USC–SIPI REPORT #114

Textured Image Segmentation Using Feature Smoothing and Proabapilistic Relaxation Techniques

by

John Yu-Hsiung Hsiao

December 1987

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

Acknowledgments

I am indebted to the chairman of my dissertation committee, Prof. Alexander A. Sawchuk, for his advice and guidance throughout my study at USC. Without his effort, this research would have been far more trying. I am also grateful to Prof. Rama Chellappa for many helpful discussions and suggestions. Prof. Larry Goldstein from the Department of Mathematics, who showed his interest in this work and served as a member of the dissertation committee, is acknowledged.

Thanks go as well to all members of the Signal and Image Processing Institute who helped make my graduate studies a memorable and enjoyable experience. I am especially grateful to Mr. Allan G. Weber for his help in computer support and Mr. Ray R. Schmidt for his help in preparation of pictures shown in this work.

I would like to acknowledge the financial support of Hughes Aircraft Company, whose Doctoral Fellowship made this work possible.

Finally, I would also like to thank my wife, Chin-Min, for her love, patience and support during the course of this research.

Contents

List of Figures

List of Tables

Abstract

The segmentation of textured imagery into homogeneous regions is an important and difficult task in scene analysis. A satisfactory segmentation result should possess not only good region interiors but also accurate boundaries. In response to these requirements, the main objective of this research is twofold. First, to improve textured image segmentation results; especially along the borders of regions. Second, to take into account the spatial relationship among pixels to improve the segmentation of region interiors.

An improved method to extract textured energy features in the feature extraction stage is proposed. The proposed method is based upon an adaptive noise smoothing concept which takes the nonstationary nature of the problem into account. Texture energy features are first estimated using a window of small size to reduce the possibility of mixing statistics along region borders. The estimated texture energy feature values are then smoothed by a quadrant method which reduces the variability of the estimates while retaining the region border accuracy.

For a supervised segmentation system, the estimated feature values of each pixel are used by the Bayes classifier to make an initial probabilistic labeling. The spatial

constraints are then enforced through the use of a probabilistic relaxation algorithm. Two probabilistic relaxation algorithms have been investigated and their results are compared by computer simulation.

For an unsupervised segmentation system, the estimated feature values of a set of subsampled pixels are used in a K-means clustering algorithm to estimate the class statistics. The estimated class statistics are then used by the Bayes classifier to make an initial probabilistic labeling. One of the selected relaxation algorithms is then applied to enforce the spatial constraints.

Limiting the probability labels by probability threshold is proposed to make the relaxation iteration more efficient. The trade-off between efficiency and degradation of performance is studied. Finally, an overview of the proposed textured image segmentation system is provided and comparisons of overall performance are made.

Chapter 1

Introduction

An important task of an image analysis system is to segment the given image into meaningful regions and to label the individual regions. Segmentation of the image into regions can be done in many ways [1], [2]: from simple pixel classification methods using gray levels and local feature values to sophisticated split-and-merge techniques using statistical homogeneity tests. These segmented and labeled regions can then be used for representation, description, and recognition.

There exist two major approaches to image segmentation: edge-based and region-based. In edge-based methods, the local discontinuities are detected first and then are connected to form longer boundaries. In region-based methods, areas of the image with homogeneous properties are found, which in turn give the boundaries. The presence of texture causes difficulty for both the edge- and region-based methods. Essentially, our hypothesis that the objects consist of relatively homogeneous surfaces is violated, and very few intensity changes now correspond to object boundaries. In edge-based methods, we will get many edges due to texture that must be differentiated from object boundaries; in region-based methods, we will get many small regions that correspond

to texture. Therefore, textured image segmentation is an important and difficult task.

1.1 Research Objectives

There are two main objectives of this research. First, to improve textured image segmentation results; especially along the borders of regions. Second, to take into account the spatial relationship of pixels in the process of solving textured image segmentation problems.

This objective is motivated by the lack of effective algorithms for textured image segmentation. By "effective" we mean that the segmentation should possess the properties pointed out by Haralick [3] (see Chapter 2 for those properties). In order to achieve this objective, we think the following ingredients are essential for a textured image segmentation system:

- A set of texture features having good discriminating power.
- A segmentation algorithm having spatial constraints.
- Estimation of texture features taking the nonstationary nature of the feature image planes into account. By doing this, the segmentation results along the borders of the regions might be improved.

1.2 Organization of the Dissertation

The organization of the dissertation is shown in Figure 1.1. In chapter 2, definitions of texture and segmentation are stated; properties of a good image segmentation are pointed out; surveys of texture features and segmentation techniques are provided; past

work on textured image segmentation is tabulated; and an overview of the proposed textured image segmentation system is presented.

In chapter 3, an improved method to extract texture energy features is discussed. Laws' textured image segmentation system is first reviewed, and one shortcoming of Laws' method is then pointed out. In order to overcome this shortcoming, an edge preserving noise smoothing algorithm is introduced and its close relationship to feature extraction is also discussed. The proposed feature extraction algorithm is presented in detail. A Bayes classifier is then used to classify the pixels based upon their feature values. A comparison of the classification accuracy between the original and proposed feature extraction methods is presented.

The incorporation of spatial constraints into the segmentation algorithm is discussed in chapter 4. The weakness of classifying pixels based solely upon feature space distribution is pointed out. A probabilistic relaxation method used to enforce the spatial constraints is then introduced. Past research work on multispectral pixel classification using probabilistic relaxation approach is reviewed. Two probabilistic relaxation algorithms are proposed as the means to reduce the local ambiguities. To the best of the author's knowledge, no previous attempts have been made in using probabilistic relaxation to solve the textured image segmentation problem. Simulation results using the proposed algorithms are presented and discussed.

In chapter 5, a procedure for segmenting textured images without the need for training prototypes is described. The interrelationships among the feature data is first analyzed by a K-means clustering algorithm; the estimated class statistics from the clustering algorithm is then used by the Bayes classifier to classify pixels. Issues

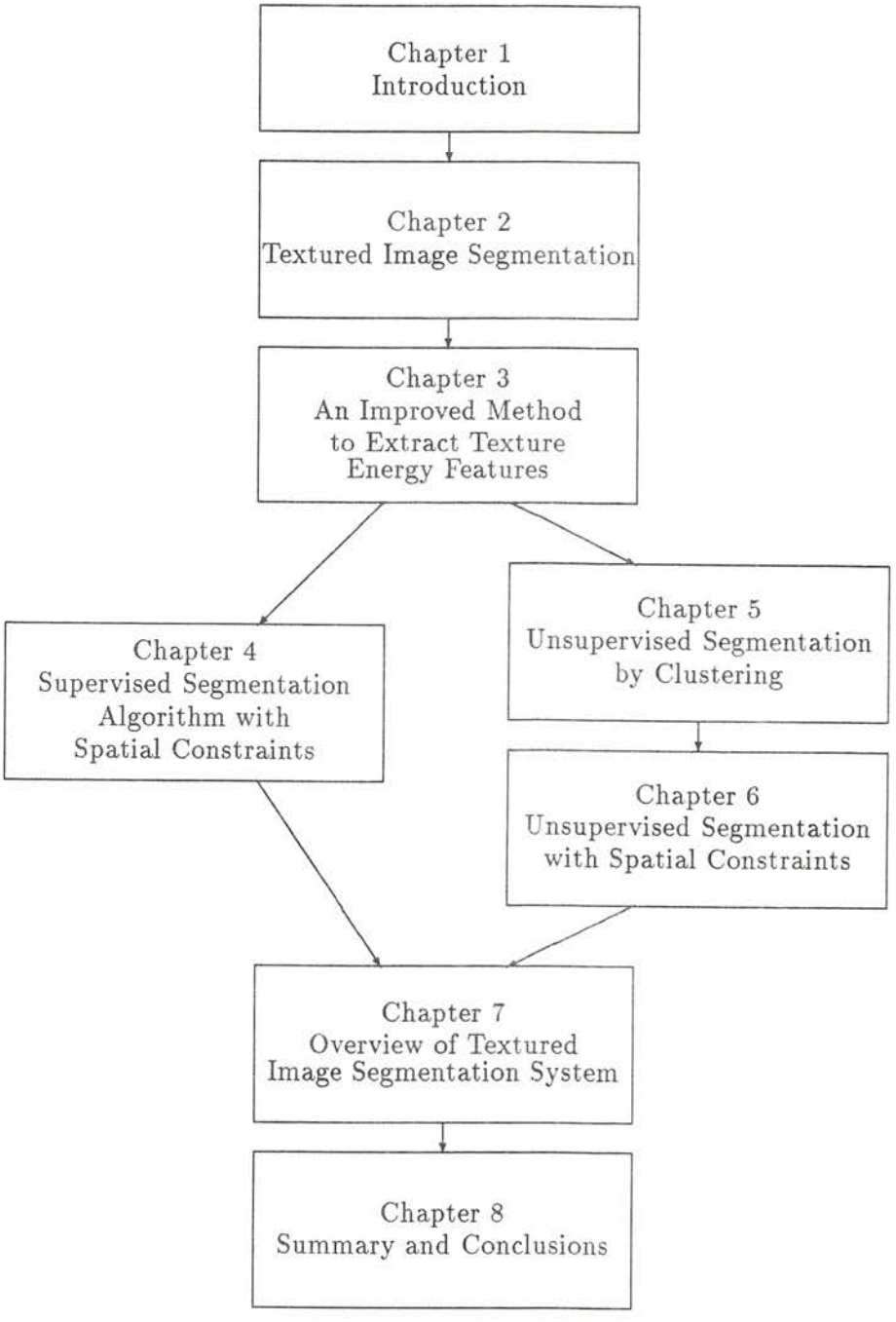


Figure 1.1: Organization of the Dissertation.

related to clustering algorithm are discussed. The proposed algorithm for clustering and cluster statistics estimation is presented in detail. Simulation results are given at the end of the chapter.

The incorporation of spatial constraints into the unsupervised segmentation algorithm is discussed in chapter 6. In addition, the effect of limiting the range of possible patterns for classification to a smaller subset of classes is studied. Proposed algorithms are described in detail. Finally, the simulation results are presented and discussed.

Chapter 7 provides an overview of the proposed textured image segmentation systems. In previous chapters, issues related to different stages of a textured image segmentation are discussed and algorithms are proposed for each of the stages. Chapter 7 presents an integrated description of the proposed supervised and unsupervised textured image segmentation systems in block diagram form. Some remaining issues are discussed at the end.

Chapter 8 concludes the dissertation. A brief summary and discussion of future research on the problem are given.

1.3 Contributions of this Research

The author's original contributions of this research can be summarized as:

- The use of quadrant window method for texture feature smoothing.
- The use of probabilistic relaxation to enforce spatial constraints for textured image segmentation problems.

- The use of texture energy features, clustering and probabilistic relaxation method to solve the unsupervised textured image segmentation problems.

- The use of probability thresholding to speed up the relaxation iterations.

Chapter 2

Textured Image Segmentation

Segmenting a picture into appropriate subregions is one of the most important stages in understanding the picture. These regions are generally areas homogeneous with respect to a given property or satisfying a given predicate. Textured image segmentation uses texture as the primary property to segment images. There are two issues that need to be addressed in a textured image segmentation problem. One is finding effective texture features that can distinguish one texture from another. The other is how to use these features to segment the picture into subregions.

Texture can be defined as [4] a structure composed of a large number of smaller similar sub-elements or patterns. Although each sub-element or pattern alone does not give the impression of texture, the collection of a large number of them does. Texture is a higher order image property, and depends on the statistics of pixels in a local neighborhood. In order to exploit textural differences, it is necessary to define and extract features that discriminate between textures. Section 2.1 provides a review of commonly used texture features.

A segmentation is defined as a partition of X, which is a set, into disjoint

nonempty subsets $X_1, X_2, \ldots, X_N$, such that

1. $\bigcup_i X_i = X$.

2. $X_i, \quad i = 1, 2, \ldots, N$, is connected.

3. $P(X_i) = \text{TRUE} \qquad \text{for} \quad i = 1, 2, \ldots, N$, where P is a logical predicate defined on a set of contiguous picture points.

4. $P(X_i \bigcup X_j) = \text{FALSE} \qquad \text{for} \quad i \neq j$, where X_i and X_j are adjacent.

Zucker [5] summarized the above conditions as follows : the first condition implies that every picture point must be in a region. This means that the segmentation algorithm should not terminate until every point is processed. The second condition implies that region must be connected, i. e. composed of contiguous lattice points. The third condition determines what kind of properties the segmented region should have. The fourth condition expresses the maximality of each region in the segmentation. Section 2.2 provides a survey of commonly used segmentation techniques.

What should a good image segmentation be? As Haralick [3] pointed out, a good image segmentation should possess the following properties:

1. Regions of an image segmentation should be uniform and homogeneous with respect to some characteristic such as gray level or texture.

2. Adjacent regions of segmentation should have significantly different values with respect to the characteristic on which they are uniform.

3. Region interiors should be simple and without many small holes.

4. Boundaries of each segment should be simple, not ragged, and must be spatially accurate.

To obtain property (1) and (2), proper selection of the features used in the classification algorithm is essential. Spatial constraints are generally needed to improve the region interiors as desired in the third property. The fourth property is a very important one and no completely satisfactory results exist.

2.1 Survey of Texture Features

Texture analysis may be performed using a statistical or structural approach. The statistical approach derives a set of local measurements for a given image over the space domain or the corresponding frequency domain. Features based on these measurements are then used for discrimination between textures. The structural approach assumes that a set of primitive spatial unit patterns can be easily identified. It then defines the texture as a combination of such primitives. The two methods can be combined in a hierarchical way, where the statistical procedure supplies the raw data for the structural procedure. We will restrict ourselves to the discussion of the statistical approach only.

2.1.1 Autocorrelation

The autocorrelation function may be used to characterize textures. If the primitives are relatively large, then the autocorrelation function will drop off slowly with distance. If the primitives are small, then the autocorrelation function will drop quickly with distance. If the primitives are spatially periodic, the autocorrelation function will drop off and rise again in a periodic manner. Thus texture coarseness is reflected

in the autocorrelation changes. Directionality of the pattern can be detected by the directional dependence of these functions.

Kaizer [6] used the distance at which the autocorrelation falls to $1/e$ of its peak as a measure for texture coarseness. The autocorrelation function measure and the texture energy measure in Fourier domain are closely related because the autocorrelation function and the power spectrum are the transforms of each other.

2.1.2 Relative Extrema Measures

Rosenfeld and Troy [7] and Rosenfeld and Thurston [8] conceive of texture not in terms of spatial frequency but in terms of the number of edges per unit area. To detect microedges, small neighborhoods can be used. To detect macroedges, large neighborhoods can be used. The local edge detector which Rosenfeld and Thurston suggested was the quick Roberts gradient (the sum of the absolute value of the differences between diagonally opposite neighboring pixels). Thus a measure of texture for any subimage can be obtained by computing the Roberts gradient image for the subimage and from it determining the average edge density in the subimage. This measure can then be used to classify textures.

Carlton and Mitchell [9] used a texture measure that counted the number of local extrema in a window centered at each pixel. Using three thresholds called "low", "medium" and "high", they produced three intermediate "gray level" pictures whose values are the number of local extrema (averaged over a window) produced by that threshold. These averaged number of local extrema can then be used to characterize textures.

2.1.3 First Order Statistical Measures

The simplest measures are based on first-order statistics - that is, the probability of single-pixel attributes. Some examples are mean and variance of intensity. More sophisticated first-order measures are based on histograms of the individual pixel attributes. These measures, even though they do not specifically use the spatial distribution of the pixel attributes, are still useful for many naturally occurring textures. Davis and Mitiche [10] used first order statistical measures in their model-driven iterative texture segmentation algorithm.

2.1.4 Spatial Gray-level Dependence: Co-occurrence

Spatial gray-level dependence (SGLD) matrices are one of the most popular sources of features. The SGLD approach computes an intermediate matrix of measures from the digitized image data, and then define features as functions on this discrete intermediate matrix. Given an image $f(i,j)$ with a set of discrete gray level I, we define for each of a set of discrete values of d and θ the intermediate matrix $S(p,q \mid d,\theta)$ as follows:

> $S(p,q \mid d,\theta)$, an entry in the matrix, is a count of the number of pixel pairs with intensities p and q, for the given d and θ values. Note that these matrices are symmetric, as only the absolute value of d is used.

Haralick [11] and others suggested fourteen features to be computed from a cooccurrence matrix. For instance, the first one, called an energy feature,

$$f_1 = \sum_{p=1}^{N} \sum_{q=1}^{N} [S(p,q \mid d,\theta)]^2 . \tag{2.1}$$

This feature is a measure of uniformity of a region. For a uniform region, the cooccurrence matrix contains a small number of large-valued elements, hence the sum of squares is higher than it would be if all transitions were equally likely.

2.1.5 Texture Energy Measures

Texture energy in the Fourier Domain

If a texture is at all spatially periodic or directional, its power spectrum will tend to have peaks for corresponding spatial frequencies. These peaks can form the basis of features of a pattern recognition discriminator. One way to define features is to partition Fourier space into bins. Two kinds of bins, radial and angular, are commonly used. These bins, together with the Fourier power spectrum are used to define features. If F is the Fourier transform of the image, the Fourier power spectrum is given by $|F|^2$.

Radial features are given by

$$V_{r_1 r_2} = \iint |F(u,v)|^2 du\, dv\,, \tag{2.2}$$

where the limits of integration are defined by

$$r_1^2 \leq u^2 + v^2 < r_2^2\,, \tag{2.3}$$

where

$$0 \leq u, v < n - 1\,. \tag{2.4}$$

Radial features are correlated with texture coarseness. A smooth texture will have high values of $V_{r_1 r_2}$ for small radii, whereas a coarse, grainy texture will tend to have relatively higher values for large radii.

Features that measure angular orientation are given by

$$V_{\theta_1 \theta_2} = \iint |F(u,v)|^2 du\, dv\,, \tag{2.5}$$

where the limits of integration are defined by

$$\theta_1 \leq tan^{-1}\left(\frac{u}{v}\right) \leq \theta_2\,, \tag{2.6}$$

where

$$0 < u, v \leq n-1\,. \tag{2.7}$$

These features exploit the sensitivity of the power spectrum to the directionality of the texture. If a texture has many lines or edges in a given direction θ, $|F|^2$ will tend to have high values clustered around the direction in frequency space $\theta + \frac{\pi}{2}$.

Texture Energy in the Spatial Domain

Laws [12], [13] proposed an approach to measure texture energy in the spatial domain. His approach to texture characterization consists of two steps. First the image is convolved with a set of filters having a small region of support in the spatial domain. These filters are called micro-texture masks, and their outputs are called micro-texture features. These masks contain weighting coefficients needed in the two-dimensional convolution process. The two-dimensional convolution of the image $f(i,j)$ and mask $h(i,j)$ is given by the the relation

$$g(i,j) = h(i,j) * f(i,j) = \sum_{k=-a}^{a} \sum_{l=-b}^{b} h(k,l) f(i+k, j+l)\,, \tag{2.8}$$

for $i = 0, 1, \ldots, N-1$ and $j = 0, 1, \ldots, N-1$, and the $*$ denotes the two-dimensional convolution. Laws' micro-texture masks are designed to act as matched filters for certain types of quasi-periodic variations commonly found in textured images. Typically these masks are of size 5×5 or smaller and are zero-sum, resulting in an filtered image which is zero mean. Figure 2.1 shows the coefficients of four micro-texture masks that had the best discrimination power for the texture mosaic Laws used. Laws' texture

mosaic is similar to ours as shown in Figure 3.8. Because the micro-texture features are quasi-periodic, we expect strong variations about the mean output as a function of mask position for masks that are matched to the local texture. Thus the relevant information for texture discrimination is now present as the image variance of the micro-texture feature planes.

$$E5L5 = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \quad R5R5 = \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 16 & -24 & 16 & -4 \\ 6 & -24 & 36 & -24 & 6 \\ -4 & 16 & -24 & 16 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

$$E5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -2 & 0 & 4 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix} \quad L5S5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

Figure 2.1: Four of Laws' micro-texture masks.

Second, macro statistical features are obtained over large windows (e. g. 15×15 or 31×31). The most useful statistics are the sample variances or sample mean deviation of the micro-texture feature planes. The sample variance in the local windows of the filtered image can be measured in a variety of ways. The true sample variance within a $2n+1$ by $2n+1$ window at point (i,j) is given by

$$\sigma^2(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} (g(k,l) - m(i,j))^2 , \tag{2.9}$$

where the mean, $m(i,j)$, is given by

$$m(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} g(k,l) . \tag{2.10}$$

Also, the local sample variance can be defined as

$$\sigma^2(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} (g(k,l) - m(k,l))^2 . \tag{2.11}$$

Because the output of the small convolution masks is theoretically zero mean, the local variance may be approximated by assuming that the image is indeed zero mean and averaging the squares of the points within the window

$$\sigma^2(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} (g(k,l))^2 . \tag{2.12}$$

In Laws' thesis a computationally more efficient statistic was used instead. This statistic, called ABSAVE (or sample mean deviation), was computed as the mean deviation within a macro statistical window

$$s(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} |g(k,l) - m(i,j)| . \tag{2.13}$$

Again, based upon the assumption that the mean is zero, the ABSAVE $s(i,j)$ becomes

$$s(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} |g(k,l)| . \tag{2.14}$$

Laws' set of features have been shown to work as well or better than most others in texture classification problems. However, when Laws applied the texture energy features to the segmentation problem, only limited success was achieved. There are considerable errors in the interiors of the large regions and the algorithm sometimes performs badly near the borders between the textures.

