EDGE SEPARATION AND ORIENTATION TEXTURE MEASURES

by

Bradley Kjell
and
Charles R. Dyer

Computer Sciences Technical Report #559

October 1984

Edge Separation and Orientation Texture Measures

Bradley Kjell
Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, WI 53706

Abstract

A method is presented for describing natural textures using average separations between extended edge segments of different orientations. Edge separations are found by a label propagation process similar to the growth of discrete generalized Voronoi diagrams. These features were used to classify 384 samples from six texture classes. The highest classification rate was 65.9% correct using one feature, 82.6% correct using two features, and 85.9% correct using three features. Edge separation features are shown generally to perform better than either gray level cooccurrence matrices or Laws' texture energy measures. The best features were the average separation between pairs of antiparallel or parallel edge segments. Extensions to the method to deal with problems of rotation and scale are discussed.

This research was supported by the National Science Foundation under grant MCS81-05904.

1. Introduction

Approaches to texture have been investigated by many authors. In his review article, Haralick classifies texture methods by i) the type of primitives used to describe the texture, and ii) the description of the spatial relation between these primitives [HARA79]. For gray level based methods, the primitive is the gray level of a single pixel or of a fixed neighborhood of pixels. The spatial relation between pixels can be expressed by any of several means, such as gray level cooccurrence matrices [HARA73], autocorrelation, or by gray level differences. In token based methods, the primitives are found by examining a region of the image. Typical of such primitives are edges [NEVA79][PIET82a], peaks [EHRI78], and regions [TOMI82]. Spatial relations between these tokens might be expressed by the angles and distances between them, or by generalized cooccurrence matrices [DAVI79] [DAVI81].

Texture perception in humans may be similar to a token-based method. Bela Julesz has found evidence that human texture discrimination is based on simple statistics of image features that he calls textons [JULE81] [JULE83]. Similar evidence has been reported by Jacob Beck [BECK80] [BECK81]. The textons of Julesz are elongated blobs (eg. rectangles, ellipses, or line segments), ends-of-lines, and crossings of lines. Among the properties of elongated blobs are color, angular orientation, width, and length. The last two textons, ends-of-lines and crossings of lines are characteristics of the first texton. Typical experiments of Julesz and colleagues have human subjects discriminate between two synthetic textures, each of which is made of thin black lines on a white background. The type of texton or the number of textons might differ in each texture. For example, one texture might be composed of lines arranged into L's, the other texture might have exactly the same lines but arranged into +'s (thus introducing the line-crossing texton). Julesz finds that humans can quickly discriminate between textures if the textons in them differ or if the densities of textons differ. The simplicity of this explanation suggests that automated texture discrimination could be based on lines found in textures.

Julesz and Beck are both concerned with rapid (less than 200ms) texture discrimination in humans. Julesz calls this preattentive texture perception. Both authors find that differences in first-order statistics of texture tokens are sufficient to explain human performance (see the papers cited above). For texture classification by computer, better descriptions of spatial distributions are needed. In the synthetic textures usually used in human perception experiments the texture primitives can be easily separated from the background. Determining their type and number is simple. This is a much more difficult problem with full gray scale images of natural textures. Feature detectors work under uncertainty. The detection of features and their attributes often depends on arbitrary definitions and thresholds. For example, the exact location of edge terminations usually is not precisely determined by most edge detectors. The output of an edge mask will gradually taper off near the end of an edge. Determining exactly where an edge ends becomes a threshold problem. The behavior of an edge detector near a sharp corner presents further problems. Near the corner the output becomes confused. Often either one edge mask will dominate, or the output of all masks will be low.

Spatial descriptions are needed to help deal with these problems. Generalized cooccurrence matrices have been used for spatial descriptions, but they have not proven very successful (see Section 8 for a longer discussion). What is needed is a description that is matched to the texture primitives being used. If the texture primitives are features that have clear physical meanings then the description of their spatial distribution should also have a physical meaning.

The texture method discussed in this report addresses both problems: finding texture primitives, and describing their spatial relations. The texture primitives we use are long single-orientation edge segments. The spatial description of these long edge segments consists of a matrix giving the average distance separating edges of each orientation. Features based on this matrix can, for example, easily discriminate between the L and + textures discussed above. This method has several advantages over other texture methods: the edges can be used for other stages of a general vision program, the spatial description has physical meaning (unlike most cooccurrence matrix features, for example), and the method can easily be adapted to deal with problems of rotation and scale. Texture classification experiments show that this method works well. Results are much better than for gray level cooccurrence matrices, and are somewhat better than for Laws' texture energy features.

Other texture methods have used edge-based features. In [HONG80] and [WANG81] simple edges were used for finding regions. These regions were then used to describe textures. In [PIET82a] quantities were computed for antiparallel pairs of edge pixels. One computed quantity was average distance between edge pixel pairs, however no edge orientation information was used. Distributions of simple edges have been described by cooccurrence matrices [NEVA79]. Our method differs from these methods by using extended edge segments that have been formed by grouping many edge pixels. These extended edges are a higher level token than the edge pixels that comprise them. Extended edges were used in [DAVI79] and [DAVI81] where their spatial properties were described by generalized cooccurrence matrices.

Sections 2 and 3 discuss the method in greater detail. Section 4 gives classification results. Comparison with standard texture methods is made in Section 5. Combining the features of this method with average edge pixel orientation features results in good classification rates. These results are given in Section 6. The robustness of the method is demonstrated in Section 7 where a different type of edge finder is used in the first step of the method and comparable results are obtained. Finally, Section 8 contains a discussion of the method and directions for further investigation.

2. Edge Detection

The first stage in the method is to apply an edge finder to the image. The goal of this edge finder is to produce long, connected edges consisting of edge pixels of only one orientation. Since each of these edge segments has a single orientation, it is possible to define simple distance measures between edge segments of different orientations. The output of the edge finding stage is similar to a pen-and-ink sketch where the artist has built up the sketch with a series of thin, straight lines. The following subsections present the steps used to produce the edges.

2.1. Edge Maps

Edge maps are produced by applying eight three by three edge templates at each pixel in the image. The greatest response of the eight masks will be the magnitude of an edge pixel and the orientation of that mask will be the edge pixel orientation. Orientations are from 0 degrees to 315 degrees in steps of 45 degrees. Two maps are produced: an edge magnitude map and an edge orientation map. Fig. 1 shows a straw texture from Brodatz' album of textures (Plate D92) [BROD66]. Fig. 2 shows the magnitude portion of the edge pixel map. The edge map is now separated into eight edge magnitude maps by placing nonzero edge pixels of each orientation in a separate map. Fig. 3 shows the magnitude map for pixels with orientation 90 degrees for the straw texture.

2.1.1. Smoothing the Edge Maps

The preceding step produces thick, ragged edges that have occasional gaps. Smoothing these edges will enhance the continuity of the thin edges. Each of the eight edge magnitude maps is smoothed separately. This is done by convolving each map with a long, thin averaging mask whose long dimension is in the same orientation as the map. Because each map has pixels of only one orientation, these long masks will smooth edges along their length, but will not blur edges perpendicular to their orientation. In the experiments done here, the following type of averaging masks were used:

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{array} \qquad \begin{array}{ccccc} 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \end{array}$$

0 deg. 45 deg.

Similar masks were used for the other orientations. Fig. 4 shows the result of this stage on the straw image.

The exact form of this smoothing step does not appear to be critical: the above masks were chosen from a small number of trials based on the visual appearance of the final edges. In particular, 1x5 masks produced nearly the same results as these 3x5 masks, both in the final edges, and in the classification of a small number of texture samples.

2.2. Edge Thinning

The result of smoothing is a set of thick edge maps. To produce the desired thin edges the thick edges must be thinned to single pixel width. Such thinning is often done by "nonmaximum suppression". However, here again the fact that the edge magnitude maps are separated by orientation can be used to advantage. In a single smoothed edge magnitude map all edges have the same orientation, and a thick edge is always bordered by zeros. Consider scanning along a line that is perpendicular to the orientation of an edge. The pixels will start out at zero, will have nonzero magnitudes for a while, then fall to zero again. The edge can be thinned by retaining the "central" pixel of each such scan line, setting the rest of the pixels along each scan line to zero. The problem is to determine the best central pixel in each scan line. Denote position along a given scan line by D. For example, in the vertical edge map D is a pixel's column coordinate in the image. By definition, pixels with the same value of D occur on a single vertical line. The central "spine" of a thick vertical edge should consist of a line of pixels with nearly the same D value. The values for D of the central spine may vary slightly, however, because the edge may actually meander somewhat or not be oriented at an exact multiple of 45 degrees.

