1, \sigma_2 = 1.6\sigma_1$). Figures 4(c)–5(c) give the peaks that are detected from the convolved images. These are the tokens used to segment the textures. The results of the segmentation is shown in Figures 4(d)–5(d). In these segmentation results also the tokens are represented as dots at their centroids for graphical representation.

The segmentation results of Figures 4 and 5 detect the borders and the interior regions of the texture pairs. There are some tokens in the interior regions of some of the textures (Figure 5) that are labeled as borders. There are two reasons for this: (a) Our comparison of statistical similarity of features is local and thus it is detecting the differences in these local regions, (b) The tokens actually have different local properties (e.g. in Figure 5(c), the wood grain texture has long vertical tokens intermixed with short and horizontal tokens). In some texture pairs (e.g., D9 grass lawn and D17 herringbone weave), complete segmentation is not achieved but a large portion of the border between the two textures is detected. We feel that better token extraction, increasing the size of the neighborhoods around each token, and some postprocessing will improve the performance of our segmentation algorithm.

5 CONCLUSION

In this paper we have developed a texture segmentation algorithm based on the Voronoi tessellation. The algorithm first builds the Voronoi tessellation of the tokens that make up the textured image, computes a set of features from the Voronoi diagrams, and finally, it performs a probabilistic relaxation labeling on the tokens, to identify the interior and the border regions of the textures present in the image.

The results of the segmentation algorithm show that the shapes of the Voronoi polygons provide a powerful set of features that reflect certain textural properties in images. Particularly, the results on the “corner-closure” type texture patterns of Figure 3 show that the important aspects of the textural properties have been captured by this representation.

An important advantage of our segmentation algorithm over other texture segmentation algorithms is that we do not have to specify the number of texture regions (segments) *a priori*. The number of regions is automatically found by identifying contiguous interior regions. The moment-based features are able to capture orientation sensitive texture properties, as well as more complicated textural properties such as textons proposed by Julesz [13] (see, for example, Figure 3(a)). Our segmentation algorithm tries to incorporate Gestalt perceptual criteria such as border smoothness. In many of the texture patterns, these Gestalt criteria improve the segmentation results as seen in Figure 2.

Future research directions range from improvements in the current algorithm to its integration with other intermediate level visual processing modules. We need to improve the method used in comparing the textural properties of the two neighborhoods of the Delaunay edge (Section 3.1). The current algorithm is sensitive to small perturbations in the locations of the tokens if the token placement is very regular. The fact that in the regular regions the variance of the features is almost zero makes the statistical comparison very sensitive to small perturbations in such areas. This results in border labels being assigned to interior tokens because slight perturbations in the location of the tokens cause the feature distributions to be seen as different.

The localization of the actual boundary has not been addressed in this paper. Some post processing is also necessary to thin the boundary and smooth the texture regions. This step may also improve the results for regular regions with very small perturbations discussed above.

In the case of the gray level texture images, we used a very simple method of token extraction. This process can be improved further which will result in better segmentation. The use of multiscale filtering in the token extraction process is one possibility.

A major extension of this work would be in the integration of the segmentation process with other visual processing modules. One immediate area is the shape from texture processing where the boundaries obtained from the texture segmentation may be used in localizing the process of surface geometry extraction. A feedback from the shape from texture algorithms may in turn provide useful information about possible borders between textured regions.

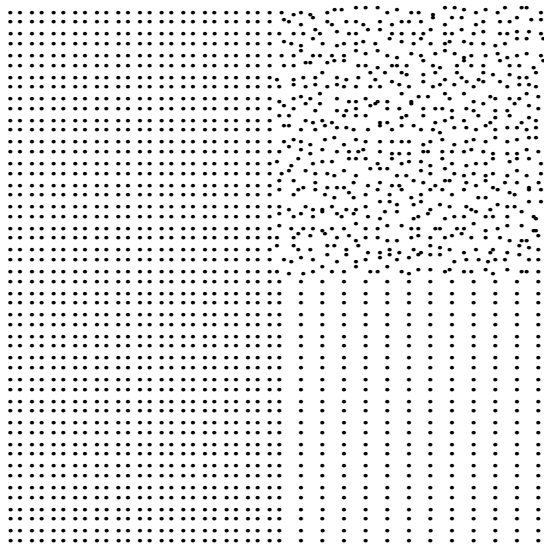
ACKNOWLEDGEMENTS

The implementation of the algorithm described in Sections 3.1 and 3.2 was partly done by Mrs. Ywhyng Lee Hoffman. Miss Sally Howden has also helped in parts of the implementation.