2.1.6 Random Field Modeling

Random field models have increasingly been used to characterize digital images. For example, Hassner and Sklansky [14] proposed using Markov random fields as probabilistic

models of texture. Kashyap *et al.* [15] assumed that the given $N \times N$ image characterized by a set of intensity levels is a realization of an underlying random field model, known as the Non-causal Autoregressive (NCAR) model. This model is characterized by a set of parameters θ whose probability density function, $p_i(\theta)$, depends on the class to which the image belongs. It was shown that the maximum likelihood estimate $\hat{\theta}$ of θ is an appropriate feature vector for classification purposes. Chatterjee and Chellappa [16] assumed each texture is represented by a non-causal Gaussian-Markov Random field whose parameters can be used for segmentation purposes. Modeling images or textures as random fields enables stochastic filtering techniques such as hypothesis testing, significance testing, and parameter estimating, to be used for segmentation.

2.2 Survey of Segmentation Techniques

Segmentation algorithms can be classified broadly into four categories:

1. Characteristic feature thresholding and clustering,
2. Region growing,
3. Edge detection, and
4. Estimation theoretic approaches.

This section briefly discusses each of these categories.

2.2.1 Characteristic Feature Thresholding and Clustering

Characteristic feature thresholding is useful when the objects of interest in an image have a distinct property. In general, the threshold operator $T(*)$ can be viewed as a

test involving a function T of the form

$$T(i, j, N(i, j), f(i, j)), \tag{2.15}$$

where $f(i, j)$ is the characteristic feature and $N(i, j)$ denotes some local property of point (i, j), such as variance over a small neighborhood. Three types of threshold operators exist: global, local and dynamic. A global operator is one in which the $T(*)$ depends only on $f(i, j)$. If $T(*)$ depends on both $f(i, j)$ and $N(i, j)$, then it is called a local operator. If $T(*)$ depends on the coordinate values i, j as well as $f(i, j)$ and $N(i, j)$, then it is called a dynamic threshold.

Clustering, the multidimensional extension of thresholding, is needed when a single feature cannot distinguish the desired objects in an image. Clustering algorithms group the points in the characteristic feature space into clusters. These clusters are then mapped back to the original spatial domain to produce the segmented image results.

2.2.2 Region Growing

Region growing algorithms segment an image by merging groups of points. An image is first divided into subsets of points. Subsets with similar properties are iteratively merged, producing an image segmented into different regions. The algorithms differ in their initial selection of subsets, their choice of properties, and their merging criteria. Five commonly used techniques for region growing are the following: (1) regional neighbor search, (2) multiregional heuristics, (3) functional approximations, (4) split and merge, and (5) regional interpretation and semantics.

2.2.3 Edge Detection

Edge detectors segment images based on the detection of edges, discontinuities characterized by sudden changes in gray levels. There are two classes of edge detectors: parallel and sequential.

A parallel detector labels each point an edge or non-edge point solely by using the gray levels of neighboring points. Three methods of parallel detection are commonly used. In the first, high-emphasis spatial frequency filtering, the image is simply high-pass filtered. In the second, gradient detection, digital approximation to the ideal gradient is used. In the third, functional approximation, ideal edges are modeled by functions, an error criterion is established with these functions, and the actual edges are found at location where the error is minimized.

A sequential detector labels a point an edge or non-edge point using the results of previously classified points. Its success requires the existence of a good starting point and the use of an effective rule to label a point based on earlier results.

2.2.4 Estimation Theoretic Approaches

In the estimation theoretic approach, a random field model is assumed for each textured region along with a model for the occurrence of textured regions within an image. During the estimation step we try to maximize the joint likelihood of the image data with the underlying model. Therrien [17] used spatial linear filters driven by white noise as models for textures and a pixel based maximum likelihood (ML) rule for segmentation. He also showed segmentation results can be improved by using a estimate of the regions based on a binary model for the region geometry. Therrien's work was

subsequently generalized by Cohen and Cooper [18]. They used Gaussian Markov random field (GMRF) models for the individual textures and maximum likelihood method to label windows. Chatterjee and Chellappa [16] also used GMRF to model textures and a maximum likelihood rule to label the individual pixels.

2.3 Past Work on Textured Image Segmentation

There has been numerous previous work on the textured image segmentation problem. In order to provide a quick survey, we use a table to summarize previous results. Textured image segmentation approaches are characterized by two features: the texture measure and the segmentation algorithm. Table 2.1 to Table 2.3 list texture features and segmentation techniques used by various research workers.

2.4 An Overview of the Proposed Textured Image Segmentation System

Our approach to segmenting textured imagery is based on the following three principles:

1. The problem of estimating texture features without destroying the boundaries between regions is similar to the problem of smoothing a noisy image. Thus, techniques used to smooth noise which do not blur edges may be extended to the textured image segmentation problem to improve the accuracy along region boundaries. The intent is to develop algorithms to verify this concept through computer simulations.

Researcher	Texture Measure	Segmentation Algorithm
Bajcsy [19]	Fourier transform domain texture energy.	Brice and Fennema algorithm. Region merging used two heuristics.
Pavlidis and Tanimoto [20]	Fourier transform.	Split and merge.
Carlton and Mitchell [9]	Number of local extrema.	Region growing.
Pavlidis and Chen [21]	Co-occurrence matrix.	Used pyramid to implement a split and merge algorithm.
Pavlidis and Chen [22]	Mean and variance.	A sequence of decision problems within the framework of a split and merge algorithm.
Pavlidis and Chen [23]	Mean and covariance.	Split and merge followed by grouping algorithm using correlation coefficients and region adjacency graph.
Mitchell *et al.* [24]	Number of local extrema in a window and 4 ranges of extrema size followed by a local averaging.	Clustering in the four dimensional feature space.
Laws [12] [13]	Spatial texture energy.	Recursive region splitting method.

Table 2.1: A Survey of Past Work on Textured Image Segmentation (I)

Researcher	Texture Measure	Segmentation Algorithm
Davis and Mitiche [10]	First-order distribution of edge-based texture features.	Iterative nonlinear smoothing algorithm.
Knutsson and Granlund [25]	Local orientation and frequency calculated from quadrature filters.	Maximum likelihood classifier.
Schachter *et al.* [26]	Total variation of gray levels over a neighborhood of each point.	Clustering.
Deguchi and Morishita [27]	Two dimensional non-causal auto-regressive model.	Region growing.
Coleman and Andrews [28]	Local edge density over different region sizes.	Clustering. No spatial information used.
Zucker *et al.* [29]	Using spot detector to measure coarseness of texture.	Thresholding.
Pietikainen and Rosenfeld [30]	Second order gray level statistic.	Top-down/bottom-up linking applied to a "pyramid" of successively reduced resolution version of a image.

Table 2.2: A Survey of Past Work on Textured Image Segmentation (II)

Researcher	Texture Measure	Segmentation Algorithm
Weber and Sawchuk [31]	Laws texture energy features.	Multiresolution and spatial information.
Bevington and Mersereau [32]	Mean and covariance of Gaussian random field.	Approximation to a maximum likelihood estimator.
Therrien [17]	Spatial linear filters driven by white noise as models for the textures.	Both ML and MAP approaches were used.
Cohen and Cooper [18]	Individual textures are modeled as Gaussian random field.	Using ML estimation scheme to label windows.
Chatterjee and Chellappa [16]	Texture are modeled as GMRF.	Using ML estimation to label pixels.
Therrien [33]	ARMA model for texture.	Region transitions were modeled by MRF. ML and MAP estimation were applied to estimate segments.
Conners *et al.* [34]	Spatial gray level dependence method.	Split-type approach.
Derin and Cole [35]	Gibbs random field.	MAP estimate of the region process – a Gibbs random field.

Table 2.3: A Survey of Past Work on Textured Image Segmentation (III)

2. The spatial relationship of pixels should be taken into account in the process of solving textured image segmentation problems. In our work we will try to use both the global information in the feature space and the spatial organization of this data in the image space.

3. We view the unsupervised textured image segmentation as a problem of clustering with spatial constraints incorporated. Since there are no training samples available, a clustering technique is used to organize the patterns into clusters and then the cluster statistics are estimated. The spatial information is then used to improve the segmentation performance.

In chapter 3, a texture energy feature extraction method is proposed in order to improve the performance along region boundaries. Laws texture energy features were chosen because they have been shown to work as well or better than most others and are also relatively easy to calculate.

Chapter 4 deals with the problem of how to incorporate spatial information into our supervised image segmentor. The general idea of our approach is to label each pixel probabilistically based upon the similarity measure in the feature space, then invoke a relaxation labelling process to gradually bring in the spatial constraints.

Chapter 5 addresses the segmentation problem under the assumption that no training samples are available. A K-means clustering algorithm is used to group the patterns and to estimate the class statistics. In this chapter, the segmentation algorithm does not take spatial information into consideration.

In chapter 6, An unsupervised segmentation system with spatial information incorporated is developed. We use the class statistics estimated from chapter 5 as

a starting point to assign pixel labels probabilistically, then the relaxation labelling process is used to enforce the spatial constraints.

Chapter 7 is a detailed overview of our proposed textured image segmentation system. All the previously proposed algorithms are integrated in this chapter to form a complete system.

Chapter 8 concludes the dissertation. A discussion of future research on the problem is given.

Chapter 3

An Improved Method to Extract Texture Energy Features

In this chapter we propose an improved method to extract texture energy features for segmentation. We hope to improve segmentation results, especially along the borders of regions. One important point is that we think more emphasis should be put into the early stages of the segmentation process (such as feature extraction) in order to improve the overall performance.

In section 3.1, we first review Laws' approach to extracting texture energy features and segmenting images. From this review we may identify several areas that still need improvement. One shortcoming is that Laws' method to estimate texture energy features does not take the nonstationary nature of the problem into account. In section 3.2, we then introduce a quadrant smoothing method used by Jiang and Sawchuk [36] for noise smoothing and show how it is closely related to the texture feature extraction problem.

Once the connection between noise smoothing and feature extraction is made,

the implementation and analysis of the quadrant method to smooth feature estimates is discussed in section 3.3. A Bayes classifier is chosen to classify the pixels based upon the feature estimates made in section 3.3. Issues and assumptions related to the Bayes classifier are addressed in section 3.4. Section 3.5 provides an overall description of the proposed algorithm. Simulation results and performance comparisons are presented in section 3.6.

3.1 Laws Textured Image Segmentation System

As we mentioned in section 2.1.5, Laws' approach to texture characterization consists of two steps. First, micro-texture features are computed using 3×3 or 5×5 convolution masks. Second, macro-statistical features are obtained over large windows. The most useful statistics are local sample variances or averaged absolute value (ABSAVE) within a macro-window of the micro-texture feature planes. These features were called texture energy measures by Laws.

Once these texture energy features are extracted, an optional feature selection and extraction step was used to reduce the dimensionality of the classification process. There are two kinds of feature selection and extraction methods depending on whether the labeled prototypes for classes and their statistics are available or not. If the prototype labels are known, the multiple discriminant analysis method [37] may be used. If the prototype labels are not known, a feature extraction procedure that is often used is called principal components analysis [37]. We discuss both procedures here.

Laws described the following procedure for feature selection and extraction assuming that prototypes of textures are available:

- A set of raw features were generated by the aforementioned two steps, i.e., convolving the micro-texture masks with the image and then estimating macro-statistical features with a macro-window.
- A multivariate statistical method called a stepwise variable selection technique [38] was used to select a subset of effective features from the raw features.
- Canonical discriminant function analysis [39] was then used to further compress the discriminatory information by linearly combining those effective features selected in previous step. For instance, Laws extracted four features out of the original fifteen raw features and still retained most of the discriminating power.
- The output of the canonical discriminant analysis was a matrix of coefficients necessary for transforming the selected features into canonical discriminant functions. These functions are derived in such a way that they best exhibit the differences among the groups. In general, the number of canonical discriminant functions is equal to either the number of input variables or the number of classes minus one, whichever is smaller.

A minimum distance-to-centroids classifier was then used to assign the label of test observation. No spatial information was included in this procedure.

If labeled prototypes were not available, Laws used the different feature extraction method called principal components analysis. The absence of class membership information of the training observations implies that the class-conditional parameters can not be inferred from the data. In such a situation the only sensible approach to feature extraction is take advantage of the general information-compression properties

of the Karhunen-Loéve (K-L) expansion [40]. More specifically, a feature extraction matrix W is constructed from the K-L coordinate axes, which are given by column vectors $\vec{w}_j$, $j = 1, \ldots, D$, satisfying

$$\Sigma_x \vec{w}_j - \lambda_j \vec{w}_j = 0$$

where Σ_x is the covariance matrix. Σ_x is defined as

$$\Sigma_x = E((\vec{x} - \vec{m}_x)(\vec{x} - \vec{m}_x)^t),$$

where $\vec{x}$ is the D-dimensional random feature vector and $\vec{m}_x$ is the mean vector of $\vec{x}$. Both $\vec{x}$ and $\vec{m}_x$ are column vectors. To retain most of the information in $\vec{x}$, vectors $\vec{w}_j$, $j = 1, \ldots, d$, $d \leq D$, which form the columns of the feature extractor

$$W = [\vec{w}_1 \vdots \vec{w}_2 \vdots \ldots \vdots \vec{w}_d]$$

must be selected in the descending order of magnitudes of their corresponding eigenvalues, i.e.,

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq \ldots \lambda_D.$$

After the principal component planes were generated, the first three were then input to the "Ohlander segmentor" algorithm [41]. The Ohlander segmentor is a region based segmentor. Regions are split recursively based upon histogram analysis of the feature values. It begins by defining a mask that initially covers all pixels of the image. Given a mask, a histogram of the masked image is computed. The separation of one mode of the histogram in the feature space from another mode is then followed. Pixels on the image are then identified with the cluster to which they belong. If there is only one feature space cluster, then the mask is terminated. If there is more than one

cluster, then each spatially connected component of all pixels with the same cluster is, in turn, used to generate a mask which is placed on a mask stack. During successive iterations the next mask in the stack selects pixels in the histogram computation process. Clustering is repeated for each new mask until the stack is empty [3].

Figure 3.1 is a block diagram of Laws texture segmentor, where $F_1, F_2, \cdots, F_p$ represent different micro-textured masks, $E_1, E_2, \cdots, E_p$ are the calculated energy features. After the feature selection and extraction, feature vectors reduce to a lower dimension q which are then fed to the classifier for classification.

Laws' set of features have been shown to work as well or better than most others in texture classification problem. However, when Laws applied the texture energy features to the segmentation problem, only limited success was achieved. There are considerable errors in the interiors of the large regions and the algorithm sometimes performs badly near the borders between the textures. One major reason for the segmentation error along the region borders is due to the method of estimating the local texture energy. As we recall, the macro statistical features are obtained by processing the micro-texture feature planes with a nonlinear "local texture energy" filter. This nonlinear filter was used to estimate the local sample standard deviation (or a similar statistic approximated by the moving window average of the absolute values) of the filtered image. Although such moving window operations are simple and fast, it introduces significant errors along the region borders. The reason for this is the overlap of the averaging window at texture boundaries. When this occurs, the resulting statistics become the mixture of two sets of statistics. This mixture of statistics sometimes resemble another distinctly different texture class. In this case, the boundary pixels will

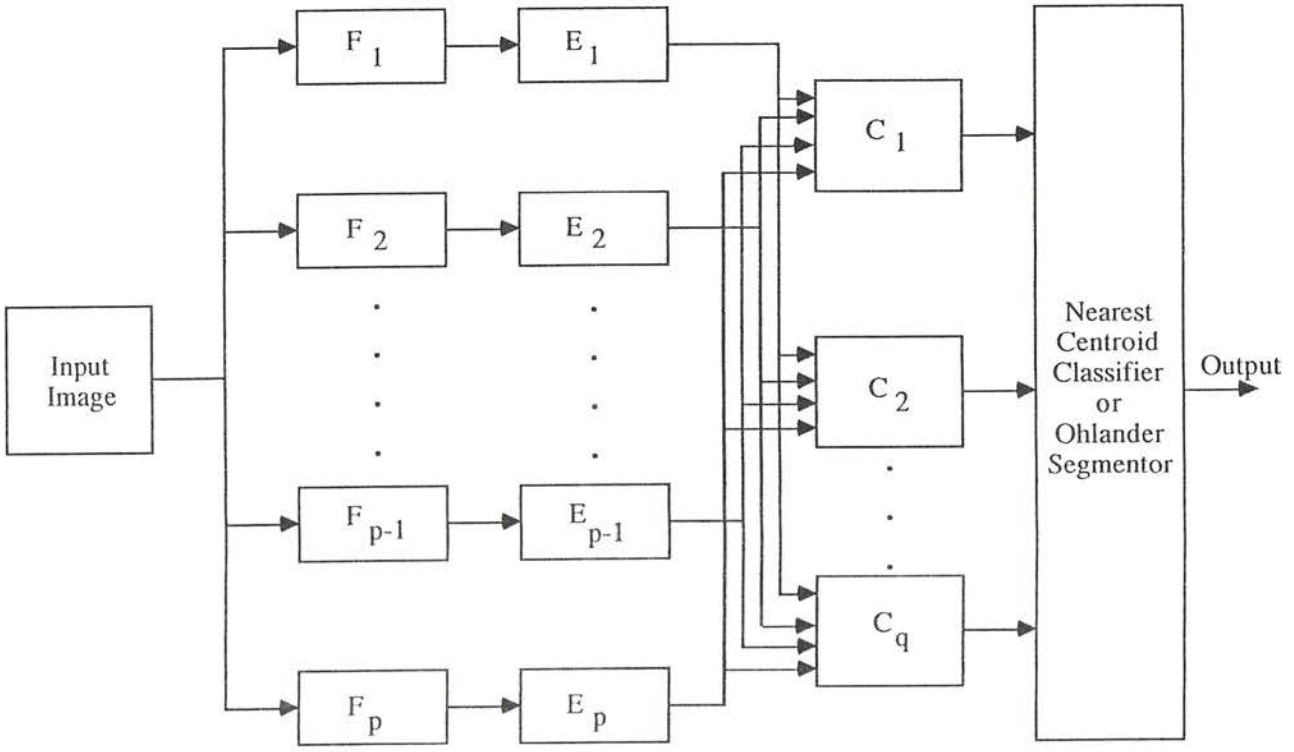


Figure 3.1: Block Diagram of Laws' Texture Segmentor.

be incorrectly classified as a third class different from the two inside the overlapping windows.

One way to avoid the aforementioned problem is to estimate the sample standard deviation (or sample mean deviation) of the micro-texture filtered images more carefully. We can perceive the process of getting the macro statistical features as the problem of smoothing a noisy image. In noise smoothing we always face the following problem: how to smooth noise without blurring the edges. There are many ways to approach the noise smoothing problem, and we want to extend some of these concepts to the feature extraction problem. A quadrant method has successfully been used by Jiang and Sawchuk [36] in their noise smoothing work. Let us first take a brief look at how quadrant method can be used to smooth noise without blurring edges.

3.2 Edge Preserving Noise Smoothing Methods

Consider a noisy observation given by

$$g(i,j) = f(i,j) + u(i,j)\,, \tag{3.1}$$

where the noise part, $u(i,j)$, has zero mean and is uncorrelated with the signal $f(i,j)$. In addition, $\{u(i,j)\}$ are pairwise uncorrelated. The Locally Linear Minimum Mean Square Error (LLMMSE) estimator suggested by Kuan *et al.* [42] has the property that it smoothes out noise in flat regions and leaves the observation unchanged in the vicinity of edges. The LLMMSE estimator is defined as

$$\hat{f}(i,j) = m_g(i,j) + \frac{v_g(i,j) - v_u(i,j)}{v_g(i,j)}\left[g(i,j) - m_g(i,j)\right]\,, \tag{3.2}$$

where $m_g(i,j)$ and $v_g(i,j)$ are the local mean and local variance of the observation g at (i,j), respectively, and $v_u(i,j)$ is the variance of the noise $u(i,j)$.

There are several ways to estimate the local statistics. In the next two subsections, we review two of these methods: Kuan *et al.*'s method and Jiang-Sawchuk's method.

3.2.1 Kuan *et al.*'s method

Kuan *et al.* [42] used a $(2m+1) \times (2n+1)$ 2-D local window to calculate the local statistics. The local mean is defined as

$$m_g(i,j) = \frac{1}{(2m+1)(2n+1)} \sum_{k=i-m}^{i+m} \sum_{l=j-n}^{j+n} g(k,l). \tag{3.3}$$

The local variance is defined as

$$v_g(i,j) = \frac{1}{(2m+1)(2n+1)} \sum_{k=i-m}^{i+m} \sum_{l=j-n}^{j+n} c(i-k,j-l) \times [g(k,l) - m_g(k,l)]^2, \tag{3.4}$$

where $c(i,j)$ is a weighting function.

There are two major points in the above equation. First, the local mean is allowed to vary within the window for the calculation of local variance. Second, the local window is not necessarily uniformly weighted. Since there is no physical support on how these weighting factors should be set, they can only be chosen on a heuristic or *ad hoc* basis. A Gaussian shaped weighting window is suggested in Ref. [42].