In summary, to thin a thick edge the following algorithm is used:

```
foreach scan line perpendicular to the map orientation do
   begin
   foreach nonzero run of edge pixels do
      begin
      Sum := 0;
      MagSum := 0;
      D   := position along scan line of first pixel of this run;
      foreach pixel in this run do
         begin
         MagSum := MagSum+PixelMagnitude;
         Sum   := Sum + D*PixelMagnitude;
         D     := D+1;
         end;
      AverageD := round( Sum/MagSum );
      Send to the output image the single central pixel of this
      run with position along this scan line of AverageD.
      end;
   end;
```

A weighted average is used so that strong edge pixels near the center of the edge will count more in determining the center than weak edge pixels near the sides of a thick edge. As an example, consider the following thick edge, in a 90 degree orientation edge magnitude map:

```
1252
11252
23241
 1131
 1211
1526
```

Since the edge direction is vertical, scan lines are along rows of the image. Calculation of the central pixel to retain for each row of this edge consists of calculating the weighted average column number, as shown in Table 1.

column coordinate distance D					weighted	weighted	
0	1	2	3	4	sum	avg	round
1	2	5	2		18	1.8	2
1	1	2	5	2	28	2.6	3
2	3	2	4	1	23	1.9	2
	1	1	3	1	16	2.6	3
	1	2	1	1	12	2.4	2
1	5	2	6		27	1.9	2

Table 1. Calculation of best central pixel in each horizontal scan line. The starting column coordinate in this example is arbitrary.

The result is a long, thin, vertical edge:

```
5
 5
2
 3
2
2
```

This thinning algorithm is not confused by plateaus or by several maxima per scan line. In real images edges are frequently thicker than the example given above and the values are smoother as a result of the smoothing operation. However, it is still useful to smooth the local jitter in the resulting thin edge (in the above example the column coordinates of the retained central pixels occasionally shift by one). This can be done by extending the idea of computing the weighted average distance along a scan line. To do this, rather than computing the average D value along a single scan line, the average D is computed using the nonzero edge pixels in three adjacent scan lines. This average value determines the best central pixel in the middle scan line. In the above example, this revised method results in a straight line of pixels in column two.

A disadvantage of this method is that when thick edges with a diagonal orientation are thinned, the scan lines should start in every other column (or row) in order to prevent thinned edges of width two. This effect is shown in the following example:

```
            every     every
start       scan      other
                      scan

   22        22        2
  121        2
 122        22         2
 12         2         2
121        21        2
21        21        2
```

Unfortunately, skipping every other scan line will occasionally introduce single pixel gaps in the thinned edges. This will be corrected in the gap filling step described below.

The result of the thinning operation on the 90 degree edges of the straw texture is shown in Fig. 5.

2.3. Filling Gaps

The next step takes the thinned edge maps and creates new maps for each orientation which have single pixel gaps filled in and isolated edge pixels removed. First, a pass which labels edge pixels having less than two edge pixel neighbors is made over each edge map. A second pass searches for pairs of labeled pixels which have a single pixel gap between them and fills the gaps. This step is not allowed to create branches in the edge segments. A final pass removes any remaining isolated edge pixels. The result of gap filling is shown in Fig. 6.

2.4. Combining the Edge Maps

The final step combines the eight edge maps into a single composite representation. The first step uniquely labels each connected edge segment in each of the orientation maps. This labeling is easy since edges don't branch and all edges in a single map are in the same orientation. To avoid duplicate labels between different orientation maps, and as a convenience to later programs. the edge labels also encode orientation. Note that at this step all edge magnitude information is lost. An edge in the image now refers to a connected chain of pixels with the same label.

The labeled edges are now combined into a single data structure. Since orientation maps are processed independently, a single pixel may. in general. have several labels associated with it. Therefore, the final result is an image of labeled edges where several different labels may be associated with a single pixel where edges meet or cross. Fig. 7 shows the final result for the straw texture. In this figure any pixel with an edge through it is white. all other pixels are black.

3. Edge Separation Calculation

In this section we define new features to be used for texture classification based on the average separation between edge segments of various orientations. Edge separation features will be calculated by a method similar to growing generalized Voronoi regions [LEE81] [PHIL83]. The following paragraphs will describe this method in detail.

The input to the algorithm is the image of labeled edges described in Section 2. Remember that this is an image of long, thin edges where each edge consists of connected edge pixels of a single orientation (i.e. having the same label). A pixel may have more than one label. The initial image is considered generation 0. In subsequent generations, all edges will be progressively thickened by expanding their perimeters. That is, at each generation each labeled pixel that finds a neighbor without that same label, propagates its label to that neighbor. It does not matter what other labels the neighbor has. The thickening of each edge is independent of all the others (although sometimes information will be recorded when two different edge labels first meet). Thus the thickened edges will, in general, overlap. In the results reported in Section 4, "neighbor" means four-neighbor.

In this study we are interested in computing features relating closest pairs of edges at various orientations. Therefore, rather than construct a complete description of all pairs of edges, we compute for each edge the minimum distance to an edge of each of the eight possible types (orientations). That is, each label computes a table of the form:

edge orientation							
0	1	2	3	4	5	6	7
d0	d1	d2	d3	d4	d5	d6	d7

For example, consider an edge labeled X. Each di specifies half the distance to the nearest edge of orientation i. This is implemented as follows. Initially all the di's for label X are zero. When label X is propagated to a pixel that has a different label, say label Y with orientation j, the value of dj for label X will be inspected. If it is zero, it will be set to the current generation number; if it is nonzero, it will not be altered. If edge X has orientation k then the value of dk for label Y is also inspected and set to the current generation number if dk is currently zero. Since the orientation of a edge is encoded by its label, every edge segment can progressively expand in parallel.

To summarize, each edge computes eight "distances" to its nearest edges of each orientation as follows:

```
Let dir(X) be the orientation of the edge labeled x.

d := 0;
for i:=1 to NumLabels do
   for j:=0 to 7 do
      i.table[j] := 0;

while any table entry is 0 do
   begin
   d := d + 1;
   /* do this in parallel at each pixel */
   foreach pixel P do
      foreach 4-neighbor N of P do
         foreach label X at P which is not at N do
            begin
            foreach label Y at N do
               if Y.table[ dir(X) ] = 0 then
                  Y.table[ dir(X) ] := d;
            LabelSet(N) := LabelSet(N) ∪ {X};
            end;
   end;
```

In the following example, edges are labeled with their direction and "*" represents a pixel with several labels.

```
. . . . . . . . . . .
. . 2 . . . . . . . .
. . 2 . . . . 1 . . .
. . 2 . . . 1 . . . .   Generation 0
. . 2 . . 1 . . . . .
. . 2 . 1 . . . 6 . .
. . . . . . . . 6 . .
. . . . . . . . 6 . .
. . . . . . . . 6 . .

. . 2 . . . . . . . .
. 2 2 2 . . . 1 . . .
. 2 2 2 . . 1 1 1 . .
. 2 2 2 . 1 1 1 . . .   Generation 1
. 2 2 2 1 1 1 . 6 . .   edge1.table[2] := 1
. 2 2 * 1 1 . 6 6 6 .   edge2.table[1] := 1
. . 2 . . . . 6 6 6 .
. . . . . . . 6 6 6 .
. . . . . . . 6 6 6 .

. 2 2 2 . . . 1 . . .
2 2 2 2 2 . 1 1 1 . .
2 2 2 2 2 1 1 1 1 1 .
2 2 2 2 * 1 1 1 * . .   Generation 2
2 2 2 * 1 1 1 * 6 6 .   edge1.table[6] := 2
2 2 * * 1 1 * 6 6 6 6   edge6.table[1] := 2
. 2 2 * 1 1 6 6 6 6 6
. . 2 . . . 6 6 6 6 6
. . . . . . 6 6 6 6 6
```

The generation number recorded for each orientation for each edge is roughly half the 4-distance between the edge and the closest edge to it at the specified orientation. There are some anomalies with this distance measure. First, the distance between two edges separated by an odd number of pixels will be the same as if they had been separated by one less pixel. Also, the closest neighbor relation is not symmetric. See the following example, where edge b is the closest 90 degree edge to the 45 degree edge c, but c is the closest 45 degree edge to the 90 degree edge a.

```
. a . . b . . . . . c
. a . . b . . . . c .
. a . . b . . . c . .
. a . . b . . c . . .
. a . . b . c . . . .
```

3.1. Edge Separation Features

The growth process stops when all labels have a distance recorded for each orientation. Next all the edges of orientation 0 are examined and the average separations between these edges and edges of each of the eight orientations are calculated. This is repeated for edges of each orientation. As a result, an 8 by 8 matrix of edge average separation distance is generated. The eight distances for edges of orientation 0 will be the first row of this matrix, the eight distances for edges of orientation 1 will be the second row, and so on. Because the closest neighbor relation is not symmetric, the matrix will not be symmetric.