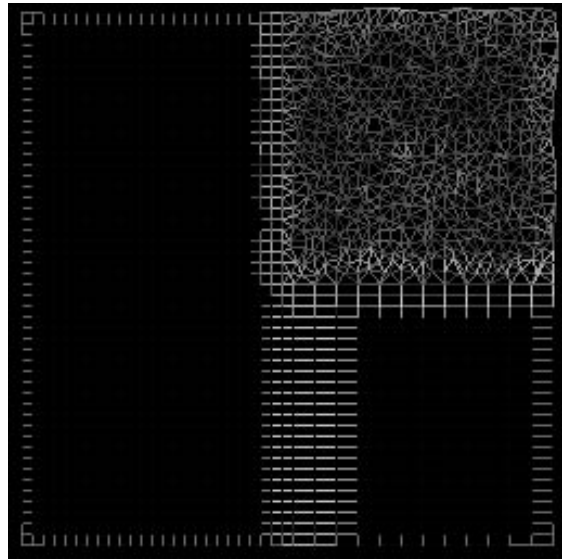
References

- [1] J. Beck, K. Prazdny, and A. Rosenfeld. A theory of textural segmentation. In Jacob Beck, Barbara Hope, and Azriel Rosenfeld, editors, *Human and Machine Vision*, pages 1–38, Academic Press, New York, 1983.
- [2] J. Beck, A. Sutter, and R. Ivry. Spatial frequency channels and perceptual grouping in texture segregation. *Computer Vision, Graphics, and Image Processing*, 37:299–325, 1987.
- [3] D. Blostein and N. Ahuja. Representation and three-dimensional interpretation of image texture: an integrated approach. In *Proc. of First International Conference on Computer Vision*, pages 444–449, 1987.
- [4] P. Brodatz. *Textures: a photographic album for artists and designers*. Dover Publications, Inc., New York, 1966.
- [5] M. Clark and A. C. Bovik. Texture segmentation using Gabor modulation/demodulation. *Pattern Recognition Letters*, 6:261–267, Sept. 1987.
- [6] J. M. Coggins and A. K. Jain. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, 3:195–203, 1985.
- [7] G. C. Cross and A. K. Jain. Markov random field texture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5:25–39, 1983.
- [8] L. S. Davis, M. Clearman, and J. K. Aggarwal. An empirical evaluation of generalized cooccurrence matrices. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-3:214–221, 1981.
- [9] A. Gagalowicz. Blind texture segmentation. In *Proc. of the 9th International Conference on Pattern Recognition*, pages 46–50, Rome, Italy, November 1988.
- [10] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, 1979.
- [11] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. on Information Theory*, IT-8:179–187, 1962.
- [12] B. Julesz. Texton gradients: the texton theory revisited. *Biological Cybernetics*, 54:245–251, 1986.