3.2.2 Jiang-Sawchuk's method

The local mean is defined the same way as before. For simplicity, a $(2m+1)\times(2m+1)$ square window is used in the following expressions. Thus the local mean is

$$m_g(i,j) = \frac{1}{(2m+1)^2} \sum_{k=-m}^{+m} \sum_{l=-m}^{+m} g(i+k, j+l)\,. \tag{3.5}$$

Jiang and Sawchuk [36] proposed that the local variance $v_g(i,j)$ be computed by

$$v_g(i,j) = \min\left[v_{g_1}(i,j), v_{g_2}(i,j), v_{g_3}(i,j), v_{g_4}(i,j)\right]\,, \tag{3.6}$$

where $v_{g_1}(i,j)$, $v_{g_2}(i,j)$, $v_{g_3}(i,j)$, and $v_{g_4}(i,j)$ are the four quadrant variances defined by

$$v_{g_1}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i+k,\, j-l) - m_g(i+k,\, j-l)\right]^2\,, \tag{3.7}$$

$$v_{g_2}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i-k,\, j-l) - m_g(i-k,\, j-l)\right]^2\,, \tag{3.8}$$

$$v_{g_3}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i-k,\, j+l) - m_g(i-k,\, j+l)\right]^2\,, \tag{3.9}$$

$$v_{g_4}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} \left[g(i+k,\, j+l) - m_g(i+k,\, j+l)\right]^2\,. \tag{3.10}$$

The locations of the neighboring pixels involved in the above four quadrant variances are schematically shown in Figure 3.2. Note that the window sizes for calculating local mean and quadrant variances are not necessarily the same. The performance of the filter strongly depends on how the local windows are chosen and how the local statistics are calculated.

The idea behind Eq. (3.6) is that when there is an edge near pixel (i,j) the smallest quadrant variance has the best chance of being the most representative statistic over those neighboring pixels around (i,j) that do not appear across the edge.

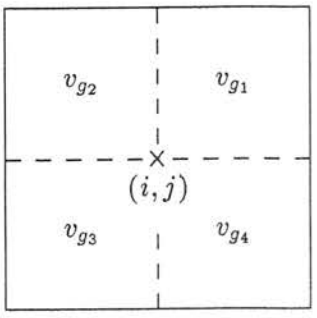


Figure 3.2: Location of the Four Quadrant Variances.

The estimated local mean and local variance are then used by the LLMMSE filter. In the next subsection we describe another way to use the local variance information estimated by the quadrant method of Eqs. (3.6) - (3.10) to perform edge preserving noise smoothing.

3.2.3 Edge Preserving Noise Smoothing Using a Quadrant Method

Our method is to use the four windows that define quadrants around each pixel and then replace the value of that pixel by the average gray level of the most homogeneous neighborhood as determined by Eq. (3.6). If a window contains a sharp edge, the variance of the gray level in that window becomes large. Here the quadrant variance as defined by Eq. (3.6) are used to measure the gray level variability around a pixel in order to find the most homogeneous neighborhood. For the convenience of future reference to this method, we call it the Edge Preserving Noise Smoothing Quadrant (EPNSQ) filter.

We can analytically derive the expression for the sample variance of a window which overlaps two regions R_1 and R_2. Let us assume pixels in R_1 are random variables

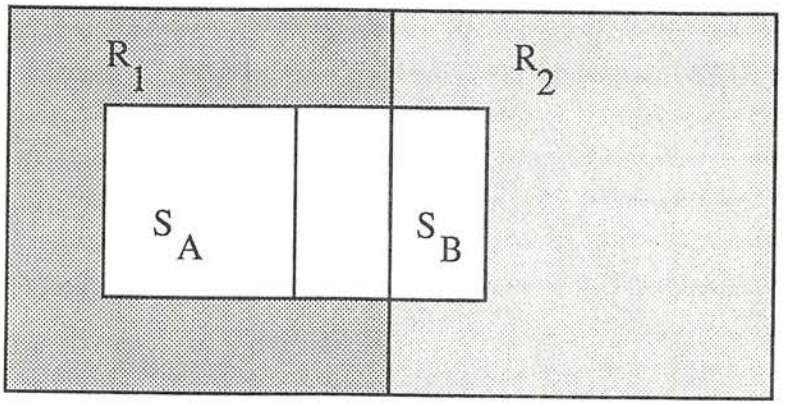


Figure 3.3: When A Pixel Is Near the Border of Two Regions with Different Means and Variances.

and variance σ_2^2 (see Figure 3.3). The sample variance of the neighborhood S_B which includes both parts of R_1 and R_2 can be written as

$$S_B^2 = \frac{1}{N}[\sum_{\substack{i=1 \\ i \epsilon R_1}}^{N_1} (X_i - \overline{X})^2 + \sum_{\substack{j=1 \\ j \epsilon R_2}}^{N_2} (X_j - \overline{X})^2], \tag{3.11}$$

where N_1 and N_2 denote the number of points in R_1 and R_2, respectively, $N_1 + N_2 = N$, and $\overline{X}$ is the sample mean

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{N} X_i \,. \tag{3.12}$$

The expected value of S_B^2 is

$$\mathrm{E}(S_B^2) \approx \frac{1}{N}[N_1\sigma_1^2 + N_2\sigma_2^2 + \frac{N_2 N_1}{N}(m_1 - m_2)^2] \,. \tag{3.13}$$

Similarly we can derive the expected value of the sample variance of the neighborhood

S_A which is completely included in R_1

$$\mathrm{E}(S_A^2) = \sigma_1^2 \,. \tag{3.14}$$

If $\mathrm{E}(S_B^2) > \mathrm{E}(S_A^2)$, that is, $\sigma_2^2 + (\frac{N_1}{N})(m_1 - m_2)^2 > \sigma_1^2$, the neighborhood containing only R_1 is selected for smoothing. In most gray level images the above condition,

$$\sigma_2^2 + (\frac{N_1}{N})(m_1 - m_2)^2 > \sigma_1^2 \,,$$

can be met.

From the above analysis we know why the EPNSQ filter can be used to smooth noise while preserving edges. Some simulation results can demonstrate the effectiveness of this EPNSQ filter.

3.2.4 Simulation Results

Figure 3.4 shows the simulation results of using a simple averaging filter and LLMMSE filter of Eq. (3.2) to smooth a noisy image. Figure 3.4(a) is the original image. Figure 3.4(b) is the degraded image with a signal-independent additive white noise. The signal-to-noise ratio (SNR) of the degraded image is about 6 dB. We define the SNR as the ratio of the signal variance to the noise variance. The signal variance is defined as the signal sample variance of the whole image. Figure 3.4(c) is the simple averaging filter output which is the the degraded image convolved with a 7×7 uniform window. The result of using the LLMMSE and quadrant method with $m = 3$ in Eq. (3.5) and $w = 4$ in Eqs. (3.7) to (3.10) for variance estimation is shown in Figure 3.4(d). The sharpness of edges and the amount of noise smoothed out are better when compared subjectively.

subjectively.

The simulation results of using the EPNSQ filter is shown in Figure 3.5. Figure 3.5(a)(b)(c) are the original, degraded, and simple averaged images, respectively. Figure 3.5(d) is the output of our EPNSQ filter with quadrant window of size 7×7. The sharpness of edges and the smoothness of interior regions are quite impressive.

In the next section, we will discuss how to use the EPNSQ filter to smooth our feature estimates.

3.3 Using the EPNSQ Filter to Smooth Feature Estimates

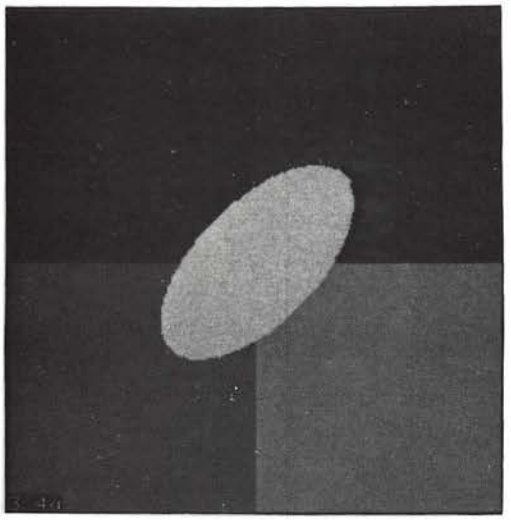
Our problem on hand can be formulated as the estimation of the local sample variance over regions having a mixture of statistics. The way we propose is to first find the Laws features by convolving the input image $f(i,j)$ with a set of micro-texture filters, each denoted by $h(i,j)$ and having a form similar to the micro-texture filter in Figure 2.1 to produce a set

$$g(i,j) = h(i,j) * f(i,j) = \sum_{m=-a}^{m=a} \sum_{n=-a}^{n=a} h(m,n) f(i+m, j+n) \,. \tag{3.15}$$

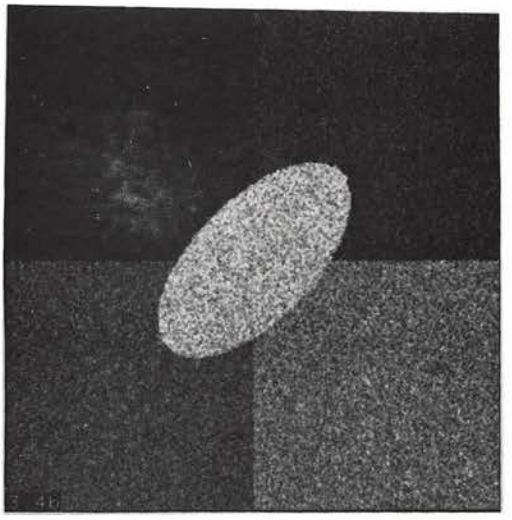
Each of the micro-texture filter output images are then processed by a 7×7 "local texture energy" filter

$$s(i,j) = \frac{1}{(2n+1)^2} \sum_{k=i-n}^{i+n} \sum_{l=j-n}^{j+n} |g(k,l)| \,. \tag{3.16}$$

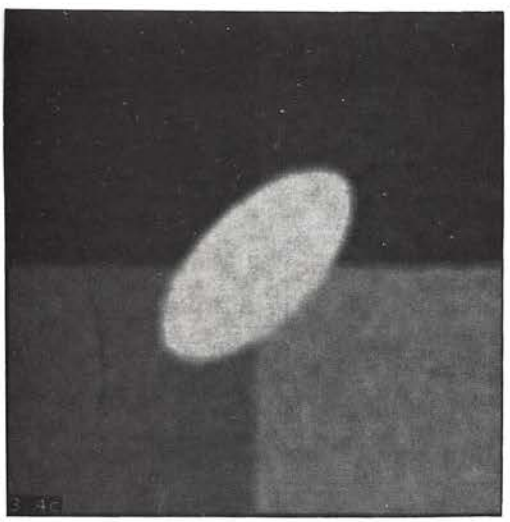
Now each pixel of $s(i,j)$ is an estimate of the local sample standard deviation. By using such a small window to estimate the local standard deviation, the variability of



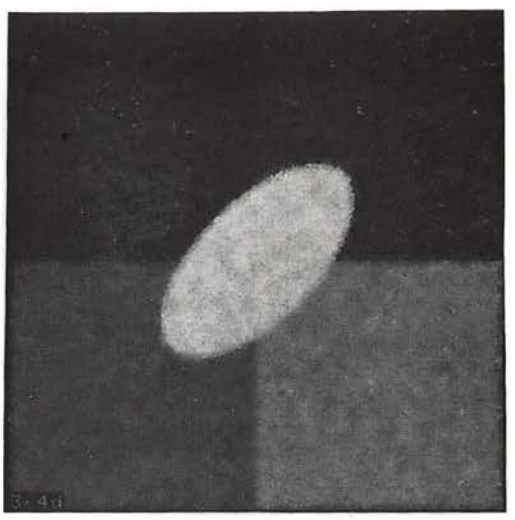
(a) Original Image



(b) Noisy Image

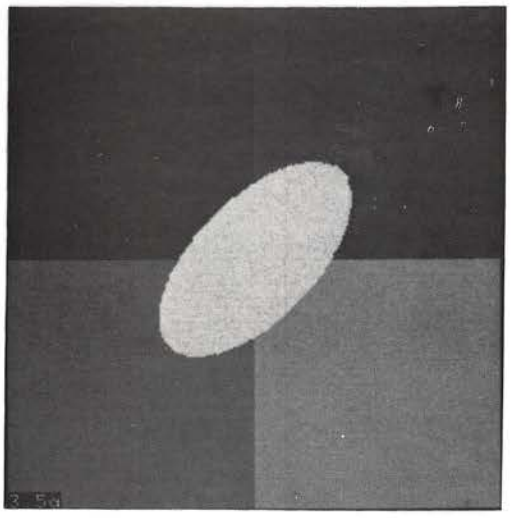


(c) Averaging Filter

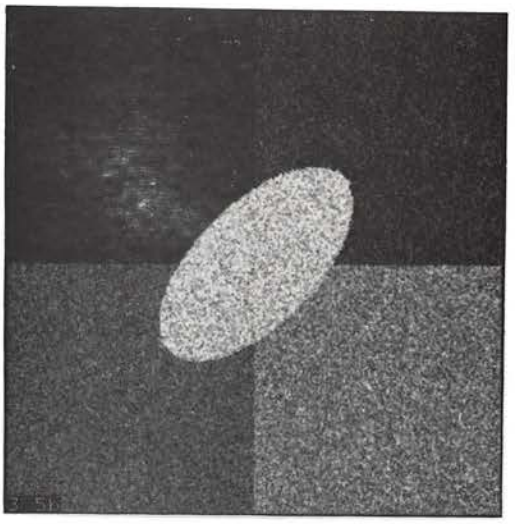


(d) LLMMSE Filter

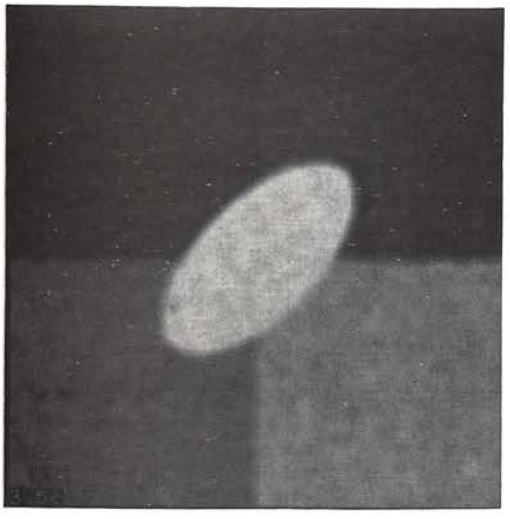
Figure 3.4: Comparison between an Averaging Filter and the LLMMSE Filter.



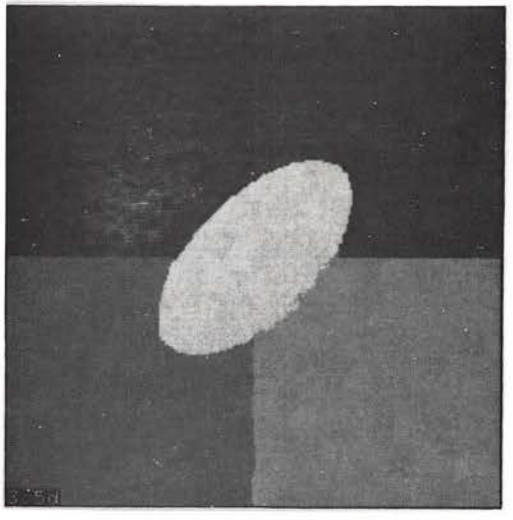
(a) Original Image



(b) Noisy Image



(c) Averaging Filter



(d) Quadrant Window Filter

Figure 3.5: Comparison between an Averaging Filter and the Proposed Filter.

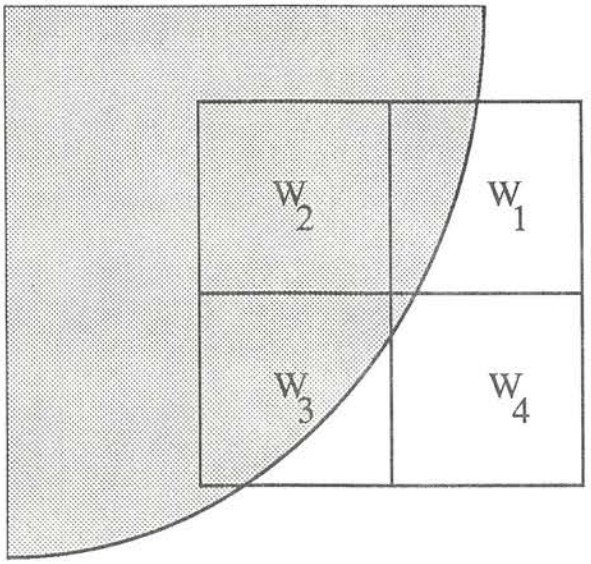


Figure 3.6: When A Pixel Is Near the Region Border At Least One of the Four Windows Is Entirely Included in One Region.

The choice of window sizes used to estimate the local statistics is actually a trade-off problem. Using small window sizes (e.g. 7×7), we can estimate the statistics more accurately near the borders and in the small regions, but we have higher statistical variability compared to the use of a large window due to the smaller number of data points. On the other hand, using large window sizes (e.g. 15×15) provides more data points to reduce the statistical variability of the estimates in the large regions, but performs poorly near the borders between the textures and in the small regions. We propose a two stage scheme with varying local window sizes (see Figure 3.7). A small window size (e.g. 7×7) is first used to estimate the local statistics. Based upon Laws' study, using 7×7 macro statistical windows can provide a reasonably good classification rate while avoiding some of the mixture of statistics along region

borders. A larger window size (e.g. 15 × 15) EPNSQ filter is then used to provide more smoothing power. Since region borders have already been accurately estimated by the smaller macro statistical window, the EPNSQ filter should be able to identify and smooth those homogeneous regions. The window size used by the quadrant filter should be smaller than the smallest region size which we expect to segment accurately.

3.4 Bayes Classifier

After the texture energy features have been extracted, a classifier needs to be selected. For purposes of our study, a Bayes classifier is used. According to the Bayes decision rule, a point with feature vector $\vec{x}$ should be classified to class λ_i if and only if

$$P(\lambda_i|\vec{x}) > P(\lambda_j|\vec{x}) \qquad for\ all \quad j \neq i\,, \tag{3.25}$$

or equivalently, by Bayes rule

$$p(\vec{x}|\lambda_i)P(\lambda_i) > p(\vec{x}|\lambda_j)P(\lambda_j) \qquad for\ all \quad j \neq i\,, \tag{3.26}$$

where $p(\vec{x}|\lambda_i)$ is the class conditional probability density function for $\vec{x}$.

It is not always necessary to compare the actual probability functions as shown above to make a decision. Instead, discriminant functions which satisfy the same relationships may be used. If we assume the class conditional probability densities are multivariate Gaussian with mean $\vec{m}_i$ and covariance matrices Σ_{λ_i}, the discriminant functions may be defined as

$$\begin{aligned} g_i(\vec{x}) &= \log(p(\vec{x}|\lambda_i)P(\lambda_i) = \log(p(\vec{x}|\lambda_i)) + \log(P(\lambda_i)) \\ &= \frac{-1}{2}(\vec{x}-\vec{m}_i)^t\Sigma_{\lambda_i}^{-1}(\vec{x}-\vec{m}_i) - \frac{d}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_{\lambda_i}| \end{aligned} \tag{3.27}$$

not sacrifice the region border accuracy, we now adapt the EPNSQ filter used for gray level image smoothing to feature smoothing. Instead of smoothing the sample standard deviation at each point indiscriminately, the EPNSQ filter smoothes the sample standard deviation estimates from a chosen neighbors that seem likely to belong to the same region as the given point. The next question is how do we know the chosen neighbors belong to the same region as the given point or not? Because our main concern is to avoid averaging over region borders, the problem becomes one of finding a measure which indicates whether there is a region border or not.

Since at this stage each pixel represents a local sample standard deviation, and we know the sample variance can be used as a measure of the nonhomogeneity of regions, with these two facts in mind we propose the use of "*sample variance of sample standard deviation*" as the measure for the decision rule in our quadrant method. At each pixel a set of four neighborhoods that lie on various sides of the point are examined. Figure 3.6 shows an example with the given pixel near a boundary of a region border and the locations of the four windows. The sample mean m_{W_1} and sample variance v_{W_1} of W_1 are defined by:

$$m_{W_1}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i+k, j-l)\,, \tag{3.17}$$

and

$$v_{W_1}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} [s(i+k, j-l) - m_{W_1}(i,j)]^2\,, \tag{3.18}$$

where $s(i,j)$ represents the local sample deviation at location (i,j). Similarly, the

sample means and sample variances of windows W_2, W_3, and W_4 are defined as:

$$m_{W_2}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i-k, j-l)\,, \tag{3.19}$$

$$m_{W_3}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i-k, j+l)\,, \tag{3.20}$$

$$m_{W_4}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} s(i+k, j+l)\,, \tag{3.21}$$

$$v_{W_2}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} [s(i-k, j-l) - m_{W_2}(i,j)]^2\,, \tag{3.22}$$

$$v_{W_3}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} [s(i-k, j+l) - m_{W_3}(i,j)]^2\,, \tag{3.23}$$

$$v_{W_4}(i,j) = \frac{1}{w^2} \sum_{k=0}^{w-1} \sum_{l=0}^{w-1} [s(i+k, j+l) - m_{W_4}(i,j)]^2\,. \tag{3.24}$$

Those windows that contain region borders generally have a higher variability introduced by the edges. Thus we define our final texture statistic as the sample mean $m_{W_p}(i,j)$ (either Eq. (3.17), (3.19), (3.20), or (3.21)) corresponding to the sample variance $v_{W_p}(i,j)$ which is the lowest among the four quadrants (either $v_{W_1}(i,j)$, $v_{W_2}(i,j)$, $v_{W_3}(i,j)$, or $v_{W_4}(i,j)$).