Table 2 is an example of this matrix with values from the straw texture. Notice that the values in the table reflect structural features visible in Fig. 1. The blades of straw are mostly vertical or slanting left. The values along the major diagonal are usually large since these are the average separations between edges of the same orientation. For 0 degree horizontal edges (row 0) the closest edges are 90 degrees (column 2) or 315 degrees (column 7). This is reasonable because with the vertical emphasis of this texture any horizontal edge will soon approach a vertical or leftward diagonal edge. The greatest separation for horizontal edges is for 0 degrees (column 0) and 225 degrees (column 5). This is again reasonable because for vertical or left-ward vertical straw blades, horizontal edges are few and widely separated. As would be expected, the separation distances for 90 degree vertical edges (row 2) show almost the opposite pattern.

	average distance to							
from	0	1	2	3	4	5	6	7
0	3.22	1.87	1.09	1.83	2.17	3.09	1.44	1.09
1	1.08	4.92	1.15	1.08	1.85	2.92	1.31	1.31
2	1.78	2.44	3.17	1.17	1.52	2.13	1.52	1.26
3	2.75	2.47	1.19	2.55	1.97	2.44	1.44	1.81
4	3.45	2.90	1.45	1.32	2.65	2.58	1.55	1.74
5	3.05	3.33	1.76	1.33	1.52	3.62	1.10	1.24
6	1.96	2.64	1.57	1.68	1.82	1.82	2.29	1.50
7	2.46	2.66	1.54	1.49	2.14	2.26	1.06	2.40

Table 2. Edge separation features for a 64x64 sample of straw texture. Entry at row column c is the average distance from edges of orientation r to edges of orientation c.

This average edge separation distance matrix contains a large number of features: the square of the number of orientations. These features are not all independent of each other. As will be seen below, many of these features are similar and do not yield improved classification rates when more than a few are used.

Other features could be calculated from this matrix to reduce the amount of information. A possibility is to use descriptors that have been successful with cooccurrence matrices, but ideally descriptors should capture physically meaningful characteristics of the edge separation distribution. Preliminary experiments with cooccurrence-like descriptors derived from the matrix of edge separation features resulted in lower classification rates that using the original features. Another possible set of descriptors would be the higher moments (standard deviation, skew, and kurtosis) of each of the edge separations. Each higher moment would result in another full matrix of features. No experimentation has been done with these descriptors.

4. Classification Results

The procedure outlined in Section 2 and Section 3 was performed on six textures from Brodatz' album of textures [BROD66] as seen in Table 3. Each 64 by 64 block of these textures was histogram flattened to 64 gray levels [ROSE82]. This was done by histograming the center-weighted sum of each three by three neighborhood, and partitioning this histogram into sixty-four bins. These textures are similar and in some cases identical (except, perhaps, in scale) to the ones used in Laws [LAWS79a,b].

texture	Brodatz' plate number	inches per 256 pixels
cork	plate D4	3.5″
grass	plate D9	3.5″
pig	plate D92	3.5″
straw	plate D15	3.5″
raffia	plate D84	3.5″
water	plate D38	3.5″

Table 3. Set of Six Textures.

Two classification programs were used, a K nearest-neighbor classifier and a Fisher linear classifier. The K nearest-neighbor classifier classifies an unknown point by finding the K points of known classes that are closest to the unknown point in parameter space. The unknown point is given the class of the majority of these K points. The classifier in this report uses the Euclidean metric to determine distance (i.e. the square root of the squares of the differences between parameters). The Fisher classifier considers two classes at a time. It finds the best straight line to separate the points in these two classes in parameter space. Here, best means maximizing the ratio of between class scatter to within class scatter. Every point in the testing set is then classified into one of the two classes. This procedure is repeated for every combination of two classes. The final classification of an unknown point is the classification it most often received in the many two class problems.

4.1. Single Features

Each of the six texture images was divided into 64 windows of size 64 by 64, yielding 384 texture samples. The Fisher classifier was used with single features to classify each sample. Both the testing set and training set included all 384 samples. Results are given in Table 4. The same was done for the KNN classifier. Results for K=3 are seen in Table 5.

	0	1	2	3	4	5	6	7
0	55.7	43.2	42.4	47.1	64.1	63.5	38.5	21.6
1	51.8	47.4	31.3	46.1	53.4	44.8	50.5	42.2
2	56.3	55.7	46.1	39.3	50.5	38.0	31.0	43.5
3	58.3	44.0	31.5	21.1	51.8	37.8	47.9	35.7
4	57.6	57.0	38.0	38.3	50.0	51.3	35.4	48.7
5	65.9	54.7	51.0	17.4	35.7	49.7	50.0	39.3
6	34.1	41.9	56.3	43.2	54.2	56.8	45.1	40.1
7	54.9	48.7	37.0	44.3	50.5	51.6	25.8	20.6

Table 4. Fisher classification with single features. 384 samples. 6 classes. The entry in row r column c gives the %correct using the average separation from edges of r*45 degrees to edges of c*45 degrees.

	0	1	2	3	4	5	6	7
0	49.2	44.0	43.2	40.6	59.6	65.6	42.2	36.2
1	49.7	46.9	43.2	46.4	51.6	54.9	51.0	40.6
2	47.4	51.3	45.3	38.5	50.3	50.5	48.2	35.4
3	56.5	47.4	41.1	36.5	52.3	50.5	44.0	40.4
4	60.7	56.0	42.4	34.4	50.0	43.5	43.2	44.0
5	62.8	53.4	46.9	40.6	48.4	55.7	46.4	44.0
6	50.3	50.5	51.3	41.1	52.1	50.3	43.0	38.0
7	52.1	45.3	41.4	41.9	51.0	59.9	38.8	43.8

Table 5. KNN classification with single features. 384 samples. 6 classes. K=3. The entry in row r column c gives the percent correct using the average separation from edges of r*45 degrees to edges of c*45 degrees.

The features almost always result in an accuracy rate that is good for a classifier using only a single feature (see Section 5 for a comparison with other texture methods). The range for KNN classification is 34.4% correct to 65.6% correct, with a median of 46.1%. The range for Fisher classification is 17.4% correct to 65.9% correct, with a median of 46.9%.

The confusion matrix for the best KNN feature is given in Table 6. It is typical of the confusion matrices for the other good features. Since it is especially fine grained and uniform, the pressed cork texture is easily distinguished from the others. Pigskin and water are hard to classify since they are not uniform textures. The pigskin image is composed of wavy hair so the edge orientations are not evenly distributed throughout the plate. The water image is an oblique view of a lake so there are perspective effects.

	classified as					
true	cork	grass	pig	raffia	straw	water
cork	63	0	0	1	0	0
grass	0	38	10	9	0	7
pig	0	17	35	1	2	9
raffia	3	18	0	41	0	2
straw	0	0	5	0	57	2
water	1	26	13	6	0	18

Table 6. Confusion matrix for KNN classifier using the 0 degree to 225 degree edge separation feature. Percent correct:65.625

The performance of the Fisher classifier and the KNN classifier are similar; the ranges are comparable and the ranking of features is about the same. There are some trends among the results: (1) Within a row, the best feature usually is at or one away from the column giving the average separation between antiparallel edges. For example one of the best features is the 0-5 feature (row 0 col 5) which gives the average separation from edges of orientation 0 degrees to edges of orientation 5*45 degrees = 180+45 degrees. In the few rows that do not show this behavior the best feature is comparable in accuracy to the antiparallel feature. (2) Within a row, the second best feature is often at or one away from the column giving the average separation between parallel edges (the major diagonal of the matrix). There are more exceptions to this trend than to the first. (3) Features involving diagonal edges and/or edges with orientations perpendicular to each other are usually not among the best features. (4) The same trends hold for the columns as well as the rows.