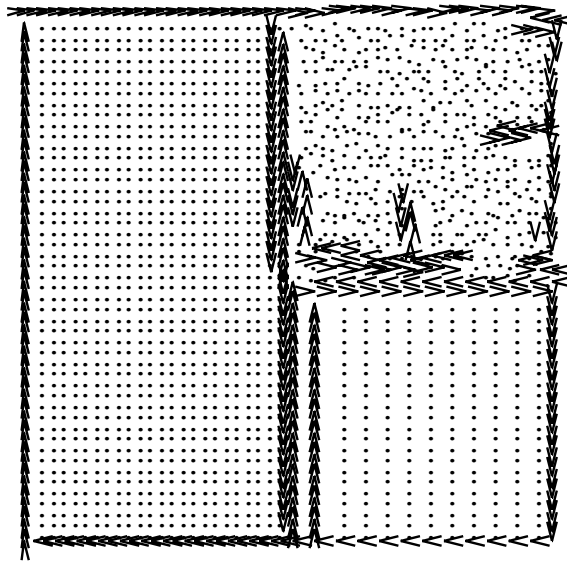
- [13] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981.
- [14] B. Julesz. Visual pattern discrimination. *IRE Trans. on Information Theory*, IT-8:84–92, 1962.
- [15] B. Julesz, E. N. Gilbert, L. A. Shepp, and H. L. Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics - revisited. *Perception*, 2:391–405, 1973.
- [16] R. L. Kashyap and R. Chellappa. Estimation and choice of neighbors in spatial interaction models of images. *IEEE Trans. on Information Theory*, 29:60–72, 1983.
- [17] R. L. Kashyap, R. Chellappa, and N. Ahuja. Decision rules for the choice of neighbors in random field models of images. *Computer Graphics and Image Processing*, 15:301–318, 1981.
- [18] D. Marr. *Vision*. W. H. Freeman and Company, San Francisco, CA, 1982.
- [19] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, New York, 1988.
- [20] T. C. Rearick. A texture analysis algorithm inspired by a theory of preattentive vision. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 312–317, San Francisco, CA, 1985.
- [21] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-6:420–433, 1976.
- [22] M. I. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th Annual Symp. on Foundations of Computer Science*, pages 131–162, 1975.
- [23] M. Tuceryan and N. Ahuja. Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt. *Computer Vision, Graphics, and Image Processing*, 1989. To appear in December.
- [24] M. R. Turner. Texture discrimination by Gabor functions. *Biological Cybernetics*, 55:71–82, 1986.
- [25] H. Voorhees and T. Poggio. Detecting textons and texture boundaries in natural images. In *Proc. of First International Conference on Computer Vision*, pages 250–258, London, 1987.
- [26] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire: recherches sur les paralléloèdres primitifs. *J. Reine Angew. Math.*, 134:198–287, 1908.
- [27] H. B. Wilson Jr. and D. S. Farrior. Computation of geometrical and inertial properties for general areas and volumes of revolution. *Computer Aided Design*, 8(4):257–263, 1976.
- [28] S. W. Zucker and R. A. Hummel. On the foundations of relaxation labeling processes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5:267–287, 1983.



(a)



(b)



(c)

Figure 2: (a) A texture with three regions. Two regions consist of regularly placed dots with different densities. In the third region the dots are randomly placed. One of the regularly placed regions has a directionally sensitive density. (b) $P_{ij}^{(0)}(NB)$ assigned to each Delaunay edge for the tokens in (a). The probabilities are shown as gray levels such that the lighter edges have higher probabilities of being broken. (c) Segmentation showing the interior regions (shown as dots) and the borders (shown as arrows in four directions) for the texture in (a).