3.3.1 Window Sizes

Another issue that needs to be addressed is the effects of using different window sizes to estimate the local sample statistics. First of all, let us assume the class of textures we are working with are those fine-grained textures typically used in our experiment. This assumption is made because statistical methods are usually suitable for micro-textures, i.e., textures with short correlation distance. Otherwise, methods such as structural or structural-statistical [43] may be a better choice.

$$+ \quad \log(P(\lambda_i)) \,, \tag{3.28}$$

where d is the dimensionality of the feature set. The $\frac{d}{2}\log 2\pi$ term is common to all functions and can be removed. For the case where the classes are present in equal number, the *a priori* probabilities are equal for all classes and the $\log(P(\lambda_i))$ term can also be removed. This results in equivalent discriminant functions given by

$$g_i(\vec{x}) = (\vec{x} - \vec{m}_i)^t \Sigma_{\lambda_i}^{-1} (\vec{x} - \vec{m}_i) + \log|\Sigma_{\lambda_i}| \,. \tag{3.29}$$

The Bayes decision rule then classifies a point to class i if and only if

$$g_i(\vec{x}) < g_j(\vec{x}) \qquad \text{for all} \quad j \neq i \,. \tag{3.30}$$

The classifier described above was implemented in our algorithm.

3.5 Proposed Algorithm Description

The following is a description of the proposed algorithm:

1. The textured image shown in Figure 3.8 is first convolved with a set of 5×5 micro-texture masks. In Laws' work, a principal component transformation was then used to allow selection of a working subset containing the most significant transformed features. In our work, because feature selection is not our main concern, a subset of four micro-texture masks, which correspond to the E5L5, E5S5, L5S5 and R5R5 masks used by Laws (shown in Figure 2.1), are selected in advance. The output of these four masks is used without performing any principal component rotation or feature selection.

2. Because the micro-texture masks are zero-sum, the resulting filtered images have mean that are close to zero, and the local standard deviation may be approximated by averaging the absolute values within the window. In our work we used a 7×7 window to estimate this statistic (see Eq. (3.16)).

3. The local standard deviation estimated by the 7×7 window in step 2 is now smoothed by the EPNSQ filter. At each pixel, we examine a set of four neighborhoods that lie on various sides of the pixel, then use local sample variance as the measure of variability over each neighborhood, and replace the pixel by the average of the neighborhood that has the lowest variance. This selects the neighborhood that is most likely to lie entirely within a uniform region of the picture. This EPNSQ filtering was applied to each of the four images representing the feature plane outputs of the four original micro-texture masks. In this step we use window size 15×15.

4. The estimated feature vectors are then sent to the Bayes classifier described in section 3.4. Equal *a priori* probabilities for each class are assumed and means and covariances for each class are estimated from the prototypes. The output of the Bayes classifier is a gray level coded classification result.

The block diagram of the proposed scheme is depicted in Figure 3.7. Some of the feature image planes smoothed by the EPNSQ filter are presented in the next section.

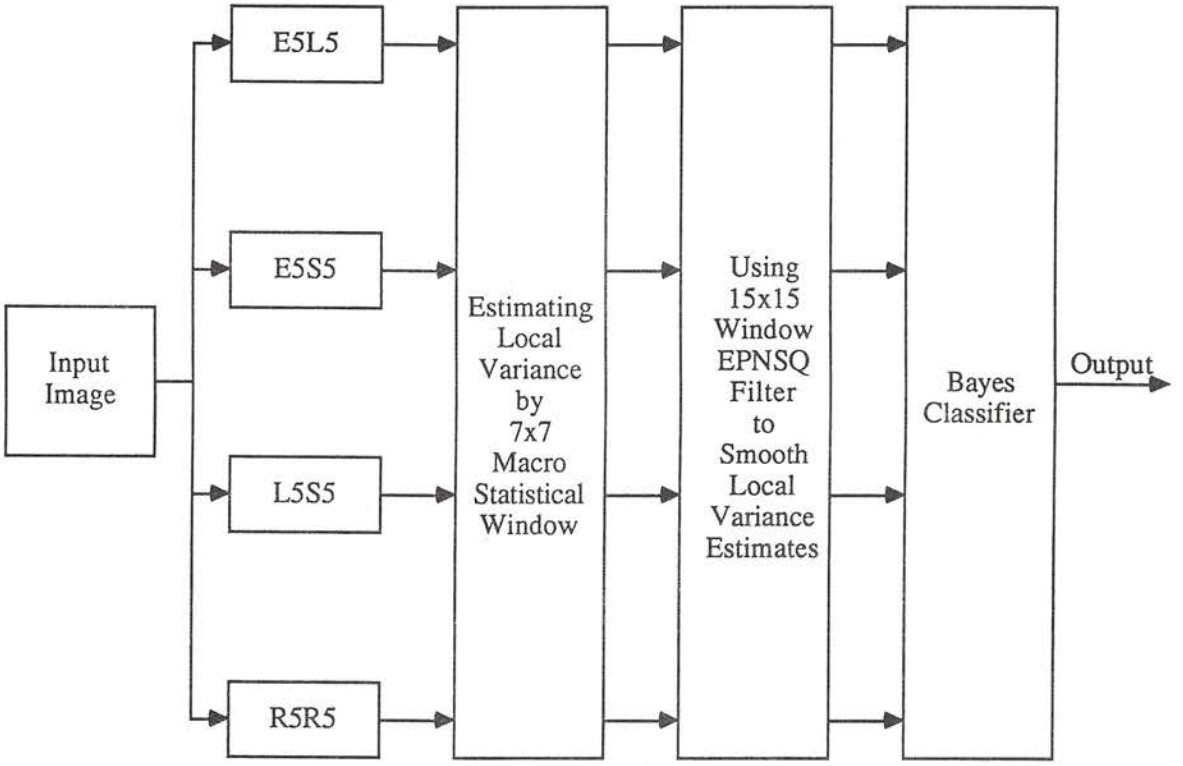


Figure 3.7: Block Diagram of the Proposed Scheme.

3.6 Simulation Results

The test image used in our work is a texture mosaic (Figure 3.8)(a) which consists of eight different textures: grass, water, sand, wool, pigskin, leather, raffia, and wood. The texture images we have chosen are from an album by Brodatz [44]. All eight textures are present in the image in squares of size 128×128, 64×64, 32×32, and 16×16. All components are histogram equalized so that all textures have same first order statistics. In other words, first order statistics alone are not sufficient for discriminating different textures. Figure 3.8(b) is the gray level coded ground truth of Figure 3.8(a).

Figures 3.9(a), 3.9(b), 3.10(a), and 3.10(b) are the resulting E5L5, E5S5, L5S5, and R5R5 feature image planes, respectively. These images have been scaled to an eight-bit range for the purpose of viewing. Notice the location of the region borders are generally very accurate and the interior regions are fairly uniform.

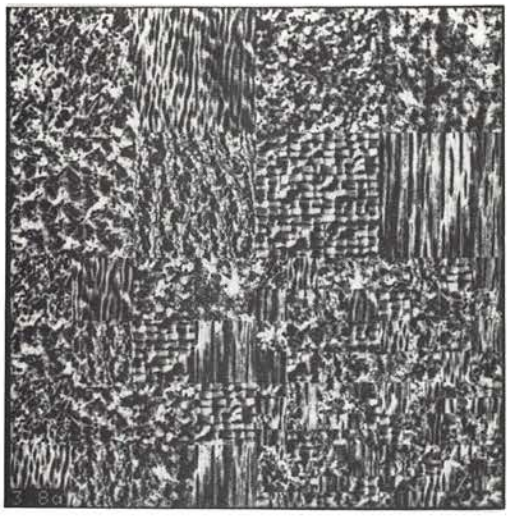
The Bayes classifier is then used to classify the estimated feature data. Figure 3.11(a) shows the gray level coded classification result using feature data estimated by Laws' method. The window size used for estimating statistics is 15×15. Note the accuracy near some of the borders between textures is not very satisfactory. Figure 3.11(b) is a binary image of the classification result in which the black pixels represent correctly classified ones, the white pixels represent misclassified ones.

Figure 3.12(a) is the gray level coded classification result using the proposed algorithm to estimate the feature data. A 7×7 window is used for statistics estimation and a 15×15 window is used for EPNSQ smoothing. Figure 3.12(b) shows the binary image of the classification result in which the black pixels represent correctly classified ones, the white pixels represent misclassified ones. Table 3.1 lists the correct

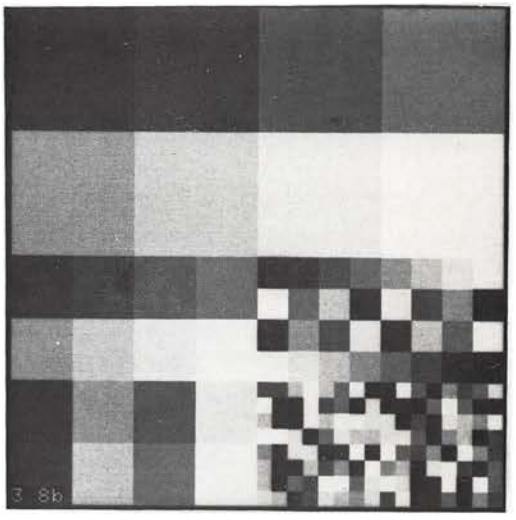
classification rate of the overall image and of each sub-image containing regions of a particular size. From Figure 3.12 and Table 3.1 we can see not only the correct classification rate but also the accuracy along borders has been improved significantly (except in the region containing 16×16 sizes, an area which is very difficult for a human to discriminate).

For our test image, the chosen window sizes, i.e., 5×5 for micro-texture masks, 7×7 for macro statistics estimation, and 15×15 for EPNSQ filter, worked very well. This sequence of window sizes may not be a good choice for all images. In other words, to choose a appropriate sequence of window sizes is a data dependent problem. In the process of choosing window sizes, the following guidelines must be considered:

- The window sizes for micro-texture masks are typically 3×3, 5×5, or 7×7, depending on the spatial scale size of the micro-texture features.
- The window size used for macro statistic estimation should be large enough to include a representative sample of the image texture. The coarser the texture the larger the window size should be.
- The window size used for EPNSQ filter should be smaller than the smallest region size in which we expect to have a good segmentation.

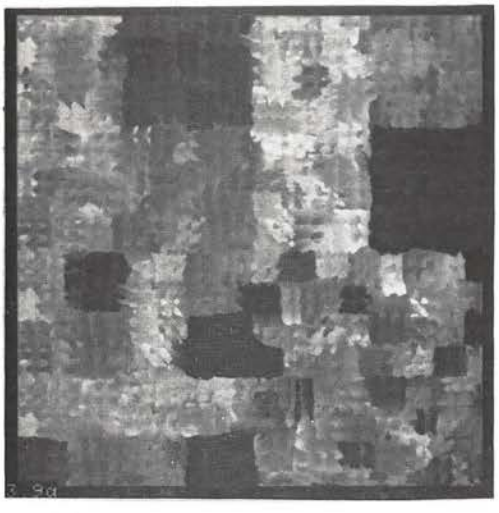


(a) Texture Mosaic

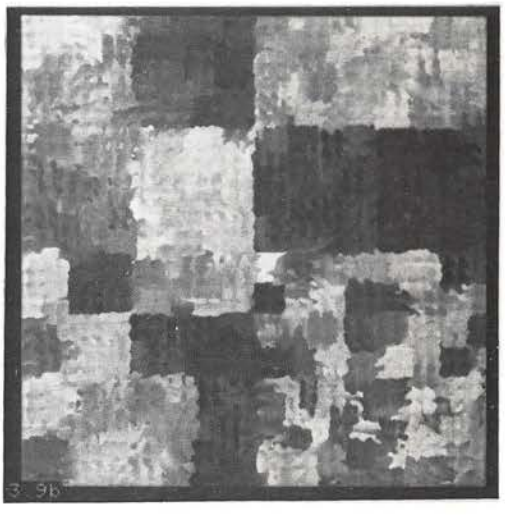


(b) Ground Truth

Figure 3.8: Texture Mosaic for Experimental Work.

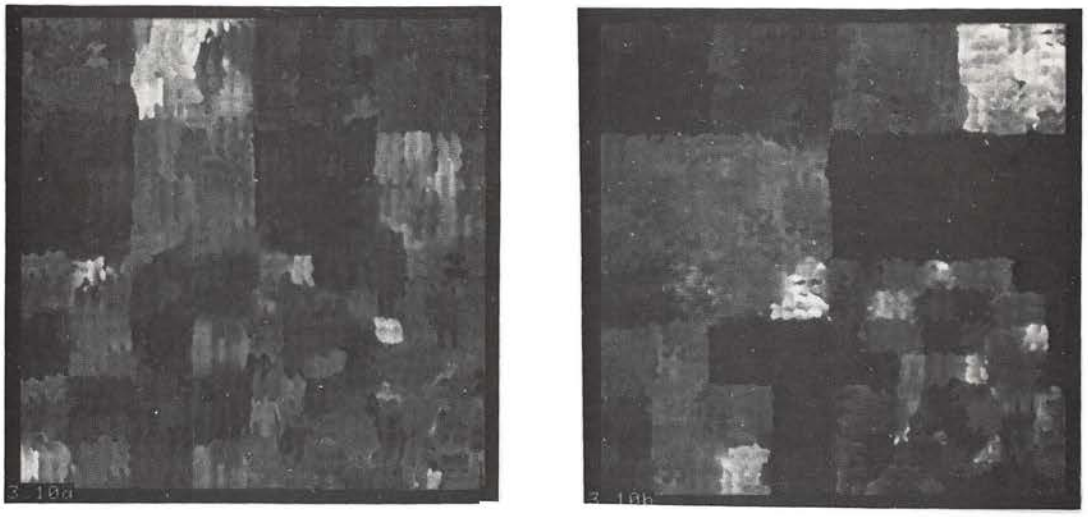


(a) E5L5



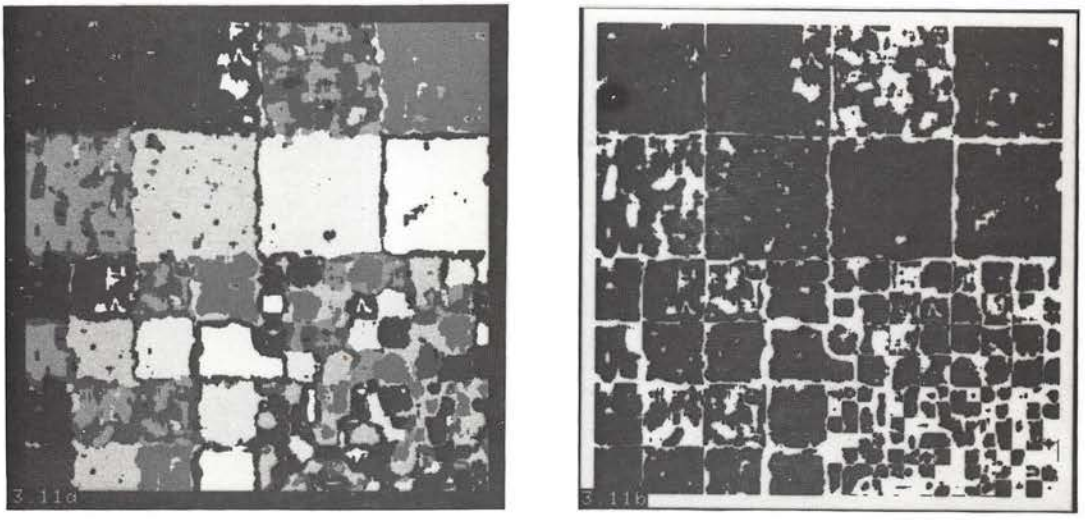
(b) E5S5

Figure 3.9: E5L5 and E5S5 Feature Image Planes.



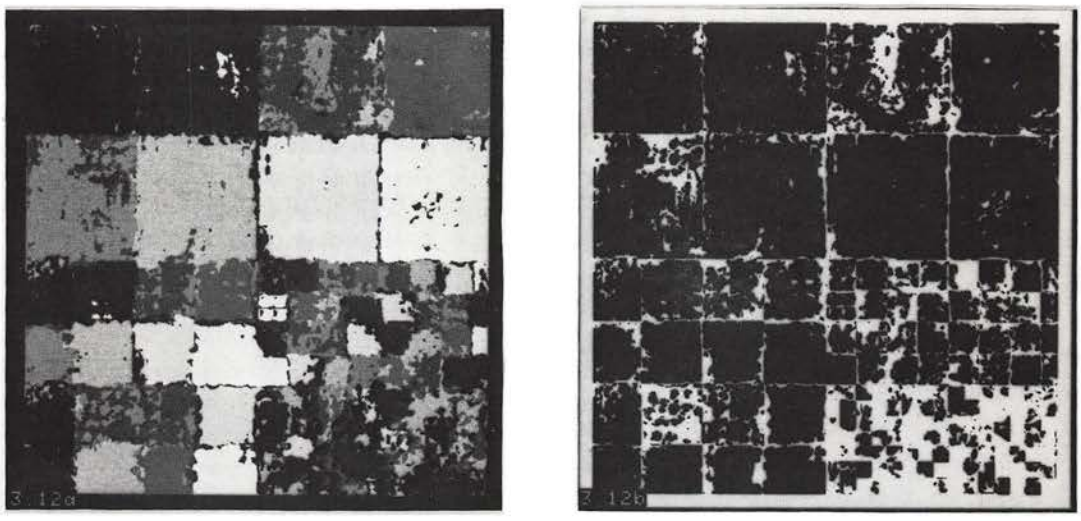
(a) L5S5 (b) R5R5

Figure 3.10: L5S5 and R5R5 Feature Image Planes.



(a) Gray Level Coded Result (b) Misclassified Pixels in White

Figure 3.11: Classification Results Using 15 × 15 Macro-statistical Window.



(a) Gray Level Coded Result

(b) Misclassified Pixels in White

Figure 3.12: Classification Results Using Proposed Method.

Region Sizes	15 × 15	7 × 7–15 × 15 EPNSQ Filter
Overall	72.9	76.6
128 × 128	81.1	88.1
64 × 64	74.1	78.6
32 × 32	64.5	66.1
16 × 16	44.3	37.3

Table 3.1: Classification Accuracy (%) of Two Feature Extraction Schemes

Chapter 4

Supervised Segmentation Algorithm with Spatial Constraints

The supervised segmentation technique we have so far considered has primarily ignored the spatial relationship between pixels. The weakness of classifying pixels based solely upon feature space distribution is that the formation of clusters in the feature space does not take into consideration the spatial distribution of points in the image. In addition, the mapping of a single class label back to the image is only a gross representation of feature space information, which loses the relationship of each point to other classes in feature space. These observations suggest that if we want to obtain better segmentation performance, we must make use not only of similarities among the pixels, but also of their relative positions.

In this chapter, we will use the probabilistic relaxation method as a means to enforce the spatial constraints into our segmentation algorithm. An introduction to

probabilistic relaxation is given in section 4.1. In this section we also review some past work that uses relaxation in classifying pixels. Section 4.2 discusses how we estimate our initial probabilistic labeling from the prototypes. In section 4.3, two probabilistic relaxation algorithms used to update the labeling probabilities are included. Section 4.4 presents the supervised segmentation simulation results.

4.1 Probabilistic Relaxation Method in Pixel Classification

The idea of effective use of cooperative processing through the mechanism of probabilistic relaxation was first introduced by Rosenfeld *et al.* [45]. Probabilistic relaxation is an iterative approach for using contextual information to reduce local ambiguities. Let us first review some of the concepts involved in probabilistic relaxation.

4.1.1 Introduction

The probabilistic relaxation process involves a set of objects $A = \{a_1, a_2, \cdots, a_n\}$ and a set of class labels $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_m\}$. In pixel classification problems, the set of objects in general is the whole set of pixels in the image represented in lexicographic notation. For each object a_i we are given a set of local measurements, which are used as a basis for estimating the probabilities $P_i(\lambda)$ of object a_i having each label λ. These probabilities satisfy the condition

$$\sum_{\lambda \epsilon \Lambda} P_i(\lambda) = 1, \quad \text{for all } a_i \epsilon A, \quad \text{and} \quad 0 \leq P_i(\lambda) \leq 1\,. \tag{4.1}$$

Suppose that the class label assignments of the objects are interdependent; in

other words, for each pair of class label assignments $a_i \epsilon \lambda$ and $a_j \epsilon \lambda'$, we have some quantitative measure of the compatibility of this pair, denoted by $r_{ij}(\lambda, \lambda')$. There are many ways to choose and to interpret the compatibility coefficients. Peleg and Rosenfeld [46] suggested that compatibility coefficients may be either interpreted as correlations or mutual information. Zucker and Mohammed [47] suggested rewriting and modifying the relaxation in such a way that the compatibility coefficients had the meaning of conditional probabilities. In subsections 4.1.2 and 4.1.3 we will show previous work using different methods to derive the compatibility coefficients.

There are also many possible iteration schemes that can be used to update the labeling probabilities based upon initial probabilities and compatibility coefficients. The original relaxation iteration scheme developed by Rosenfeld *et al.* [45] is described in subsection 4.1.2. Peleg [48] developed an improved probabilistic relaxation scheme motivated by Bayes' formula. Subsection 4.1.3 gives a detailed discussion of Peleg's work on pixel classification.

Some theoretical studies of the convergence properties of relaxation schemes have been conducted in Haralick *et al.* [49] and Zucker *et al.* [47], but we will not treat them here. There are several guidelines for evaluating the relaxation algorithms' performance [1]:

- The sum of absolute probability differences $\sum_{i \epsilon A} |P_i^{(k)}(\lambda) - P_i^{(k+1)}(\lambda)|$ should become small after a few iterations.
- The final probabilities should not be too far away, on the average, from the initial ones; we would not be satisfied with the process if it converged to an arbitrary set of final probabilities unrelated to the initial ones. Thus $\sum_{i \epsilon A} |P_i^{(k)}(\lambda) - P_i^{(0)}(\lambda)|$

should not become very large.