4.1.1. Discussion

The trends listed above are pleasing to observe; they can be explained in terms of "objects" or primitives in the texture. Consider the first observation, that antiparallel edge separations are good features. Say a texture were made up of sparsely distributed discrete objects. If the edge finder described above worked optimally, each object would be bordered by a sequence of overlapping straight edges of the eight orientations. The average distance between pairs of antiparallel edges would give the length of various cross sections of the object. It would seem that these cross sections would be good features for measuring the texture, and the experimental data tends to confirm this (but note that the textures used are not made up of discrete objects). For example, consider a black and white checkerboard. The average edge separation from edges of orientation 0 degrees to edges of 180 degrees would give half the vertical width of the white squares (remember that in the above procedure each edge grows and meets another edge at about half their mutual separation). Similarly, the average edge separation from edges of orientation 90 degrees to edges of orientation 270 degrees would give half the horizontal width of the squares. For many textures, however, the constituent "objects" are not so obvious. For example, in a texture of dark bricks with white mortar, the 0 to 180 degree measurement would give the width of the mortar on the horizontal surfaces of the bricks, and the 90 degree to 270 degree measurement would give the width of the mortar on the vertical surface of the bricks. These two measurements would likely be the same, even though the length and thickness of the bricks are likely to be different.

Parallel edge separations give another set of features which are physically meaningful. Recall that these features were often the second best features in the above classifications. For the black and white checkerboard example, the 0 degree to 0 degree edge separation would give the vertical thickness of a black square on top of a white square; the 90 degree to 90 degree edge separation would give the horizontal thickness of a black square beside a white square. In this example, this gives no more information over the antiparallel separation features. In the brick example, the 0 degree to 0 degree edge separation would give the thickness of the mortar on the vertical face of the bricks. This is because the Voronoi-like region grower elongates as well as thickens edges. The growth of an edge is in all directions, generating an expanding region. So a horizontal edge in a brick texture is going to grow into a nearby horizontal edge on a horizontally neighboring brick. So again the parallel edge separation adds little new information over the antiparallel edge separation. It seems physically reasonable that this feature is usually second best.

It also seems reasonable that the separation of edges in perpendicular orientations is a poor feature. In the two example textures above, these features would not have any meaning if the edge finder worked well since such edges would meet at the right angles of the squares. If the edge finder occasionally missed corners, these features would just reflect the sloppiness of the edge finder, not anything physically meaningful about the texture.

The textures used in the experiment are not made up of discrete objects on a uniform background. None could easily be considered a collection of objects whose average cross section was being measured. For example, consider the pressed cork texture. The individual grains might be considered objects. But this will not work since the grains are so small, have many different shapes and sizes, overlap, and cast shadows. Individual grains cannot be found in either the edge map produced by the 3 by 3 template operator or in the final edge drawing. The edges found by the edge finder seem to be edges formed by various groupings of highlights and shadows in the image. The "objects" in this texture are the various groupings, not the individual grains. Similar comments can be made about the other textures.

4.2. Pairs of Features

Pairs of features were used to classify the same 384 texture samples. Again, the training set and testing set were the same. The antiparallel edge separation feature pairs were the best or close to the best. Table 7 shows the accuracy of the Fisher classifier using these pairs of features. The range is 45.8% correct to 78.6% correct. In Table 8, each feature selected is the best feature in one of the rows of Table 4, the single feature classification accuracy matrix. The classification accuracy has improved somewhat by using pairs instead of single features. The range is 46.4% correct to 82.6% correct. In both of these tables, the best classifications result when dissimilar features are paired. For example, in Table 8 the best pair is feature 0-5 and feature 6-2, two roughly antiparallel pairs which are perpendicular to each other.

	1-5	2-6	3-7	4-0	5-1	6-2	7-3
0-4	66.9	77.1	62.8	65.1	70.8	76.8	69.3
1-5		76.0	58.6	68.8	59.9	77.9	62.5
2-6			61.5	74.2	78.6	56.0	67.4
3-7				59.6	58.8	66.1	45.8
4-0					64.6	75.8	66.1
5-1						77.3	60.2
6-2							67.7

Table 7. Fisher classification with selected pairs of features. 384 samples. 6 classes. Each feature is an antiparallel edge separation. 1-5 means the average separation from edges of orientation 1*45 degrees to edges of orientation 5*45 degrees. The entry at row r and column c gives the percent correct using that pair of features.

	1-5	2-1	3-0	4-0	5-0	6-2	7-0
0-5	71.6	66.1	73.2	71.6	72.1	82.6	68.0
1-5		65.1	67.7	68.8	46.4	77.9	64.1
2-1			62.8	65.9	65.6	73.2	60.4
3-0				65.4	67.2	75.8	64.6
4-0					70.8	75.8	59.4
5-0						79.7	66.7
6-2							71.4

Table 8. Fisher classification with selected pairs of features. 384 samples. 6 classes. The selected features were the best feature in each row of Table 4. The entry at row r and column c gives the percent correct using that pair of features.

Table 9 gives the classification rates using pairs of antiparallel edge features with the KNN classifier. The classification rates are about the same. Again, the best rates seem to be obtained when dissimilar pairs of features are used.

	1-5	2-6	3-7	4-0	5-1	6-2	7-3
0-4	62.0	76.8	58.6	60.4	68.2	74.5	63.8
1-5		77.9	55.2	64.3	60.9	74.5	56.8
2-6			58.6	77.3	76.0	61.5	64.3
3-7				63.3	54.7	56.0	44.0
4-0					65.9	74.0	64.3
5-1						75.3	54.2
6-2							61.5

Table 9. KNN classification with selected pairs of features. 384 samples. 6 classes. K=3. Each feature is an average separation between antiparallel edges.

Table 10 gives the classification rate for the KNN classifier using selected triples of features. Each feature in this table is an antiparallel edge separation. The highest classification rate is 85.9% correct, achieved with a horizontal, a vertical, and a diagonal antiparallel edge separation feature. The lowest classification rate is 59.1% correct, achieved with 3-7, 5-1, and 7-3 features. Since the first and last of these features are similar, it is not surprising that these three features do not do well. Combinations using more features, or features other than the antiparallel edge separations have not been tried (however, see Section 5).

	2-6	3-7	4-0	5-1	6-2	7-3
0-4 + 1-5	83.6	64.6	64.6	71.4	78.6	69.5
0-4 + 2-6		78.6	81.8	85.9	81.0	80.7
0-4 + 3-7			65.1	69.5	78.4	65.6
0-4 + 4-0				69.3	80.2	68.5
0-4 + 5-1					82.3	71.9
0-4 + 6-2						78.4
1-5 + 2-6		78.4	82.0	79.2	79.7	77.9
1-5 + 3-7			66.9	61.5	75.0	57.3
1-5 + 4-0				68.8	80.2	67.2
1-5 + 5-1					76.8	61.2
1-5 + 6-2						76.6
2-6 + 3-7			82.6	78.4	62.5	64.1
2-6 + 4-0				82.8	79.7	80.7
2-6 + 5-1					79.4	78.1
2-6 + 6-2						65.9
3-7 + 4-0				69.0	72.7	65.9
3-7 + 5-1					75.3	59.1
3-7 + 6-2						62.5
4-0 + 5-1					84.1	66.1
4-0 + 6-2						74.2
5-1 + 6-2						74.5

Table 10. Classification rate with triples of features. Each feature is an antiparallel edge separation. 3-Nearest Neighbor classification. 384 samples.

5. Comparison with Other Texture Features

This section compares the edge separation texture classification method developed above with Laws' features and cooccurrence matrix features. These comparison studies will use the same 384 texture samples used in the above experiments.