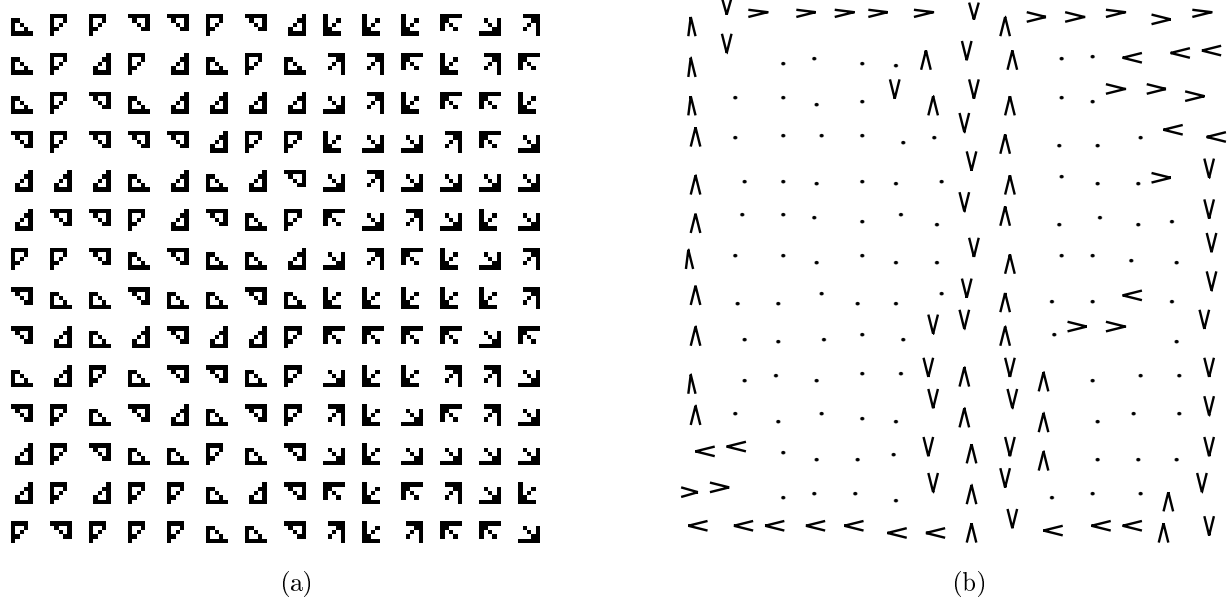


Figure 3: (a) An example texture pair with identical second-order statistics that can be preattentively discriminated. (b) The resulting segmentation produced by the algorithm. The dots are the place holders for the token and represent interior regions.

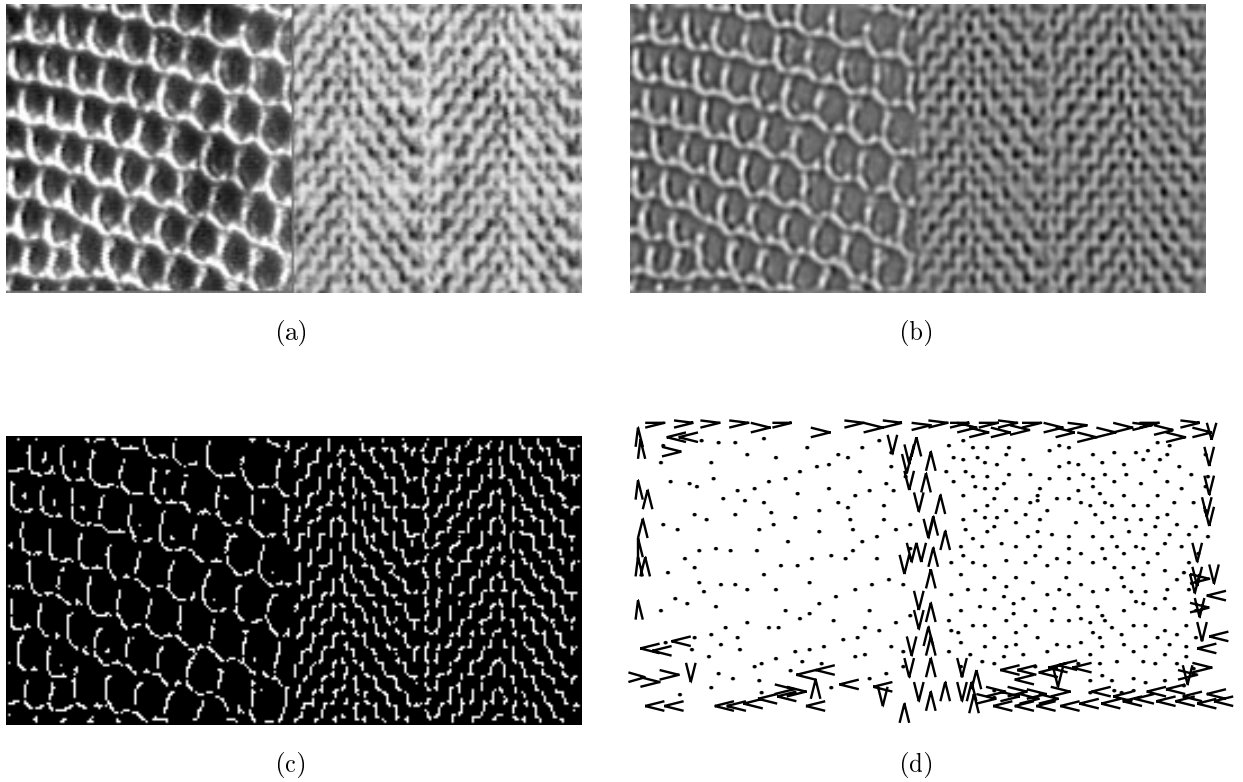


Figure 4: (a) Texture pair D3 (reptile skin) and D17 (herringbone weave) from Brodatz texture book. (b) Image in (a) convolved with a DoG filter with $\sigma_1 = 1$ and $\sigma_2 = 1.6$. (c) Detected peaks in the filtered image. The connected components form the tokens. (d) Resulting segmentation. Each token is displayed as a dot at its centroid.