- The entropy of the probabilistic classification after applying relaxation should be less than the entropy of the initial one. In other words, we expect

$$-\sum_{i\epsilon A} P_i^{(k)}(\lambda)\log P_i^{(k)}(\lambda) < -\sum_{i\epsilon A} P_i^{(0)}(\lambda)\log P_i^{(0)}(\lambda).$$

4.1.2 Rosenfeld-Hummel-Zucker's Work

Rosenfeld *et al.* [45] have emphasized the importance of graph labeling and proposed an approximate, iterative, parallel relaxation method as a solution to it. The probabilistic relaxation approach was then used by Eklundh *et al.* [50] in a multispectral pixel classification problem. Their experiment was conducted on a color picture of house. The picture was hand segmented into five regions, which were regarded as the ground truth when evaluating results. The means and covariance matrices for the five classes were then estimated. Initial estimates of the probabilities of membership in these classes were made by assuming the clusters to be normally distributed, and using the hand segmentation to define the prior probabilities $P(\lambda)$ of the classes. Because they assumed a normal distribution, i.e., the class conditional density function may be written as

$$p(\vec{x}|\lambda) = \frac{1}{(2\pi)^{1/2}|\Sigma_\lambda|^{1/2}} \exp(-\frac{1}{2}(\vec{x}-\vec{\mu}_\lambda)^t{\Sigma_\lambda}^{-1}(\vec{x}-\vec{\mu}_\lambda)) \,. \tag{4.2}$$

The class probabilities were assigned to each pixel according to the formula:

$$P_i(\lambda|\vec{x}) = \frac{p(\vec{x}|\lambda)P(\lambda)}{\sum_{\lambda'} p(\vec{x}|\lambda')P(\lambda')} \,. \tag{4.3}$$

These probabilities were used as the initial probabilities $P_i^{(0)}(\lambda) = P_i(\lambda|\vec{x})$.

They defined the compatibility coefficients based on the mutual information of the labels at neighboring points. The probability of any point having the label λ may be estimated by

$$\hat{P}(\lambda) = \frac{1}{n} \sum_i P_i^{(0)}(\lambda) , \tag{4.4}$$

and the joint probability of a pair of points having labels λ and λ' by

$$\hat{P}_{i,i+\delta}(\lambda, \lambda') = \frac{1}{n} \sum_i P_i^{(0)}(\lambda) P_{i+\delta}^{(0)}(\lambda') \quad \delta \epsilon \Delta , \tag{4.5}$$

where n is the number of points, pixel $i + \delta$ is a specific neighbor of pixel i, and Δ defines the neighborhood about the particular pixel being considered. An estimate of the mutual information that the contribution of λ' to the information about λ is expressed as

$$I_{i,i+\delta}(\lambda, \lambda') = \ln \frac{\hat{P}_{i,i+\delta}(\lambda, \lambda')}{\hat{P}(\lambda)\hat{P}(\lambda')} \quad \delta \epsilon \Delta . \tag{4.6}$$

Assume the events that cause

$$\frac{P_{i,i+\delta}(\lambda, \lambda')}{\hat{P}(\lambda)\hat{P}(\lambda')} \quad \delta \epsilon \Delta , \tag{4.7}$$

to be outside the range $[e^{-5}, e^5]$ can be ignored. Thus values of $I_{i,i+\delta}(\lambda, \lambda')$ can be considered to lie in the range $[-5, 5]$ and the compatibility coefficient is defined as

$$r_{i,i+\delta}(\lambda, \lambda') = \frac{1}{5} I_{i,i+\delta}(\lambda, \lambda') \quad \delta \epsilon \Delta , \tag{4.8}$$

which lie in the range $[-1, 1]$. The compatibility coefficients are fixed throughout the process.

The compatibility coefficients can then be used to compute the updating factor

$$q_i^{(k)}(\lambda) = \sum_{\delta \epsilon \Delta} d_{i+\delta} \sum_{\lambda'} r_{i,i+\delta}(\lambda, \lambda') P_{i+\delta}^{(k)}(\lambda') , \tag{4.9}$$

where λ' is a dummy variable of summation, $d_{i+\delta}$ are a set of neighbor weights that can be used to give different neighbors differing degrees of influence in the neighborhood function. Eklundh *et al.* chose all neighbor weights as a constant of 1 and Δ as the 8-connected neighbors plus the central pixel (i.e. that under consideration). The relaxation iterations have the form

$$P_i^{(k+1)}(\lambda) = \frac{P_i^{(k)}(\lambda)[1 + q_i^{(k)}(\lambda)]}{\sum_{\lambda'} P_i^{(k)}(\lambda')[1 + q_i^{(k)}(\lambda')]} \,. \tag{4.10}$$

In their experiment they iterated until the classification error was a minimum.

4.1.3 Peleg's Work

Peleg [48] suggested that the compatibility coefficients $r_{i,i+\delta}(\lambda, \lambda')$ should take the form

$$r_{i,i+\delta}(\lambda, \lambda') = \frac{\hat{P}_{i,i+\delta}(\lambda, \lambda')}{\hat{P}(\lambda)\hat{P}(\lambda')} \quad \delta \epsilon \Delta \,, \tag{4.11}$$

and that a vector of probabilities be associated with every pixel. These probabilities define a random variable representing the possible labels of the pixel. Probabilities at neighboring pixels are used iteratively to update the probabilities at a given pixel based on statistical relations among pixel labels. The advantage of this method is that since the updating rule is analytically derived, all coefficients are defined, eliminating the need to guess them. Peleg applied his relaxation method to a problem involving pixel classification on the basis of color. Since it is closely related to our textured image segmentation problem, a more detailed discussion is provided.

Let $P_i^{(0)}(\lambda)$ specify the initial probability estimates assigned to the classes at pixel a_i, based on the clustering results in color space. These probability estimates are then updated. Let $\{a_1, a_2, \ldots, a_n\}$ be the n pixels in the picture, and let Λ be the

set of possible labels for each pixel. We assume *a priori* uniformity over the picture, so that the *a priori* probabilities $P(\lambda)$ are the same for all pixels. The *a priori* joint probabilities $P_{i,i+\delta}(\lambda, \lambda')$ also do not depend on the specific pixels, but only on the relation between them. Given these uniformity assumptions, it is simple to compute the coefficients $r_{i,i+\delta}(\lambda, \lambda')$ for the process from the initial probability assignment. The *a priori* probabilities $P(\lambda)$ are estimated by

$$\hat{P}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} P_i^{(0)}(\lambda) . \tag{4.12}$$

The estimates for the joint probabilities are

$$\hat{P}_{i,i+\delta}(\lambda, \lambda') = \frac{1}{n} \sum_{i} P_i^{(0)}(\lambda) P_{i+\delta}^{(0)}(\lambda') \quad \delta \epsilon \Delta , \tag{4.13}$$

where n is the number of points, pixel $i + \delta$ is a specific neighbor of pixel i, and again Δ defines the neighborhood about the particular pixel being considered. The compatibility coefficients are then defined as

$$r_{i,i+\delta}(\lambda, \lambda') = \frac{\hat{P}_{i,i+\delta}(\lambda, \lambda')}{\hat{P}(\lambda)\hat{P}(\lambda')} \quad \delta \epsilon \Delta . \tag{4.14}$$

After finding the coefficients, the updating process can take place. For a specific $i + \delta$ we compute

$$S_{i,i+\delta}^{(k+1)}(\lambda) = \sum_{\lambda' \epsilon \Lambda} \hat{P}_i^{(k)}(\lambda) \hat{P}_{i+\delta}^{(k)}(\lambda') r_{i,i+\delta}(\lambda, \lambda') \quad \delta \epsilon \Delta , \tag{4.15}$$

and

$$q_{i,i+\delta}^{(k+1)}(\lambda) = \frac{S_{i,i+\delta}^{(k+1)}(\lambda)}{\sum_{\lambda' \epsilon \Lambda} S_{i,i+\delta}^{(k+1)}(\lambda')} \quad \delta \epsilon \Delta , \tag{4.16}$$

here $q_{i,i+\delta}^{(k+1)}$ is the new probability estimate for the labels at pixel a_i based upon the previous estimates at a_i and $a_{i+\delta}$. For the case of using eight connected neighbors, the

new $P_i^{(k+1)}$ will be the average of these eight estimates:

$$P_i^{(k+1)}(\lambda) = \frac{1}{8}\sum_{\delta\epsilon\Delta} q_{i,i+\delta}^{(k+1)}(\lambda) \quad \lambda\epsilon\Lambda\,. \tag{4.17}$$

The final pixel classification is the maximum-probability class for each pixel.

In the next three sections we will discuss how to apply the probabilistic relaxation method to our supervised textured image segmentation problem. To the best of the author's knowledge, no previous attempts have been made in using probabilistic relaxation to solve the textured image segmentation problem.

4.2 Initial Probabilistic Labeling

Assume that a prototype of each texture under test is given, we may use the method described in chapter 3 to generate the texture energy features. We define the mean vector $\vec{\mu}_{\lambda_k}$ and covariance matrix Σ_{λ_k} of class λ_k according to

$$\vec{\mu}_{\lambda_k} = \begin{bmatrix} \mu_{k1} \\ \mu_{k2} \\ \vdots \\ \mu_{kn} \end{bmatrix} \qquad \Sigma_{\lambda_k} = \begin{bmatrix} \sigma_{k11} & \sigma_{k12} & \cdots & \sigma_{k1n} \\ \sigma_{k21} & \sigma_{k22} & \cdots & \sigma_{k2n} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{kn1} & \sigma_{kn2} & \cdots & \sigma_{knn} \end{bmatrix} \tag{4.18}$$

We also assume that the distribution of n texture energy features has the form of an n-dimensional multivariate Gaussian density, then the probability density function can be written as

$$p(\vec{x}|\lambda_k) = \frac{1}{(2\pi)^{1/2}|\Sigma_{\lambda_k}|^{1/2}} \exp(-\frac{1}{2}(\vec{x} - \vec{\mu}_{\lambda_k})^t {\Sigma_{\lambda_k}}^{-1}(\vec{x} - \vec{\mu}_{\lambda_k}))\,. \tag{4.19}$$

In practice, the mean vector and covariance matrix for each class are unknown and must be estimated from prototypes. Let $\hat{\mu}_{\lambda_k}$ and $\hat{\Sigma}_{\lambda_k}$ be unbiased estimates of $\vec{\mu}_{\lambda_k}$ and

Σ_{λ_k}. Then $\hat{\mu}_{\lambda_k}$ and $\hat{\Sigma}_{\lambda_k}$ are given by

$$\hat{\mu}_{kj} = \frac{1}{n_k}\sum_{l=1}^{n_k} x_{jl} \quad j = 1, 2, \ldots, n\,, \tag{4.20}$$

$$\hat{\sigma}_{kjm} = \frac{1}{n_k - 1}\sum_{l=1}^{n_k}(x_{jl} - \hat{\mu}_{kj})(x_{ml} - \hat{\mu}_{km}) \quad j = 1, 2, \ldots, n; \quad m = 1, 2, \ldots, n\,, \tag{4.21}$$

where n_k is the number of prototype samples in class λ_k.

Assuming the *a priori* probability of each class $P(\lambda_k)$ is equal and the class conditional probability density function $p(\vec{x}|\lambda_k)$ is estimated as previously described, then the *a posteriori* probability of assigning a pixel with feature vector $\vec{x}$ to class λ_k is given by

$$P_i(\lambda_k|\vec{x}) = \frac{p(\vec{x}|\lambda_k)P(\lambda_k)}{\sum_{\lambda_j \epsilon \Lambda} p(\vec{x}|\lambda_j)P(\lambda_j)}\,. \tag{4.22}$$

These *a posteriori* probabilities of each pixel may be used as the initial probability labeling $P_i^{(0)}(\lambda_k)$ and compatibility coefficients may then be calculated.

4.3 Two Relaxation Iteration Algorithms

Once the initial labeling probabilities of each pixel are generated, there are several ways to iteratively update them. Two algorithms are chosen here for detailed study: Rosenfeld-Hummel-Zucker (RHZ) and Peleg schemes. In our study we assume the set of neighbors of a pixel to be the 8-connected neighbors.

The RHZ probabilistic relaxation algorithm may be summarized as follows:

1. The mutual information coefficients are chosen to be the compatibility coefficients. For each one of the 8-connected neighbors we use Eqs. (4.4)-(4.8) to calculate the compatibility coefficients. The coefficients $r_{i,i+\delta}(\lambda, \lambda')$ are fixed throughout the iteration process.

2. With the initial labeling probabilities of each pixel and the compatibility coefficients on hand, we may start the iteration by first computing the updating factor

$$q_i^{(k)}(\lambda) = \sum_{\delta \epsilon \Delta} \frac{1}{8} \sum_{\lambda'} r_{i,i+\delta}(\lambda, \lambda') P_{i+\delta}^{(k)}(\lambda') \quad k = 0, \cdots, \tag{4.23}$$

where Δ is the set of 8-connected neighbors of the pixel under consideration.

3. The updating factor is then used to compute $P_i^{(k+1)}(\lambda)$ according to Eq. (4.10).

4. Repeat steps (2)–(3) until the stopping criteria is met. In our case we specified the number of iterations in advance.

In next section we will present the simulation results of applying the RHZ relaxation algorithm to our textured image segmentation problem.

Similarly, the Peleg probabilistic relaxation algorithm is summarized as follows:

1. For each one of the 8-connected neighbors we use Eqs. (4.12)–(4.14) to calculate the compatibility coefficients. The coefficients $r_{i,i+\delta}(\lambda, \lambda')$ are fixed throughout the iteration process.

2. The iterative updating process follows by using Eqs. (4.15)–(4.16). The $q_{i,i+\delta}^{(k+1)}(\lambda)$ is the pairwise effect of one of the 8-connected neighborhood.

3. Since every pixel has eight neighbors, the new $P_i^{(k+1)}(\lambda)$ is the average of these eight estimates as in Eq. (4.17).

4. Repeat steps (2)–(3) until the stopping criteria is met. In our case we specified the number of iterations in advance.

Simulation results of Peleg algorithm will also be presented in next section.

4.4 Simulation Results

The first test image is a texture mosaic (Figure 4.1(a)) which consists of four different textures: grass, water, pigskin, and leather. All four textures are present in the image in squares of size 128×128. Figure 4.1(b) is the gray level coded ground truth of Figure 4.1(a).

We still use E5L5, E5S5, L5S5, and R5R5 as our four micro-texture masks. The texture energy features are extracted by the method described in chapter 3. After the features are extracted, the mean vector and covariance matrix of each class are estimated by Eqs. (4.20)–(4.21). Table 4.1 shows the estimated mean vectors and covariance matrices of four texture classes.

The initial labeling probabilities for each pixel are calculated by Eq. (4.22). Figure 4.2(a) shows the initial probability class labels for each pixel. The white pixels in Figure 4.2(b) represent misclassified pixels, and the overall error rate is 3.12 percent.

We first apply the RIIZ relaxation algorithm to update the probabilistic classification. Table 4.2 shows the mutual information coefficients obtained from the initial classification. The row index is the class of the center pixel, and column index is the class of the neighbor. The neighbor numbering convention in our work is as follows:

$$\begin{matrix} 1 & 2 & 3 \\ 4 & i & 5 \\ 6 & 7 & 8 \end{matrix}.$$

Figure 4.3(a) shows the classification result of 50 iterations using the mutual information coefficients as the compatibility coefficients. Figure 4.3(b) is the corresponding misclassified pixels. The error rate went down from 3.12 percent to 2.4 percent as shown in Figure 4.5.

Class 1:

$$\hat{\mu}_1 = \begin{bmatrix} 2135.25 \\ 336.27 \\ 1357.57 \\ 346.95 \end{bmatrix} \quad \hat{\Sigma}_1 = \begin{bmatrix} 118030.70 & 3502.57 & -13461.62 & -6747.39 \\ 3502.57 & 1088.82 & 940.45 & -146.85 \\ -13461.62 & 940.45 & 26987.62 & 2764.14 \\ -6747.39 & -146.85 & 2764.14 & 2263.66 \end{bmatrix}$$

Class 2:

$$\hat{\mu}_2 = \begin{bmatrix} 1870.31 \\ 359.87 \\ 1568.77 \\ 910.32 \end{bmatrix} \quad \hat{\Sigma}_2 = \begin{bmatrix} 53143.55 & 537.39 & -11388.94 & -87.65 \\ 537.39 & 1476.77 & 552.65 & 976.38 \\ -11388.94 & 552.65 & 44351.40 & -1877.91 \\ -87.65 & 976.38 & -1877.91 & 8673.87 \end{bmatrix}$$

Class 3:

$$\hat{\mu}_3 = \begin{bmatrix} 2260.11 \\ 314.14 \\ 1098.20 \\ 831.98 \end{bmatrix} \quad \hat{\Sigma}_3 = \begin{bmatrix} 70611.10 & 3649.78 & 5137.45 & -2067.59 \\ 3649.78 & 1164.97 & 1084.15 & 236.87 \\ 5137.45 & 1084.15 & 13935.00 & 977.75 \\ -2067.59 & 236.87 & 977.75 & 12565.73 \end{bmatrix}$$

Class 4:

$$\hat{\mu}_4 = \begin{bmatrix} 1239.23 \\ 243.43 \\ 2244.91 \\ 566.13 \end{bmatrix} \quad \hat{\Sigma}_4 = \begin{bmatrix} 26088.67 & 1376.89 & -20402.53 & -1984.53 \\ 1376.89 & 879.86 & -4279.33 & -686.43 \\ -20402.53 & -4279.33 & 247810.80 & 11116.73 \\ -1984.53 & -686.43 & 11116.73 & 5619.75 \end{bmatrix}$$

Table 4.1: Mean Vectors and Covariance Matrices for Four Texture Classes.

Direction 1	0.247	−0.733	−0.389	−0.699
	−0.574	0.315	−0.587	−0.586
	−0.431	−0.518	0.247	−0.776
	−0.901	−0.494	−0.808	0.271
Direction 2	0.249	−0.735	−0.446	−0.702
	−0.681	0.317	−0.661	−0.582
	−0.442	−0.547	0.248	−0.773
	−0.971	−0.594	−0.848	0.273
Direction 3	0.248	−0.583	−0.444	−0.686
	−0.668	0.315	−0.626	−0.475
	−0.394	−0.509	0.246	−0.743
	−0.953	−0.583	−0.859	0.273
Direction 4	0.249	−0.749	−0.405	−0.852
	−0.609	0.317	−0.619	−0.618
	−0.454	−0.649	0.250	−0.869
	−0.834	−0.502	−0.816	0.271
Direction 5	0.249	−0.596	−0.456	−0.829
	−0.749	0.316	−0.657	−0.495
	−0.406	−0.624	0.248	−0.815
	−0.848	−0.607	−0.870	0.273
Direction 6	0.248	−0.678	−0.391	−0.958
	−0.598	0.314	−0.508	−0.596
	−0.436	−0.620	0.249	−0.857
	−0.690	−0.483	−0.742	0.269
Direction 7	0.250	−0.691	−0.440	−0.971
	−0.740	0.316	−0.550	−0.597
	−0.440	−0.662	0.249	−0.846
	−0.702	−0.583	−0.771	0.271
Direction 8	0.248	−0.568	−0.431	−0.894
	−0.740	0.314	−0.526	−0.489
	−0.385	−0.595	0.247	−0.804
	−0.696	−0.579	−0.772	0.271

Table 4.2: Mutual Information Coefficients as Compatibility Coefficients.

Similarly, we applied Peleg's relaxation scheme to our problem using the same initial labeling probabilities. Table 4.3 shows the compatibility coefficients obtained from the initial classification. The neighbor numbering convention is the same as before. Figure 4.4(a) shows the classification result of 50 iterations using Peleg's scheme. Figure 4.4(b) shows the corresponding misclassified pixels. The error rate went down from 3.12 percent to 2.06 percent as shown in Figure 4.5.

From Figure 4.5 we can clear see that the Peleg's scheme converges much faster than RHZ's scheme. In addition, both methods exhibit the desirable fact that most of the error reduction occurs in the first 50 iterations or so and then the error decreases slowly. The effectiveness of iterative relaxation is demonstrated in Figure 4.4(a), which shows that most isolated small islands in the initial classification are eliminated.

The second test image is a texture mosaic which consists of five different textures: grass, water, pigskin, leather, and raffia (Figure 4.6(a)). The fifth texture class, raffia, is presented in the middle of the image as a tilted ellipse. From the second image we can test how well the square quadrant window shape performs on different region shapes. Figure 4.6(b) is the gray level coded ground truth of Figure 4.6(a).