5.1. Laws' Features

Laws found that the output of a set of very simple masks can be used as an effective texture measure [LAWS79a] [LAWS79b] [PIET82b]. The version of his method used in the comparison study here convolves the image with nine three by three masks. The average, standard deviation, skew, and kurtosis of the output of each mask is computed for each texture sample, giving a total of 36 features. Table 11 shows the percent correct classification using single features with the 384 texture samples and using the Fisher classifier. As before, all 384 samples were in the training as well as the testing set. Table 12 shows the same for the KNN classifier. The classification accuracy using the mean of the output of one of the masks is usually poor (often 16.7%, the pure chance rate for 6 classes). This is expected since all but one of the Laws operators are zero sum masks. Skew and kurtosis, the normalized third and fourth moments, are usually not very useful either. The best features are the standard deviations of the mask responses. This agrees with what Laws himself found. The standard deviation of these masks he calls "texture energy". Laws' features used one at a time do not perform as well as our edge separation features. The range for single Laws features using the Fisher classifier is 16.7% correct to 57.8% correct, with a median of 20.1% correct. For edge separation features, the corresponding range using the Fisher classifier is 17.4% correct to 65.9% correct with a median of 46.9% correct. The range for single Laws features using the KNN classifier is 16.7% correct to 52.9% correct with a median of 33.3% correct. For edge separation features, the corresponding range using the KNN classifier is 34.4% correct to 65.6% correct with a median of 46.1% correct.

	mean	sd	skew	kurt
l3l3	19.0	31.0	25.8	32.6
l3e3	39.6	48.7	39.3	47.1
l3s3	20.3	56.0	20.1	20.8
e3l3	45.8	44.0	33.1	41.1
e3e3	16.7	36.5	16.7	16.7
e3s3	16.7	20.1	16.7	16.7
s3l3	26.3	57.8	20.1	30.2
s3e3	16.7	18.5	16.7	16.7
s3s3	16.7	16.7	16.7	16.7

Table 11. Fisher classification with single Laws features. 384 texture samples. 6 classes.

	mean	sd	skew	kurt
l3l3	18.0	24.7	20.1	24.5
l3e3	37.2	49.0	42.2	39.6
l3s3	22.7	52.6	36.2	43.0
e3l3	39.1	43.8	33.3	37.5
e3e3	18.5	34.9	29.9	41.9
e3s3	19.0	40.1	22.9	28.4
s3l3	31.0	52.9	35.7	33.9
s3e3	16.7	49.5	29.2	27.1
s3s3	19.3	40.6	25.5	32.0

Table 12. KNN classification with single Laws features. K=3. 384 texture samples. 6 classes.

Table 13 shows the classification rate of the Fisher classifier using pairs of texture energy features. The range of accuracy is 18.8% correct to 81.3% correct. The range of accuracy for selected pairs of the best edge separation features is 46.4% correct to 82.6% correct. Again, the edge separation features appear to do better. This comparison may not be fair, since the eight edge separation features that were selected were the best out of 64 features, while the nine Laws features were the nine texture energy features, not necessarily the best out of the 36 Laws features.

	l3e3	l3s3	e3l3	e3e3	e3s3	s3l3	s3e3	s3s3
l3l3	53.6	67.4	47.4	56.8	28.1	60.2	28.1	30.5
l3e3		63.3	72.7	62.2	54.7	71.4	50.5	51.8
l3s3			78.4	65.4	58.6	81.3	55.2	56.3
e3l3				54.4	45.6	65.9	43.8	43.2
e3e3					38.0	63.8	27.6	28.9
e3s3						58.3	22.4	21.1
s3l3							58.1	57.8
s3e3								18.8

Table 13. Fisher classifier with pairs of Laws features. 384 samples. 6 classes. Each feature is a standard deviation of one of the 3x3 Laws operators.

5.2. Cooccurrence Matrix Features

Cooccurrence matrices are commonly used for texture discrimination and are often used in comparison studies [WESZ76] [LAWS79a]. A single matrix is built for a particular angle and displacement. Typically, the angles are horizontal, vertical, and the two diagonals and the displacements are a few pixels. To cut down the size of the matrices, the gray levels of the image are often grouped into a small number of ranges. A particular matrix records the frequency that pixels of each range of gray level can be found at the given angle and displacement to each other. Matrices may be symmetric or asymmetric depending on whether the orientation between two pixels is regarded as undirected or directed (180 degrees difference depending on which pixel is chosen). For the matrices used here, gray levels were put into sixteen ranges and the matrices were symmetric. There are very many cooccurrence matrix features. The four chosen here are commonly used, and were used in [WESZ76].

As before, 384 texture samples were used. The training set was the same as the testing set. Table 14 gives the result when single cooccurrence features were used to classify the textures using the Fisher classifier. The range was 20.1% correct to 48.7% correct with a median of 40.6% correct. The distance one features are the best, a result also found in [WESZ76]. Overall, the edge separation features appear to perform better than the cooccurrence features.

dir	distance = 1			
	CON	ASM	ENT	COR
0	48.4	47.7	46.4	48.7
1	38.5	36.2	34.7	38.0
2	45.1	41.1	39.8	44.5
3	40.1	41.9	38.5	40.4

dir	distance = 2			
	CON	ASM	ENT	COR
0	46.9	45.3	41.7	47.9
1	40.4	39.3	40.4	40.6
2	39.8	40.4	38.3	39.6
3	40.6	43.0	44.8	41.4

dir	distance = 4			
	CON	ASM	ENT	COR
0	44.0	41.1	41.4	44.3
1	44.0	30.2	29.2	44.0
2	41.1	34.6	38.0	40.9
3	37.5	20.1	26.6	37.8

Table 14. Fisher classification using single cooccurrence matrix features. 6 classes. 384 samples. Matrices were built using four displacements in four directions.

Pairs of features from the displacement one matrices were also tested using the Fisher classifier. The range was 34.6% correct to 72.9% correct. with a median of 46.4. With pairs of features from the displacement two matrices the range was 46.9% correct to 72.7% correct. with a median of 70.3. With pairs of features from the displacement four matrices the range was 44.0% correct to 70.6% correct. with a median of 65.4. Again. the edge separation features give better results.

6. Average Edge Orientation Features

The three by three template operator discussed in Section 2 produces an edge map for each texture. The edge pixels in these edge maps have attributes of magnitude and orientation. The average edge pixel orientation could also be used as a texture feature. Recall that orientation is one of the properties of the "elongated blob" texton of Julesz. However, for each collection of edge pixels of a particular orientation there will be another collection of edge pixels of the antiparallel orientation. Since these corresponding collections will have approximately the same numbers of pixels in them, the average orientation will be meaningless for most textures. To avoid this problem, parallel and antiparallel edge pixels are given the same orientation in the range 0 to 135 degrees (in 45 degree increments) by taking the orientation modulo 180 degrees. The average and higher moments are now calculated. Classification results using these features and the Fisher classifier are shown in Table 15.

avg orientation	70.6%
standard deviation	52.6%
skew	68.2%
kurtosis	52.9%

Table 15. Edge orientation features. Fisher classifier. 384 samples.

These classification rates are very good. The accuracy rate for average edge orientation is the best found for these textures for any single features in our study. No spatial distribution information is contained in these features. This suggests that combining them with some of the average edge separation features should yield good results. Table 16 shows the classification accuracy for pairs of average edge orientation and edge separation features using the Fisher classifier.

	0	1	2	3	4	5	6	7
0	68.0	71.9	77.1	79.7	79.2	84.4	74.2	70.1
1	73.4	70.3	74.0	77.3	74.0	77.6	75.8	71.1
2	71.2	73.2	71.4	72.9	72.9	74.7	78.1	73.2
3	73.4	74.2	71.9	72.7	70.3	76.3	75.0	73.7
4	76.8	76.8	74.0	70.1	66.9	72.1	76.0	74.2
5	81.5	77.6	75.8	76.3	71.6	70.6	74.5	76.6
6	69.0	74.0	76.8	72.9	76.8	77.1	75.3	74.7
7	71.4	76.6	71.9	76.6	74.7	79.2	72.7	71.9

Table 16. Fisher classification with pairs of features. One feature is always average edge orientation, the other is the separation between extended edges of orientation indicated by the row and column numbers. 385 samples, 6 classes.

Pairing edge orientation with edge separation features gives consistently good results, but the improvement over single feature classification rates is often small. For example, edge separation feature 0-5 gave an accuracy of 65.6% when used alone. Average edge orientation gave an accuracy of 70.6% when used alone. When both were used in the classifier, these two features gave an accuracy of 84.4%. This seems like a small improvement for features that perform so well alone.