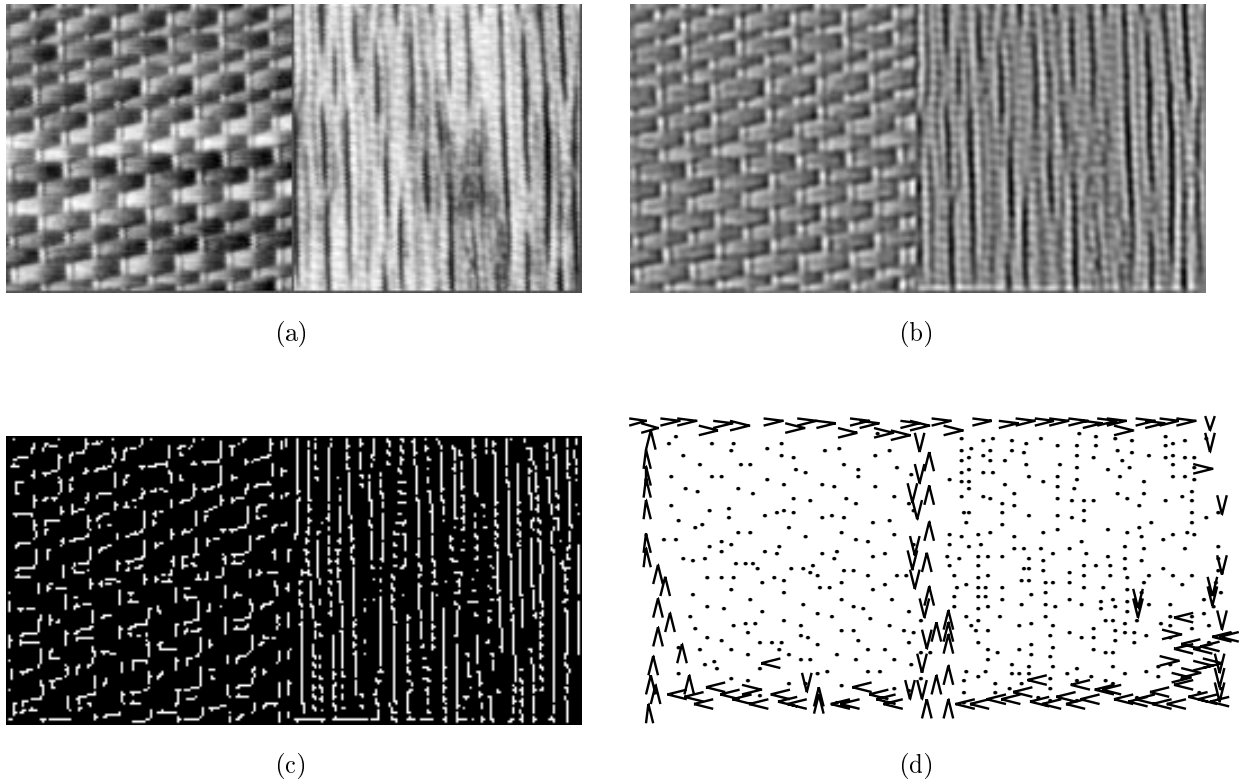


Figure 5: (a) Texture pair D55 (straw matting) and D68 (wood grain) from the Brodatz texture book. (b) Image in (a) convolved with a DoG filter with $\sigma_1 = 1$ and $\sigma_2 = 1.6$. (c) Detected peaks in the filtered image. (d) Resulting segmentation. Each token is displayed as a dot at its centroid.