The initial classification based on labeling pixels with the maximum initial probability is shown in Figure 4.7(a). The white pixels in Figure 4.7(b) indicate the misclassified ones, the error rate is 4.93 percent. The square quadrant window shape works very well on the middle region whose shape is a tilted ellipse. This result gives us an indication that as long as the region size is much larger than the size of quadrant window then the square quadrant window shape works fairly well. Figure 4.8(a) shows the classification result of applying 25 iterations of Peleg's relaxation scheme. Figure 4.8(b)

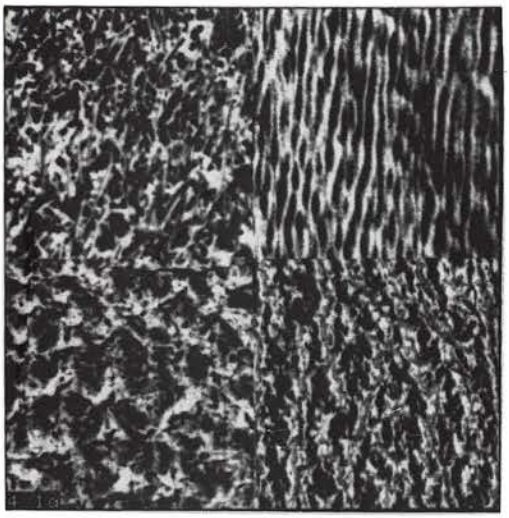
Direction 1	3.4377	0.0256	0.1432	0.0303
	0.0568	4.8341	0.0530	0.0532
	0.1157	0.0750	3.4448	0.0207
	0.0111	0.0846	0.0176	3.8827
Direction 2	3.4738	0.0254	0.1076	0.0298
	0.0333	4.8877	0.0368	0.0545
	0.1095	0.0648	3.4580	0.0210
	0.0078	0.0512	0.0144	3.9161
Direction 3	3.4499	0.0543	0.1084	0.0324
	0.0355	4.8259	0.0437	0.0931
	0.1392	0.0785	3.4154	0.0244
	0.0085	0.0541	0.0136	3.9138
Direction 4	3.4652	0.0236	0.1319	0.0141
	0.0474	4.8677	0.0452	0.0455
	0.1031	0.0388	3.4905	0.0129
	0.0153	0.0812	0.0169	3.8814
Direction 5	3.4739	0.0506	0.1022	0.0158
	0.0237	4.8629	0.0374	0.0840
	0.1313	0.0441	3.4547	0.0169
	0.0144	0.0480	0.0129	3.9131
Direction 6	3.4535	0.0337	0.1417	0.0083
	0.0502	4.8106	0.0787	0.0508
	0.1126	0.0451	3.4757	0.0138
	0.0317	0.0893	0.0245	3.8487
Direction 7	3.4865	0.0315	0.1106	0.0078
	0.0247	4.8669	0.0639	0.0506
	0.1107	0.0364	3.4832	0.0145
	0.0299	0.0541	0.0211	3.8823
Direction 8	3.4591	0.0583	0.1156	0.0114
	0.0247	4.8099	0.0721	0.0866
	0.1459	0.0509	3.4344	0.0179
	0.0308	0.0553	0.0210	3.8804

Table 4.3: Peleg Compatibility Coefficients.

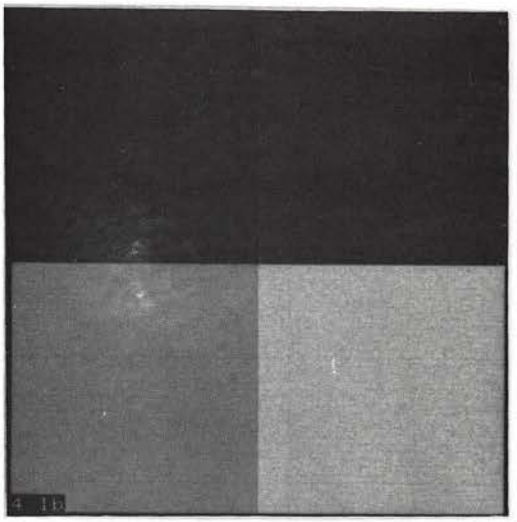
is the corresponding misclassified pixels. The error rate went down from 4.93 percent to 3.5 percent.

As a comparison, we also process the second test image using Laws' feature extraction method as described in chapter 3. The window size used to estimate macro statistical features is 15×15. Figure 4.9(a) shows the initial classification result with an error rate of 7.2 percent. After 25 iterations, the classification result is shown in Figure 4.9(b). The error rate went down from 7.2 percent to 5.4 percent. From this comparison, we can clearly see the performance is improved by using our proposed method.

In summary, the following observations can be made from the simulation results. First, probabilistic relaxation methods are an effective means to incorporate spatial constraints into segmentation algorithms. Second, probabilistic relaxation provides most of the error reduction in the first 25 iterations or so, after that the error decreases slowly. Third, Peleg's scheme is usually converges much faster than RHZ's scheme. Finally, the square quadrant window shape works fairly well on regions with arbitrary shape provided that the region size is much larger than the size of the quadrant window.

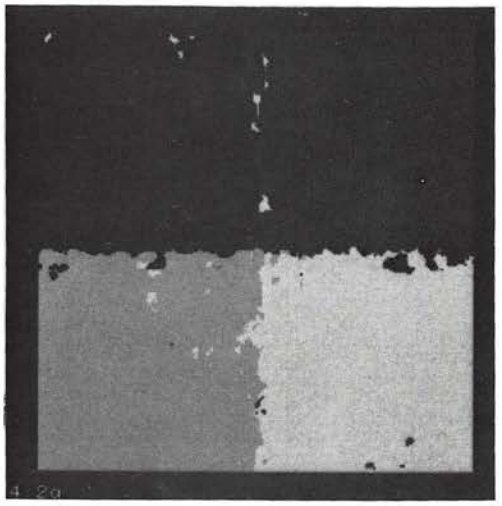


(a) Texture Mosaic

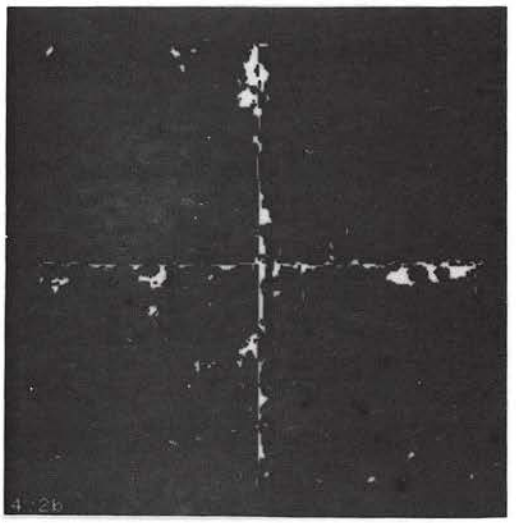


(b) Ground Truth

Figure 4.1: Texture Mosaic 1 for Experimental Work.

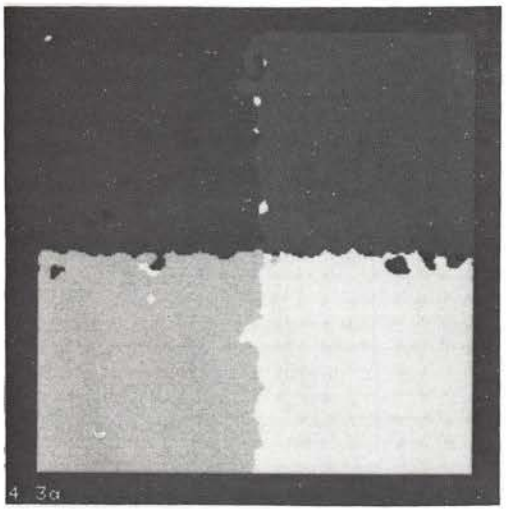


(a) Gray Level Coded Result

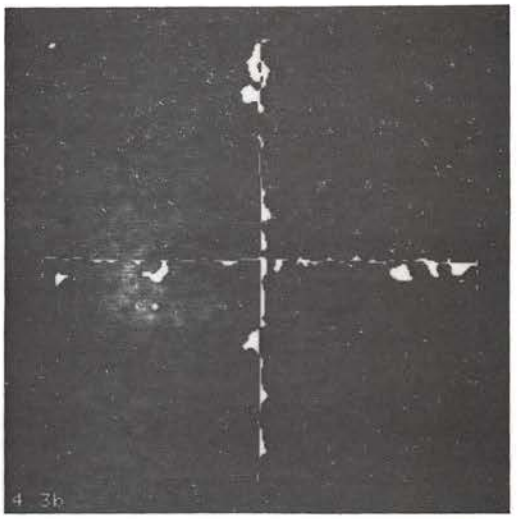


(b) Misclassified Pixels in White

Figure 4.2: Initial Probabilistic Classification Results.

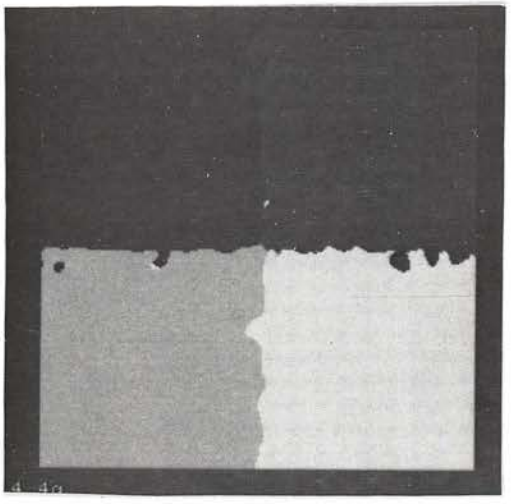


(a) Gray Level Coded Result

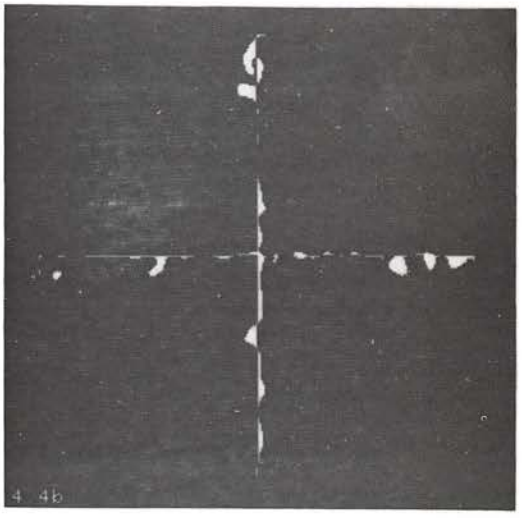


(b) Misclassified Pixels in White

Figure 4.3: Classification Result After Applying RHZ's Relaxation 50 Iterations.



(a) Gray Level Coded Result



(b) Misclassified Pixels in White

Figure 4.4: Classification Result After Applying Peleg's Relaxation 50 Iterations.

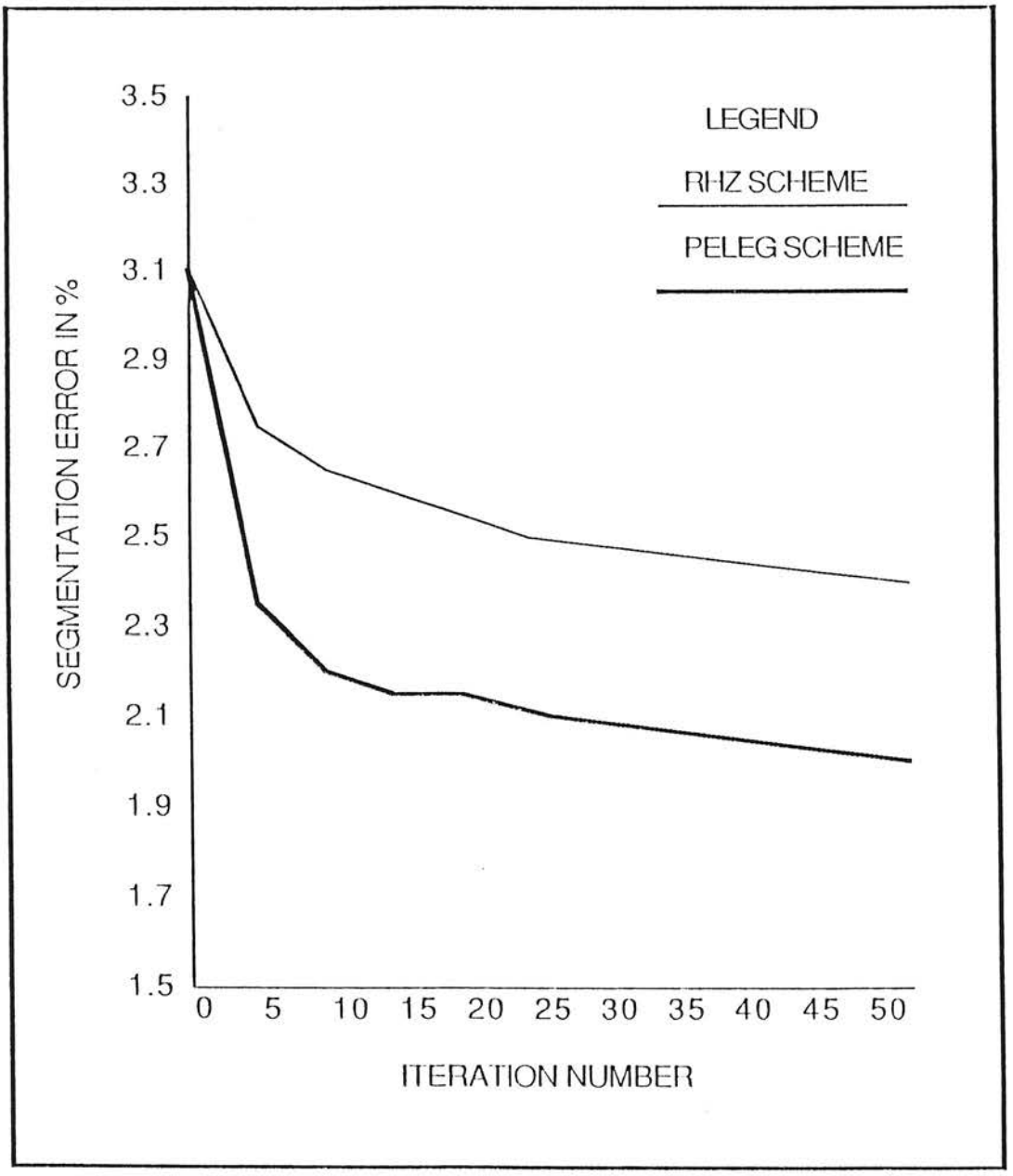
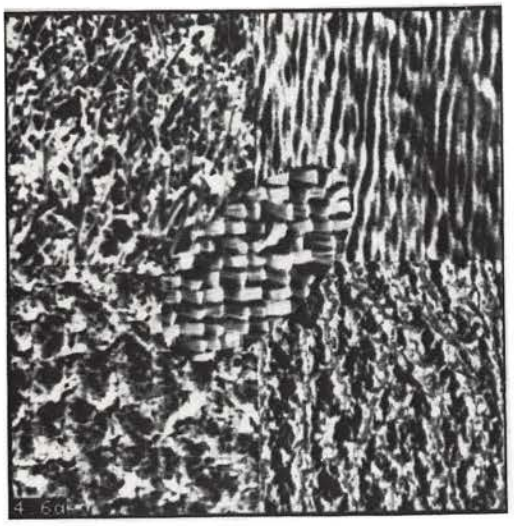
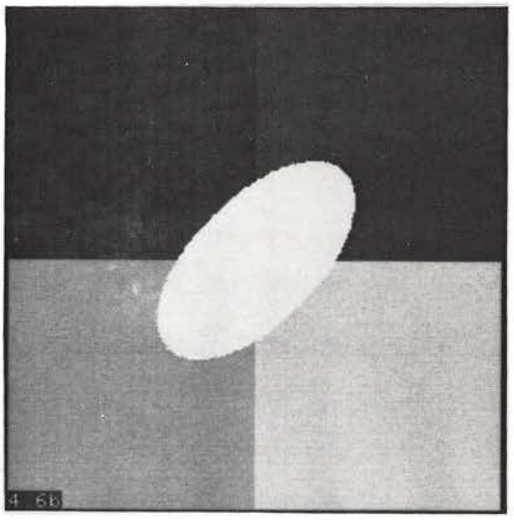


Figure 4.5: The Error Rates of RHZ and Peleg Scheme.

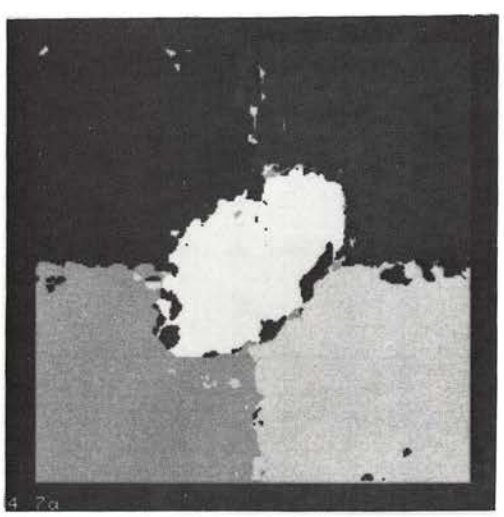


(a) Texture Mosaic

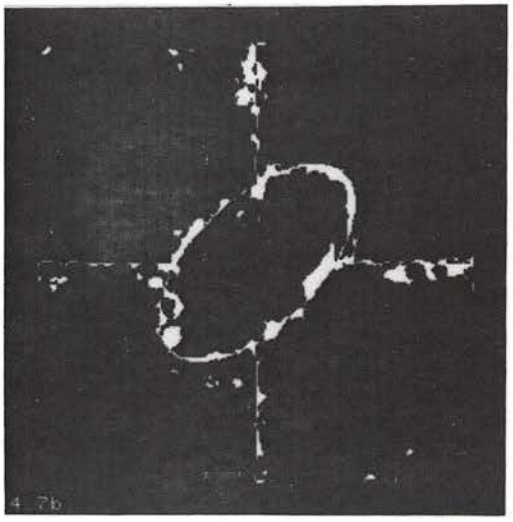


(b) Ground Truth

Figure 4.6: Texture Mosaic 2 for Experimental Work.

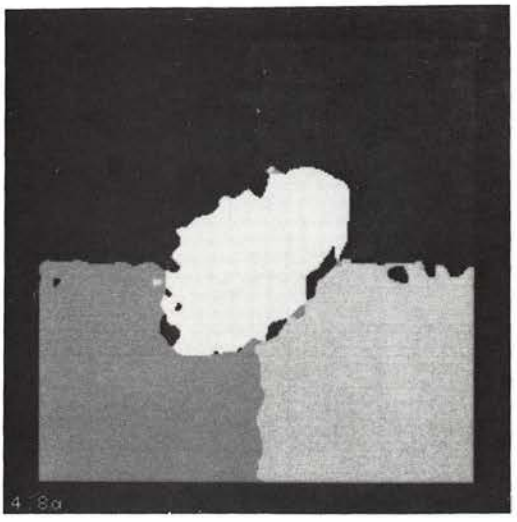


(a) Gray Level Coded Result



(b) Misclassified Pixels in White

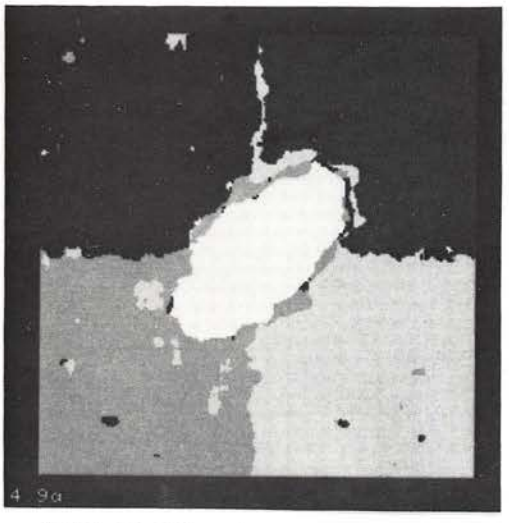
Figure 4.7: Initial Probabilistic Classification Results of Mosaic 2.



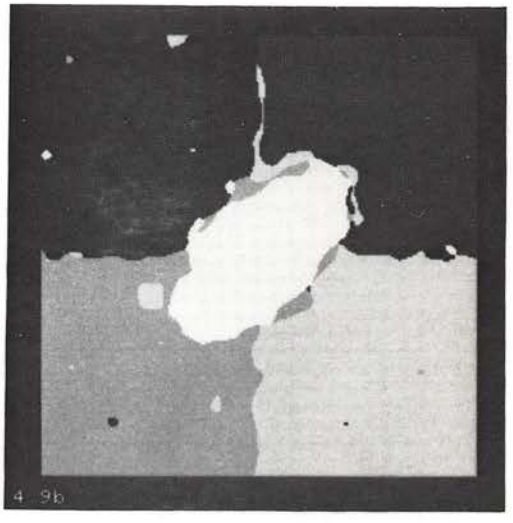
(a) Gray Level Coded Result

(b) Misclassified Pixels in White

Figure 4.8: Classification Result After Applying Peleg's Relaxation 25 Iterations.



(a) Initial Result



(b) After 25 Iterations

Figure 4.9: Classification Results of Mosaic 2 Using Laws' Method.

Chapter 5

Unsupervised Segmentation by Clustering

What we have discussed so far has been supervised segmentation; that is, there is a supervisor that first teaches the system using a known set of prototypes; then the system classifies new unknown data. In such systems we need *a priori* information to form the basis of teaching. However, in many classification problems, there is little prior information available about the data and the decision-maker wishes to make as few assumptions about the data as possible. This restricts one to studying the interrelationships among the data points to make a preliminary assessment of their structure. Cluster analysis is one tool that attempts to assess the interaction among patterns by organizing the patterns into groups or clusters such that patterns within a cluster are more similar to each other than are patterns belonging to different clusters.

In this chapter we describe a procedure for segmenting textured images without the need for training prototypes. Section 5.1 discusses the issues involved in using clustering algorithms. A specific clustering algorithm for our textured image segmentation

problem is described in section 5.2. Section 5.3 presents some simulation results.