7. An Alternate Method for Computing Line Separation Features

This section discuses an alternative method for generating edge separation features. This method retains the main ideas of the method described in Sections 2 and 3, but the approach is different in several aspects. It is discussed here to demonstrate that the edge separation features can withstand wide variation in the method by which they are generated. The principle differences between the previous method and this alternate method are:

(1) Only four orientations are used, in 90 degree steps.

(2) An edge magnitude map for each orientation is produced directly from the original gray level image by convolution with a long, thin edge template of size five by two. Because of this, a pixel can have a nonzero magnitude in several edge maps.

(3) No smoothing is done except for that implicit in the convolution masks.

(4) Thinning the edges is done as before except now edges are bordered by negative magnitudes as well as by zeros.

(5) Growth of edges is done by propagating labels to 8-neighbors rather than 4-neighbors (for a discussion of the effect of various metrics on discrete Voronoi diagrams, see [PHIL83]).

(6) Since there are only four orientations, the feature matrix is four by four.

Table 17 shows the classification accuracy when these single features were used in the Fisher classifier with the same 384 samples that have been used before. The range of accuracy is 24.5% correct to 50.5% correct, with a median of 33.3% correct. Recall that with features generated by the previous method the range was 17.4% correct to 65.9% correct, with a median of 46.9% correct. It appears that features generated by the alternate method are not as good as those generated by the first method.

	0	1	2	3
0	30.7	40.9	36.7	39.1
1	28.9	34.1	33.3	50.5
2	41.7	33.3	24.5	44.5
3	27.9	50.5	28.6	31.5

Table 17. Fisher classification with single line separation features (alternate method). The entry in row r column c gives the % correct for separation from lines of r*90 to lines of c*90.

Table 18 shows the percent classification accuracy when selected pairs of features are used. Each feature in this table is either a separation between edges of antiparallel orientation, or a separation between edges of parallel orientation. The range is 36.7% correct to 74.2% correct, with a median of 52.1% correct. Again, this is somewhat worse than the rates using pairs of features generated by the first method, given in Table 8. However, classification accuracy using pairs of alternate features is closer to that using pairs of the previous method than the single features were. Note that the best pairs combine one of the two separations for horizontal antiparallel edges with one of the two separations for antiparallel vertical edges (eg. the 1-3, 2-0 pair is the best). This trend was also observed with features generated by the previous method.

These features also combined well with the edge orientation features. When the (1-3, 2-0) pair was used with average edge pixel angle in a 3NN classifier, the accuracy was 94.2% correct. Oddly, no two of the features generated by the previous method when combined with any average edge orientation feature give such a high classification rate.

	0-2	1-1	1-3	2-0	2-2	3-1	3-3
0-0	38.0	45.3	59.4	41.7	36.7	52.1	44.5
0-2		54.9	69.5	44.0	43.8	63.0	51.0
1-1			57.6	58.9	40.4	53.1	42.7
1-3				74.2	53.6	58.3	55.7
2-0					46.1	66.4	56.5
2-2						47.1	37.0
3-1							54.4

Table 18. Fisher classification with pairs of features. 384 samples. 6 classes. x-y means the separation between edges of x*90 degrees to edges of y*90 degrees.

Although not as good as the previous features, the features calculated by this alternate method are still good features when compared to cooccurrence features. This suggests that the general idea of using edge separation features is a good one, since it works reasonably well with edges generated by two different methods. The poorer performance of the alternate version may have been due to the fact that it used edges in only four orientations, where the first method used edges in eight orientations.

8. Concluding Remarks

The previous sections have shown that features of edge segments found in a texture are good for texture discrimination. This section discusses how this technique is related to generalized cooccurrence matrix features and ways in which our method could be extended.

8.1. Relationship to Generalized Cooccurrence Matrices

Davis and colleagues have proposed generalized cooccurrence matrices, GCM's, to describe structural feature distribution in a texture [DAVI79][DAVI81]. Although their method may be used with any image feature that can be assigned locations and properties, in the following it will be explained as it applies to extended edges (in their experimental studies, the only image features used were gray level, edge pixels, and extended edges). Suppose that a texture has been convolved with a set of edge operators and similar edge pixels have been linked together so that the resulting image is a distribution of extended edges of several different orientations. The different orientations may be used to label the rows and columns of a matrix, as seen below:

	0	45	90	135
0				
45				
90				
135				

A cell of this matrix is incremented whenever a pair of extended edges is found that have the orientations given by the row and column of the cell, and also satisfy a particular spatial predicate. One of the spatial predicates used by Davis is Ak(f1,f2), where Ak(f1,f2) is true if either end of the extended-edge f1 lies in either of the k by k square neighborhoods centered about either end of extended edge f2. After a GCM has been computed, various descriptors can be computed from it to describe the texture by means similar to those used for gray level cooccurrence matrices.

The reported classification results using GCM's with extended edges were worse than for gray level cooccurrence matrices [DAVI81]. This is surprising, in light of the success with the edge features reported here. Classification results when GCM's were used with edge pixels were only slightly better than those for gray level cooccurrence matrices.

Although appealing, GCM's have several problems describing texture. Each feature is expected to have a set of coordinates that describe its location in the image. It is difficult to choose such a set for many types of image features. For example, the spatial predicate Ak(f1,f2) described above depends on knowing the exact endings of extended edges. This is difficult, for reasons discussed in Section 1. Another spatial predicate used by Davis involved knowing the center of an extended edge. Again, this is difficult to determine precisely. To overcome this problem, single pixel image features might be used so that their locations are easily determined. An example of this is the edge pixels that performed better than the extended edges [DAVI81]. Unfortunately, now the method is no longer describing higher level features. In contrast, the edge separation features described in this report do not depend critically on describing an edge by a few location coordinates and do not assume that the edge data are noise-free.

The meaning of a GCM is not very clear when a nonintuitive spatial predicate is used. To make matters worse, the features used with the classifiers are composite descriptors calculated from the GCM (e.g. contrast, entropy, and correlation). These descriptors are even harder to understand physically. With edge separation features, the individual cell values were used for classification with good results. Although they suggest using them, Davis and colleagues do not report any results using individual cell values of GCM's as features for a classifier. It might be that some cells would have been better features than the descriptors calculated from the matrix as a whole.

The edge separation method developed in this report also suffers from generating too many features. Preliminary experiments were performed to try to find composite descriptors of our edge separation matrix similar to those used for GCM's. However, none of the standard cooccurrence descriptors performed nearly as well as the single edge separation features.

8.2. Possible Extensions

There are two main ways in which the edge separation method presented here could be extended. The first way is to consider alternative spatial properties for describing inter-edge relations (or other non-local features). For example, in Section 3 an edge grew when each pixel along the perimeter passed its labels on to any 4-neighbor that did not already have them. By restricting which neighbors can receive new labels from a pixel, edges can be forced to grow in only selected directions. For example, growth could be restricted to the horizontal direction only. The edge separation features would then be a measure of the horizontal distances between edges of all orientations. The same could be done for vertical and diagonal growth. The resulting set of features would then describe the average distance and direction between edges of all the orientations. Other changes in the way a region grows would yield other features. The direction in which an edge grows could be a function of its orientation (so that horizontal edges would only grow vertically and vertical edges grow only horizontally).

Alternative growth patterns also could be used to calculate GCM-like features. Consider growing edges in the horizontal direction only. Allow the process to continue for a fixed number of generations. When the growth is stopped, a GCM could be calculated by scanning through the image, noticing which labels occur at the same pixel. The attributes of the edges which cooccur in this manner could be used to determine the row and column of the GCM. This method would not have to be restricted to edges. As long as an image feature can be described as a connected region a growth process can be used to define a spatial predicate. This has the advantage that distinguished points (such as endpoints and midpoints) do not have to be detected for the method to work. Clearly, these GCM features and the edge separation features could be calculated simultaneously. The full set of resulting features might be very effective for texture discrimination.

A second way to extend our method is to define new spatial statistics of edge representations. For example, in addition to calculating the average edge separation between edges at two particular orientations, higher moments could be calculated. These higher moments are not very appealing, however, because their relation to the original image is not very clear. To deal with the large number of features generated by these methods, other composite descriptors of the edge separation matrix are needed. As discussed above, cooccurrence matrix descriptors perform poorly. But there might be more meaningful spatial statistics that could be calculated.

Another motivation for calculating descriptors of the matrix is to allow texture analysis to be insensitive to rotations of the texture. Rotating the texture by 45 degrees has the effect of rotating the row and column numbers of the matrix of edge separation distances. So, e.g., the distance at row r column c is now at row (r+1 mod N) column (c+1 mod N), where N is the number of directions. Rotation insensitive matrix descriptors that utilize this characteristic are described in [DAVI81].

Features which are insensitive to texture scale (probably within a limited range) are also important. Line separation features are geometric distances and change linearly with changes of scale. The ratio of the distances describing a feature will remain constant as the scale changes, so the ratio of edge separation features should also remain constant. A possible limitation is that the edge finder might not produce scaled versions of the edges at widely different scales. Another limitation is that non-Euclidean metrics are not orientation insensitive when computing inter-edge distances [PHIL83]. Neither the 4-neighbor metric used in Section 3 nor the 8-neighbor metric used in Section 7 yield even growth in all directions. It is possible that the Euclidean metric will have to be used to obtain rotation insensitive features. In spite of these difficulties, edge separation features depend on scale in a way that is clearer than for any other texture measure.

REFERENCES

[BECK80] BECK, J. *Texture segregation.* Technical Report TR-971, Computer Vision Laboratory, University of Maryland, College Park, MD, 1980.

[BECK81] BECK, J. *A theory of textural segmentation.* Technical Report TR-1083, Computer Vision Laboratory, University of Maryland, College Park, MD, 1981.

[BROD66] BRODATZ, P. *Textures: a Photographics Album for Artists and Designers.* Dover Pub., New York, 1966.

[DAVI79] DAVIS, L. S., JOHNS J. A., and AGGARWAL, J. K. "Texture analysis using generalized co-occurrence matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-1,** 3 (July 1979), 251-259.

[DAVI81] DAVIS, L. S., CLEARMAN, M., and AGGARWAL, J. K. "An empirical evaluation of generalized cooccurrence matrices," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-3,** 2 (March 1981), 214-221.

[EHRI78] EHRICH, R. W., and FOITH, J. P. "A view of texture topology and texture description," *Computer Graphics and Image Processing* **8** (1978), 174-202.

[HARA73] HARALICK, R. M., SHANMUGAM, K., and DINSTEIN, I. "Textural features for image classification," *IEEE Transactions of Systems, Man, and Cybernetics* **SMC-3,** 6 (Nov. 1973), 610-621.

[HARA79] HARALICK, R. M. "Statistical and structural approaches to texture," *Proceedings IEEE* **67,** 5 (May 1979), 786-804.

[HONG80] HONG, T., DYER, C. R., and ROSENFELD, A. "Texture primitive extraction using and edge-based approach," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-10,** 10 (Oct. 1980), 659-675.

[JULE81] JULESZ, B. "A theory of preattentive texture discrimination based on first-order statistics of textons," *Biological Cybernetics* **41,** (1981), 131-138.

[JULE83] JULESZ, B., and BERGEN, J. R. "Textons, the fundamental elements in preattentive vision and perception of textures," *Bell System Technical Journal* **62,** 6 (July/Aug. 1983), 1619-1645.

[LAWS79a] LAWS, K. I. *Textured image segmentation.* Ph. D. dissertation. University of Southern California, 1979.

[LAWS79b] LAWS, K. I. "Texture energy measures," in *Proc. Image Understanding Workshop,* Los Angeles, Nov. 1979, 47-51.

[LEE81] LEE, D. T., and DRYSDALE III, R. L. "Generalization of Voronoi diagrams in the plane," *SIAM Journal of Computing* **10,** 1 (Feb. 1981), 73-87.

[NEVA79] NEVATIA, R., PRICE, K., and VILNROTTER, F. "Describing natural textures," in *Proc. 6th International Joint Conference on Artificial Intelligence*, 1979, 642-644.

[PHIL83] PHILLIPS, T., MATSUYAMA, T., *The labeled discrete voronoi diagram.* Technical Report TR-CS-1278, Center for Automation Reserch, University of Maryland, College Park, MD. May 1983.

[PIET82a] PIETIKAINEN, M., and ROSENFELD, A. "Edge-based texture measures," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-12,** 4, (July/August 1982), 585-594.

[PIET82b] PIETIKAINEN, M., ROSENFELD, A., and DAVIS L. S. "Texture classification using averages of local pattern matches," in *Proc. 6th International Conference on Pattern Recognition*, Munich, Germany 1982, 301-303.

[ROSE82] ROSENFELD, A. and KAK, A. C. *Digital Picture Processing*, Academic Press, New York, 1982.

[TAMU78] TAMURA, H., MORI, S., and YAMAWAKI, T. "Textural features corresponding to visual perception," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-8,** 6 (June 1978), 460-473.

[TOMI82] TOMITA, F., SHIRAI, Y., and TSUJI, S. "Description of textures by a structural analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-4,** 2 (March 1982), 183-191.

[WANG81] WANG, S., DIAS VELASCO, F. R., WU, A. Y., and ROSENFELD, A. "Relative effectiveness of selected texture primitive statistics for texture discrimination," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-11,** 5 (May 1981), 360-370.

[WESZ76] WESZKA, J. S., DYER, C. R., and ROSENFELD, A. "A comparative study of texture measures for terrain classification," *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-6,** 4 (April 1976), 269-285.

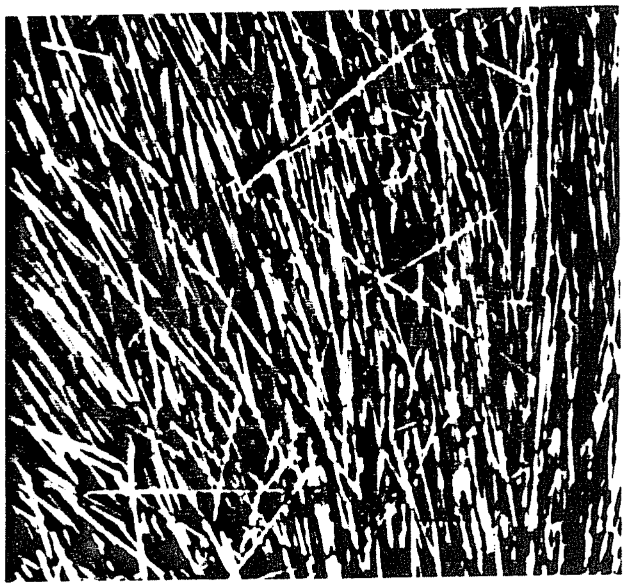


Fig. 1. Histogram flattened straw texture, 512 x 512 pixels.

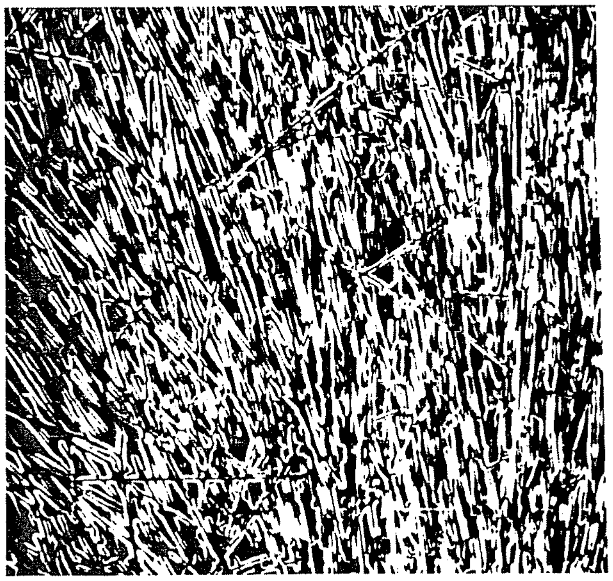


Fig. 2. Magnitude portion of edge pixel map of straw texture.