5.1 Introduction

The process of clustering can be stated as: find the regions $R_1, R_2, \cdots, R_M$ such that every $\vec{x}_i$, $i = 1, 2, \cdots, n$, falls into one of these regions and no $\vec{x}_i$ falls in two regions. The $\vec{x}_i$ is the p-dimensional feature vector measured on the pattern. There are a large number of clustering algorithms that have been suggested. They can be broadly classified into one of two types: hierarchical or partitional. A hierarchical clustering technique imposes a hierarchical structure on the data consisting of a sequence of clusterings. Usually, the hierarchical clustering techniques expect that the data are available in the form of a proximity matrix. A proximity matrix is an $n \times n$ matrix whose rows and columns both represent patterns and whose entries measures proximity (similarity or dissimilarity) between all pairs of patterns. The hierarchical clustering techniques can be divided into two distinct classes, agglomerative and divisive. Agglomerative techniques start with n singleton clusters and form the sequence by successively merging clusters. Divisive techniques start with all of the samples in one cluster and form the sequence by successively splitting clusters.

A partitional clustering technique organizes the patterns into a small number of clusters by labeling each pattern in some way. Unlike hierarchical techniques, which give a sequence of partitions, a partitional clustering technique gives a single partition. A pattern matrix, an $n \times p$ matrix where each row is a pattern and each column denotes a feature, is usually clustered in this way. The partitional clustering techniques may be used with much larger problems than hierarchical techniques because it is not necessary

to calculate and store the proximity matrix. Due to the large amount of data in images, we use the partitional technique and discuss issues related to this type of clustering in the next few subsections.

5.1.1 Initial Configurations

Anderberg [51] reviewed a variety of techniques which may be used to establish an initial partition of the patterns. Some techniques used a set of M seed points as cluster nuclei around which the set of n patterns can be grouped. The following methods have been used to generate seed points:

1. Choose the first M patterns in the data set as the initial seed points.
2. Label the patterns from 1 to n and choose the seed points such that they are labeled n/M, $2n/M$, $\cdots$, $(M-1)n/M$, and n.
3. Subjectively choose any M patterns from the data set.
4. Label the patterns from 1 to n and choose the patterns corresponding to M different random numbers in the range 1 to n.
5. Generate M synthetic points as vectors of coordinates where each coordinate is a random number from the range of the associated variable.
6. Take any desired partition of the patterns into M mutually exclusive groups and compute the group centroids as seed points.
7. Choose seed points which span the data set, that is, most patterns are relatively close to a seed point but the seed points are well separated from each other.

8. Take the overall mean vector of the data set as the first seed point; select subsequent seed points by examining the patterns in their input sequence and accept any pattern which is at least some specified distance, say d, from all previously chosen seed points; continue choosing points until M seed points are accumulated or the data set is exhausted.

There are some clustering methods that use an initial partition of the patterns rather than a set of seed points to begin with. The following methods have been used to generate such partitions:

1. For a given set of seed points, assign each pattern to the cluster built around the nearest seed point. The seed point remain stationary throughout the assignment of the full data set.

2. Given a set of seed points, let each seed point initially be a cluster of one member; then assign patterns one at a time to the cluster with the nearest centroid; after a pattern is assigned to a cluster, update the centroid so that it is the true mean vector for all the patterns currently in that cluster.

3. Using a hierarchical clustering method on one or more subsets of patterns to generate an initial partition.

4. Various random allocation schemes could be used to generate an initial partition.

5. The initial partition could be generated by the analyst using his judgement.

5.1.2 Normalization

Under some situations, our usual concepts of distance may bear no relation to the problem. For instance, if two features are measured in units different from each other and one has much larger numerical values than the other. This will cause the feature with large numerical values dominating the distance calculations. One remedy to this problem is to normalize the data prior to calculating the distance.

One normalization method is to find the range of values for each feature and then normalize the feature by the range. This method considers only the extreme values; an alternative approach is to use the standard deviations of the features as the normalizing factor. Let σ_k be the standard deviation of the kth feature then the normalized variables are x_k/σ_k. The standard deviation of the kth feature can be estimated by first estimating the variance $\hat{\sigma}_k^2$

$$\hat{\sigma}_k^2 = \frac{1}{n-1}\sum_{l=1}^{n}(x_{kl} - \hat{\mu}_k)^2$$

where $\hat{\mu}_k$ is the estimated mean of the kth feature and n is the number of prototype samples.

5.1.3 Similarity Measures

The similarity measure (or dissimilarity measure) is needed in the process of finding natural groupings in a set of data, and is usually given in numerical form. The simplest and most frequently used measure is squared Euclidean distance,

$$d_E^2(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^t(\vec{x}_i - \vec{x}_j) = |\vec{x}_i - \vec{x}_j|^2 \,, \tag{5.1}$$

in multidimensional Euclidean space.

An alternative to normalizing the data and using Euclidean distance is to use some kind of normalized distance, such as the Mahalanobis distance [52]. The squared Mahalanobis distance from $\vec{x}_i$ to $\vec{x}_j$ is in the form

$$d_M^2(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^t \Sigma^{-1} (\vec{x}_i - \vec{x}_j) \,, \tag{5.2}$$

where Σ^{-1} is the inverse of the covariance matrix.

The similarity measure may also needed in describing the subregions corresponding to clusters. For instance, a cluster can be defined by defining a set of centers $\vec{\mu}_1, \vec{\mu}_2, \cdots, \vec{\mu}_M$ and a measure of similarity $d(\vec{x}, \vec{\mu}_i)$. A cluster is then the set of points which are nearer to particular cluster center, that is

$$R_i = \{\vec{x} | d(\vec{x}, \vec{\mu}_i) \leq d(\vec{x}, \vec{\mu}_j) \text{ for all } j \neq i\} \,. \tag{5.3}$$

One similarity measure in which we are interested is the weighted Euclidean distance

$$d^2(\vec{x}, \vec{\mu}_i) = \sum_{j=1}^{n} (\frac{x_j - \mu_{ij}}{\sigma_{ij}})^2 \,. \tag{5.4}$$

The clusters are described by a set of M mean vectors and a set of M variance vectors.

5.1.4 Number of Clusters

The number of clusters can be assumed either known or unknown depending on how much knowledge about the data we have beforehand. When the number of clusters is unknown, various methods have been proposed to find the "correct" number of clusters. Ball and Hall [53] use merging and splitting to arrive at a final number of clusters. Thus clusters having variances that are larger than a threshold will be split and clusters whose means are separated by less than a threshold will be merged. The merging and splitting

thresholds must be established *a priori*. A procedure for determining these thresholds from the data has been developed by Fromm and Northouse [54].

Coleman and Andrews [28] proposed another approach that the intrinsic number of clusters is based upon a measure of the clustering quality represented by the parameter β. The parameter β is defined as

$$\beta = tr(S_b)\, tr(S_w)\ , \tag{5.5}$$

where $tr(*)$ indicates "trace" or sum of the diagonal elements of a matrix, S_b is the between-cluster scatter matrix, and S_w is the within-cluster scatter matrix. The between-cluster scatter matrix when the total number of clusters M is ≥ 2 is defined as

$$S_b = \frac{1}{M} \sum_{m=1}^{M} (\vec{\mu}_m - \vec{\mu}_0)(\vec{\mu}_m - \vec{\mu}_0)^t\ , \tag{5.6}$$

where $\vec{\mu}_0$ is the overall mean vector of the entire mixture, and is given by

$$\vec{\mu}_0 = \frac{1}{n} \sum_{i=1}^{n} \vec{x}_i\ , \tag{5.7}$$

where n represents the total number of feature vectors to be clustered. The within-cluster scatter matrix is based on the scatter of the data about the cluster means, and is given by

$$S_w = \frac{1}{M} \sum_{m=1}^{M} S_m\ , \tag{5.8}$$

where S_m is the scatter matrix for m-th cluster,

$$S_m = \frac{1}{n_m} \sum_{\vec{x}_i \epsilon R_m} (\vec{x}_i - \vec{\mu}_m)(\vec{x}_i - \vec{\mu}_m)^t\ , \tag{5.9}$$

where $\vec{\mu}_m$ is the mean of the m-th cluster, n_m is the number of elements in the m-th cluster, and $\vec{x}_i$ is an element in the m-th cluster. This parameter β passes through

a maximum at the intrinsic number of clusters. The K-means algorithm was used to generate $M = 2, 3, 4, \cdots, 16$ clusters. At each step, the β is computed, and the cluster number which causes the β to achieve the maximum value is then used as the intrinsic number of clusters.

When the number of clusters is known, a popular clustering method called K-means is often used to form the clusters. This method is based on the minimization of the sum of squared distances from all points in a cluster domain to the cluster center. There are many different algorithms to implement the K-means method, we use Forgy's algorithm to illustrate the basic idea. Forgy [55], [51] suggests a simple algorithm consisting of the following sequence of steps:

1. Begin with any desired initial configuration. Go to step 2 if beginning with a set of seed points; go to step 3 if beginning with a partition of the patterns.

2. Allocate each pattern to the cluster with the nearest seed point. The seed points remain fixed for a full cycle through the entire data set.

3. Compute new seed points as the centroids of the clusters of patterns.

4. Alternate steps 2 and 3 until the process converges; that is, continue until no patterns change their cluster membership at step 2.

The convergence properties of the K-means method have been studied by MacQueen [56]. The detailed proof of convergence is long and tedious. A very informative sketch of the proof which shows that the squared-error based clustering methods actually converge is given by Anderberg [51].

5.2 Estimation of Class Statistics by a Clustering Method

In this section we discuss a clustering algorithm that is used to estimate the class statistics from the unlabeled samples in the texture energy feature domain. The overall approach is depicted in the general flow chart of Figure 5.1. We assume that the texture energy features at each pixel have already been computed and the number of classes is known. If the number of classes is unknown, the method introduced by Coleman and Andrews [28] may be used to identify it.

In order to reduce the computation time, the clustering algorithm works on a subset of the data consisting of every fourth line and every fourth pixel of the feature planes. A global normalization was then performed on this subsampled data set. Each raw feature was normalized by its sample standard deviation to become a normalized feature.

The M initial seed points were chosen based upon Ball and Hall's approach, that is, the first seed point is the overall mean vector of the data set, and the subsequent seed points are selected by examining the patterns in their input sequence and only the pattern which is at least some specified distance, say d_{min}, from all previously chosen seed points are accepted. In our work the d_{min} is set to be one sample standard deviation of the data set.

An initial partition is obtained by assigning patterns to the closest initial seed point. The Euclidean distance is used for the similarity measure. The new cluster center is simply the average of all the patterns assigned to that cluster. The iteration

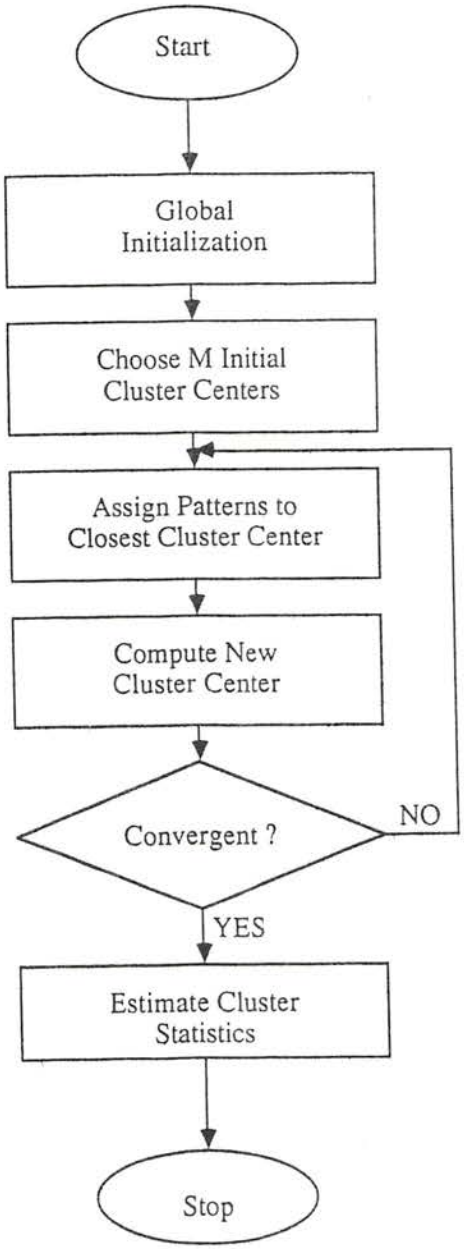


Figure 5.1: Flow Chart of the Overall Clustering Approach.

continues until either the process converges or the number of iterations exceeds a user-supplied limit. Convergence is assumed to have been reached when the difference of the means of all clusters from one iteration to the next is smaller than a prespecified value δ. In our experiment δ is set to 10^{-6}.

In our problem, the purpose of clustering is to learn the statistics needed for the classifier from the unlabeled samples. Therefore, at the end of the clustering algorithm we estimated the sample mean vector and covariance matrix from each cluster and then pass them to the Bayes classifier. In the next section we will present some simulation results based upon the algorithm described here.

5.3 Simulation Results

Figure 5.2(a) shows the first test image which is a texture mosaic consists of four different textures: grass, water, pigskin, and leather. All four textures are present in the image in squares of size 128 × 128. Figure 5.2(b) is the gray level coded ground truth of Figure 5.2(a).

In order to visualize how patterns are distributed in the feature domain, we plot the 2-dimensional scatter diagram in Figure 5.4 to Figure 5.9. Each diagram represents the scatter plot of the four texture classes in the selected feature pair. The label associated with each pattern is for illustration purpose only, the unsupervised classifier does not need labeled training patterns available. Samples are taken from every fourth line and every fourth pixel from the feature planes. From these plots we can clearly see that the patterns form clusters and that the clusters are reasonably well separated in some of the feature pairs such as $E5S5715 - R5R5715$ and $L5S5715 - R5R5715$. This

gives us a good indication that the problem itself is well suited to a clustering algorithm. In addition, each cluster seems to have a simple hyperellipsoidal shape, which indicates the multivariate normal distribution assumption about the patterns for each class is a valid one.

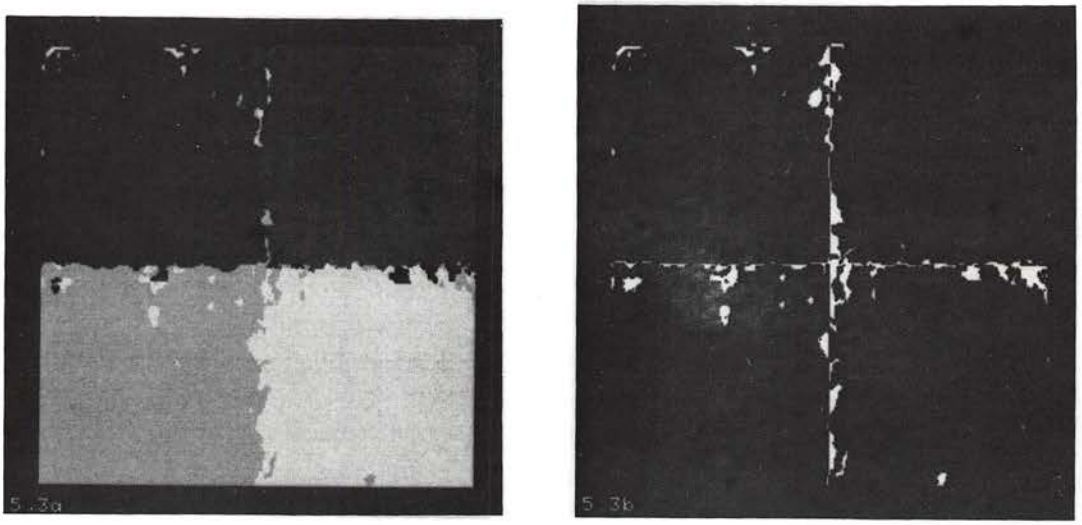
After we apply the clustering algorithm described in the last section to the data, the statistics estimated for each cluster are shown in Table 5.1. Based upon these estimated cluster statistics and the multivariate normal distribution assumption for each cluster, the Bayes classifier is then used to classify every pixel. Figure 5.3(a) shows the result produced by Bayes classifier in gray level coded form. Comparing this result to the ground truth we can generate a binary image, Figure 5.3(b), which shows all the misclassified pixels in white. The error rate is about 3.68 percent, which is slightly higher than the error rate of 3.12 percent obtained using a supervised classifier as described in chapter 4.

Figure 5.10(a) shows the second test image which consists of five different textures: grass, water, pigskin, leather, and raffia. The gray level coded ground truth of Figure 5.10(a) is shown in Figure 5.10(b). The statistics of each cluster learned from the clustering algorithm are shown in Table 5.2. Figure 5.11(a) shows the classification result in gray level coded form. The misclassified pixels (white) are shown in Figure 5.11(b). The error rate is about 4.23 percent, which is surprisingly lower than the 4.93 percent that we have obtained in chapter 4. This means that the statistics estimated from the clustering algorithm sometimes may even better than the statistics estimated from training prototypes.

From the previous two examples we may conclude that the feature data sets we

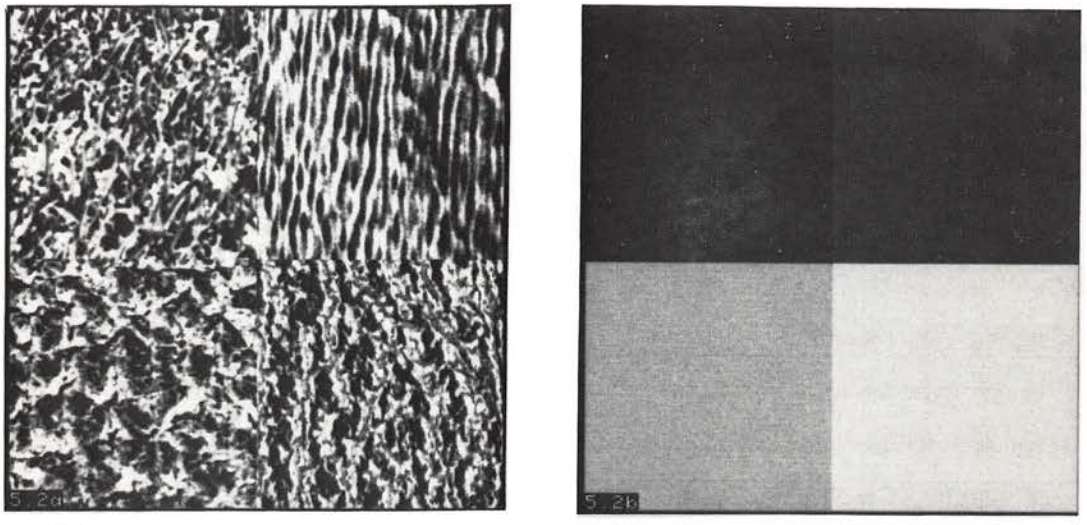
have on hand are fairly well-separated, thus the supervised and unsupervised results are comparable. We also note from the scatter plots that even though classes are reasonably well separated in some of the feature pairs, they still partially overlap in the feature space. This overlap causes mislabeling of pixels. Another question concerns the effect of subsampling of patterns used in the cluster analysis on the class statistic estimates. We think it should not result in any noticeable differences of class statistic estimates, since neighborhood pixels are highly correlated as a result of the large size quadrant window smoothing operation.

Up to this point we have not enforced any spatial constraints to reduce the ambiguities of labeled pixels. This will be the subject of the next chapter.



(a) Texture Mosaic (b) Ground Truth

Figure 5.2: Texture Mosaic 1 for Experimental Work.



(a) Gray Level Coded Result (b) Misclassified Pixels in White

Figure 5.3: Classification Results of Mosaic 1 Using Estimated Statistics.