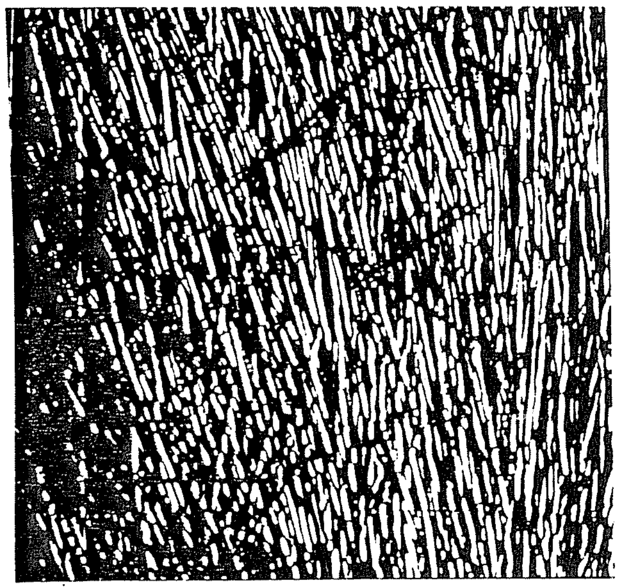


Fig. 3. Magnitude map of 90 degree edge pixels.

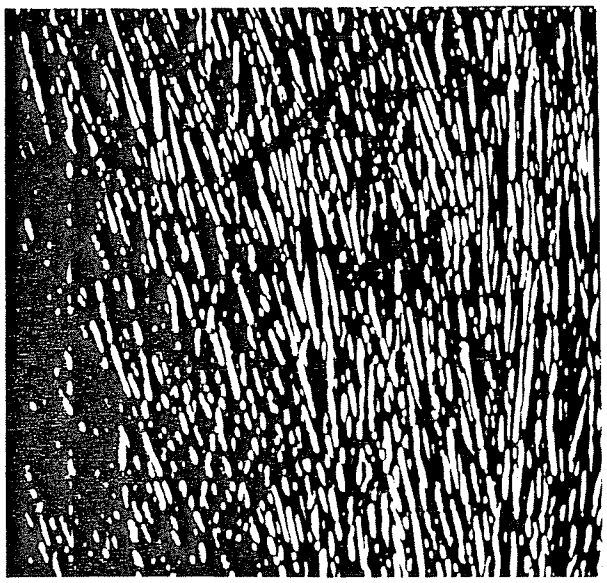
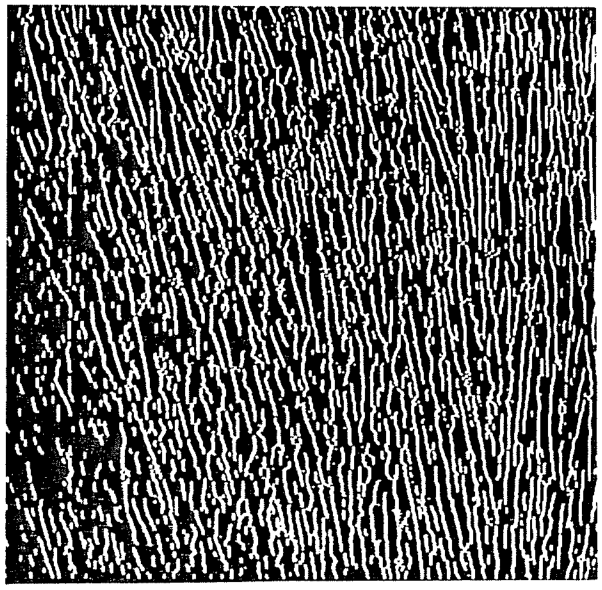
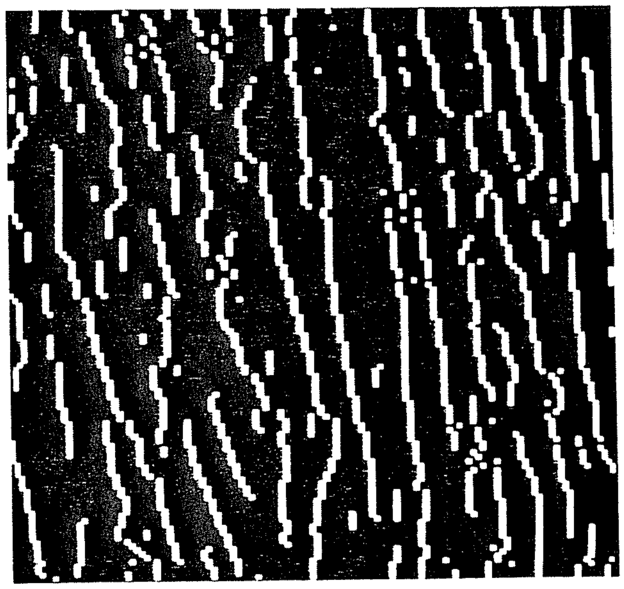


Fig. 4. Smoothed magnitude map.

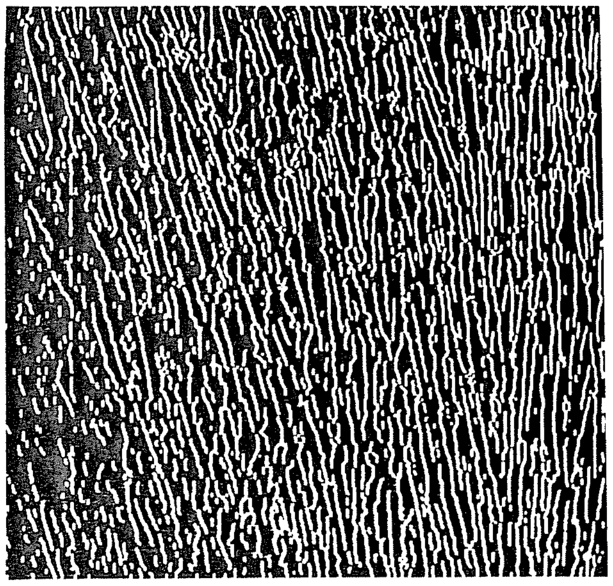


(a)

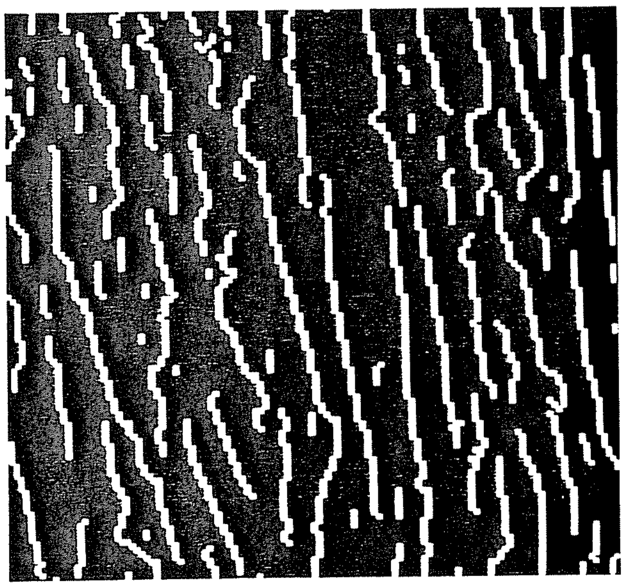


(b)

Fig. 5. (a) Thinned 90 degree edges, 512 x 512 pixels.
(b) Window of above, 64 x 64 pixels.



(a)



(b)

Fig. 6. (a) Thinned 90 degree edges after gap filling, 512 x 512 pixels.
(b) Window of above, 64 x 64 pixels.

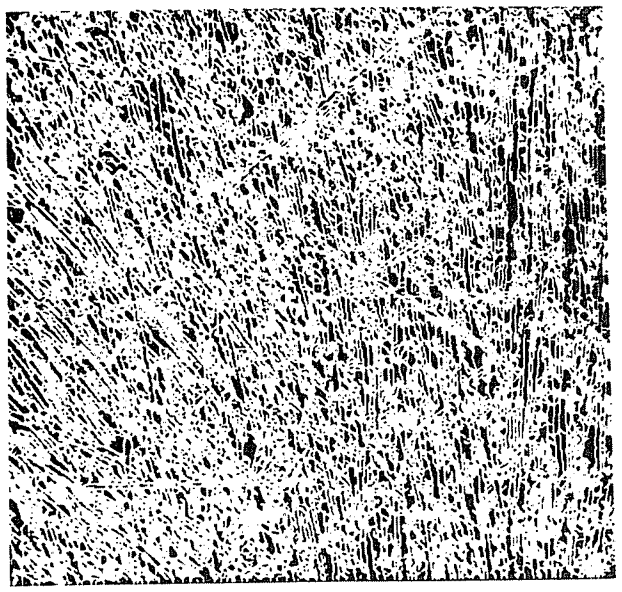


Fig. 7. Final edge segments of straw texture.