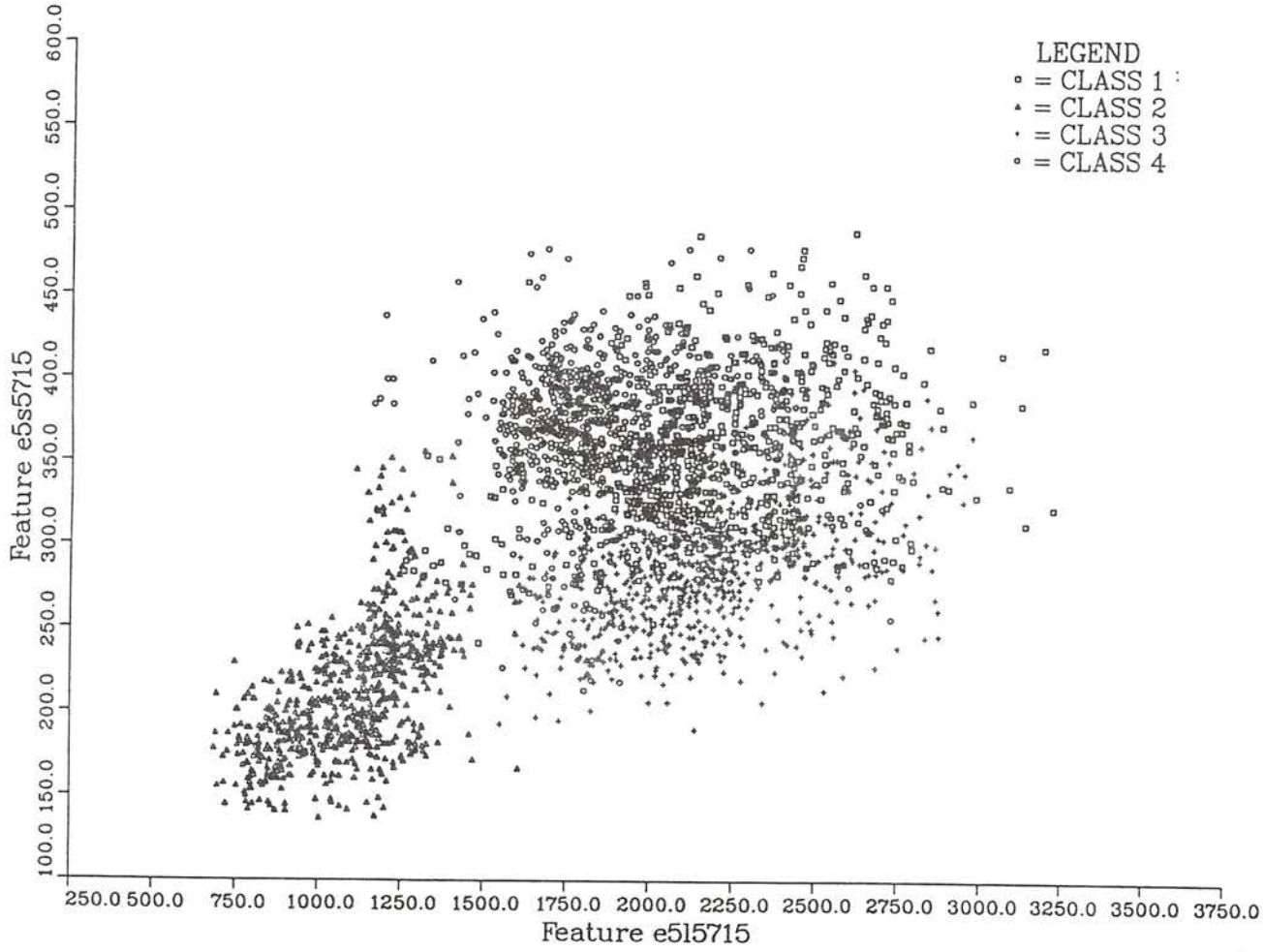


Figure 5.4: Scatter Diagram of Two Features $E5L5715$ versus $E5S5715$.

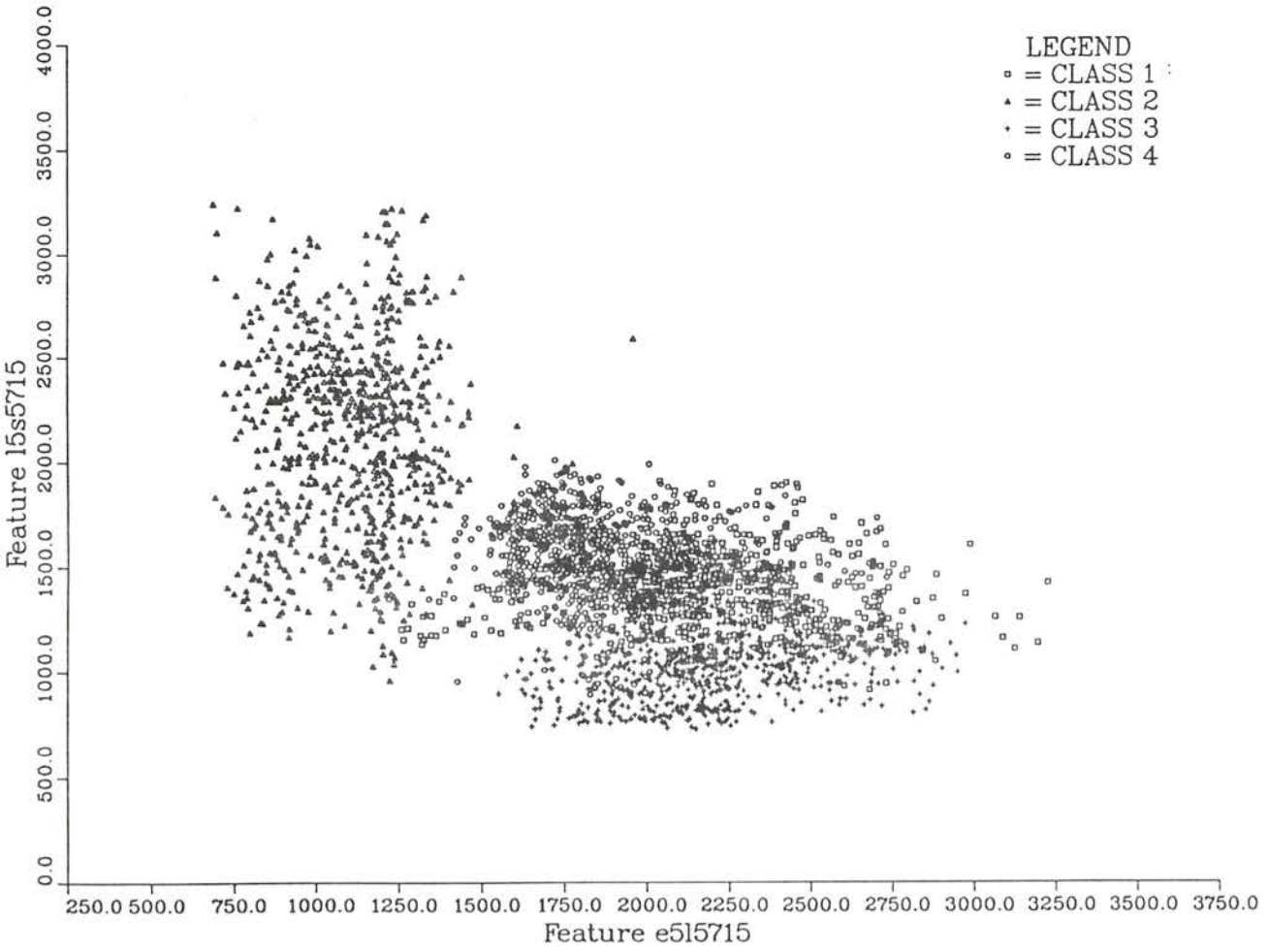


Figure 5.5: Scatter Diagram of Two Features $E5L5715$ versus $L5S5715$.

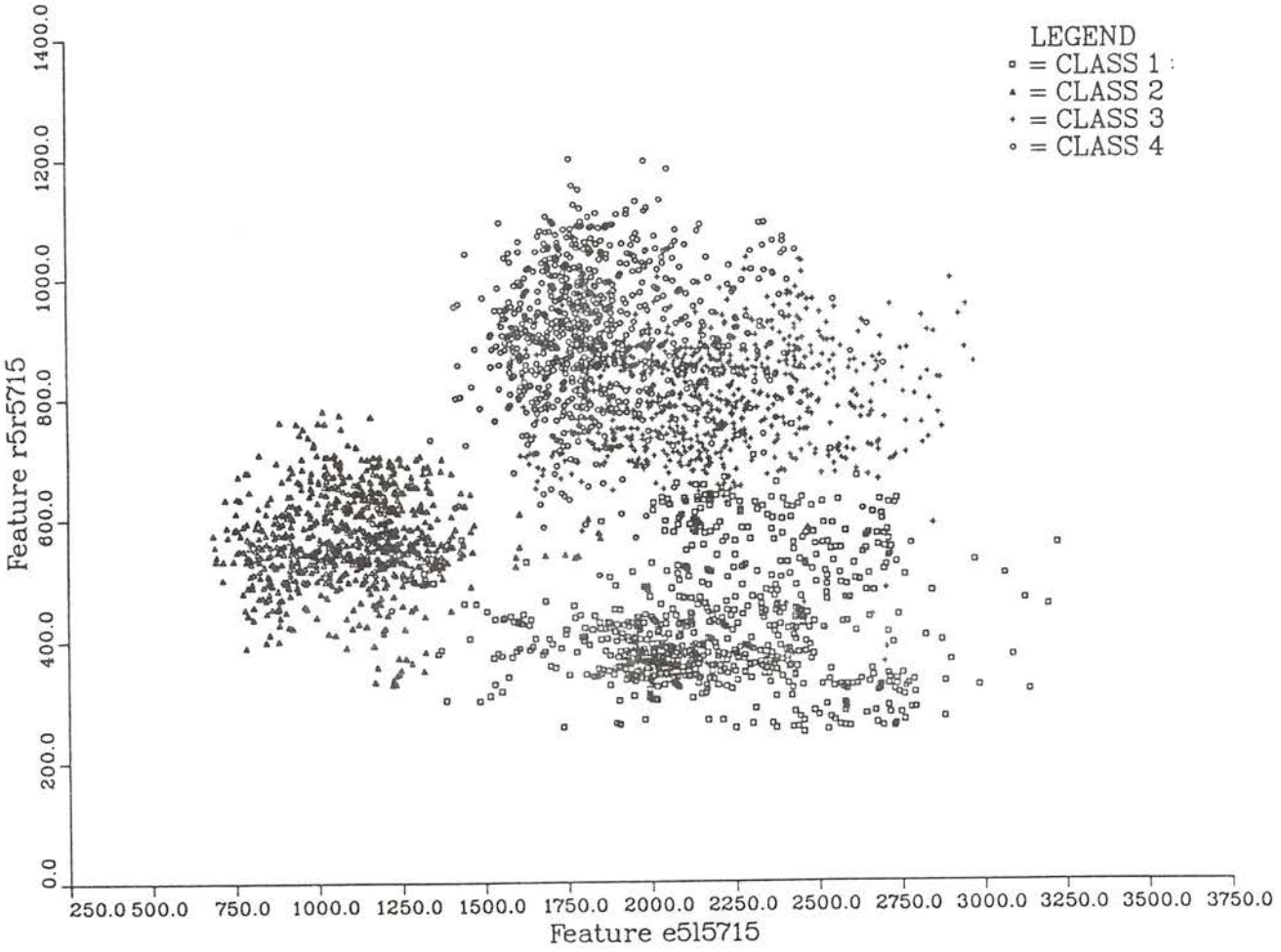


Figure 5.6: Scatter Diagram of Two Features $E5L5715$ versus $R5R5715$.

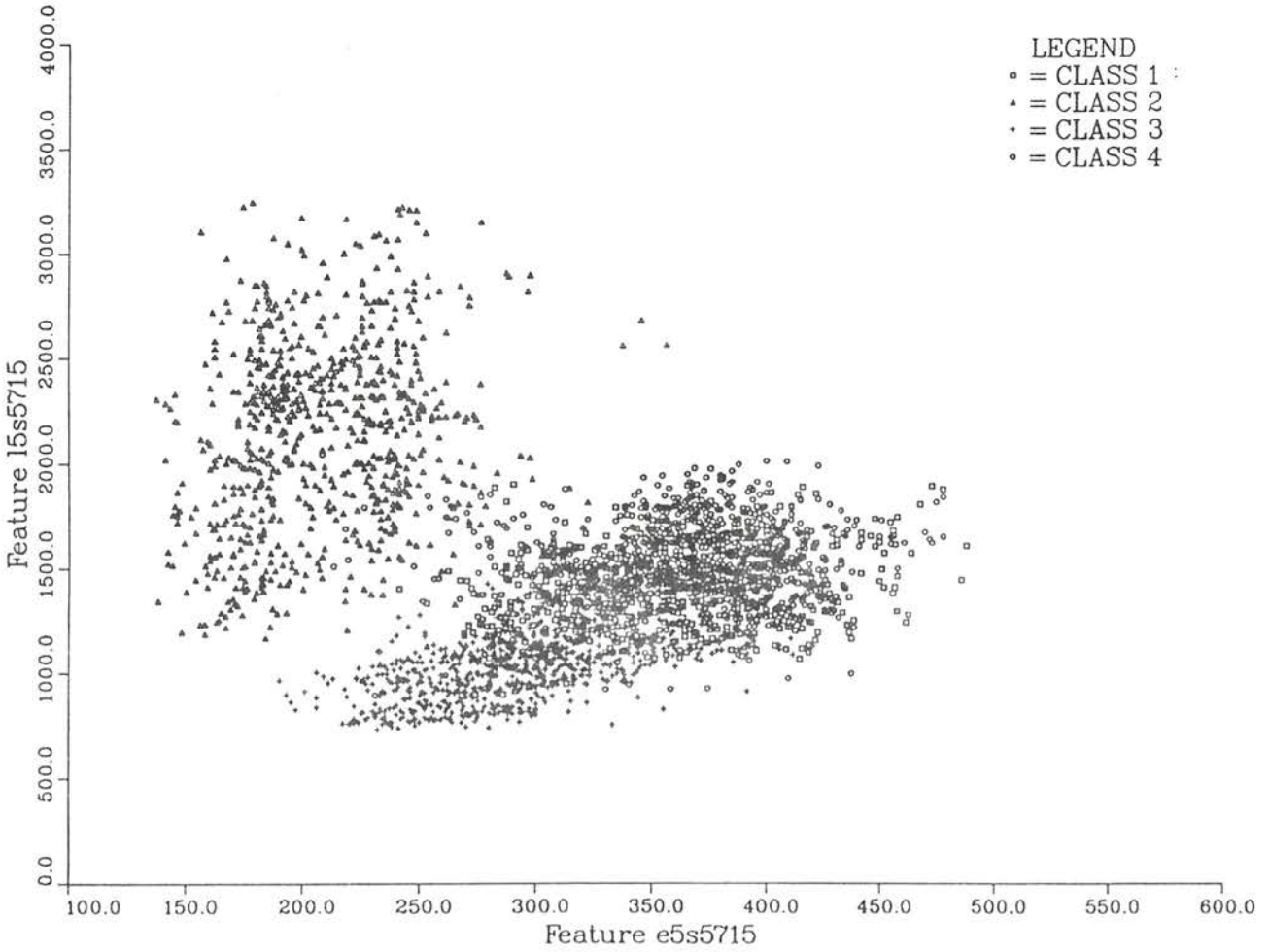


Figure 5.7: Scatter Diagram of Two Features $E5S5715$ versus $L5S5715$.

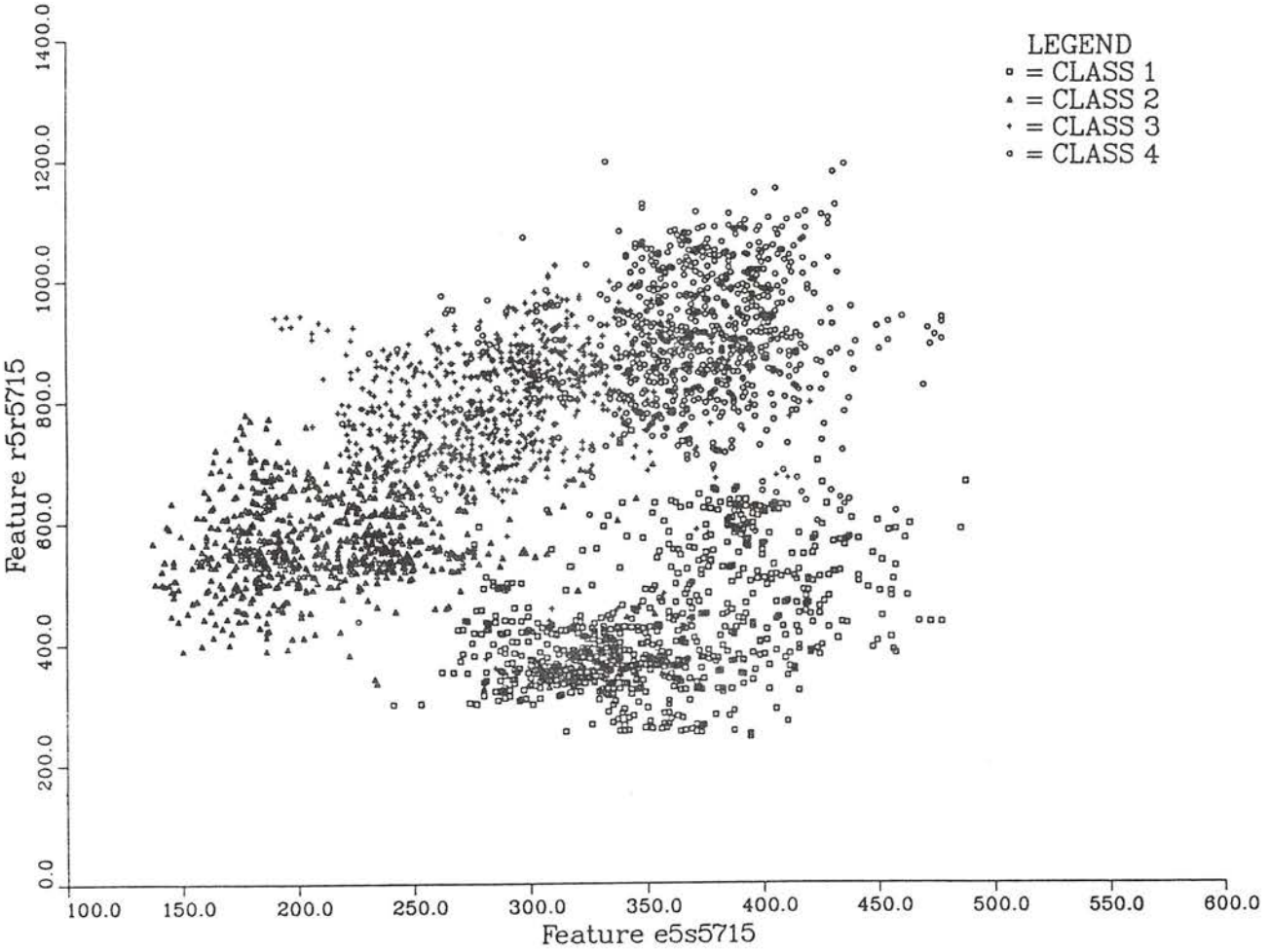


Figure 5.8: Scatter Diagram of Two Features $E5S5715$ versus $R5R5715$.

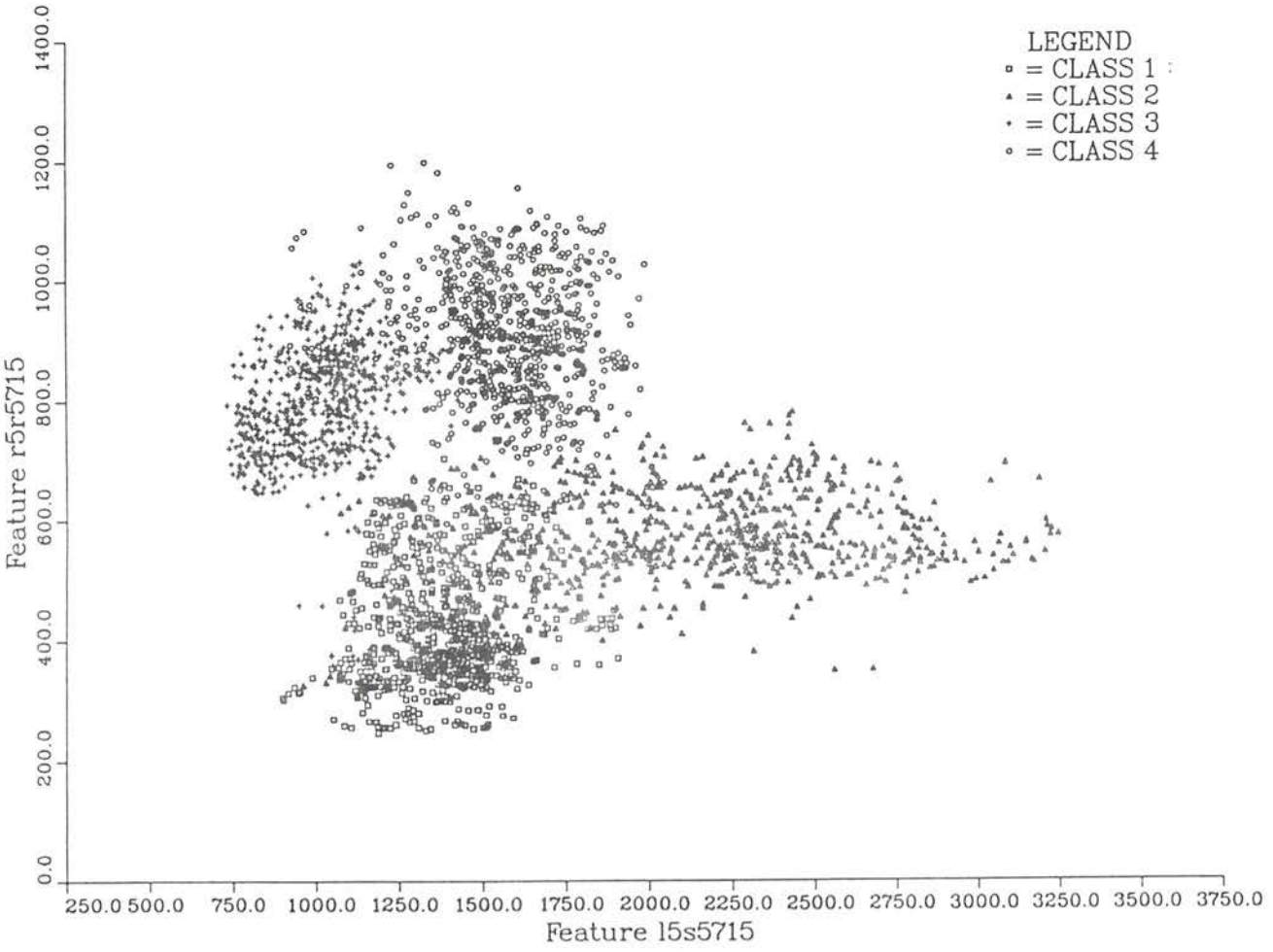


Figure 5.9: Scatter Diagram of Two Features $E5S5715$ versus $R5R5715$.

Class 1:

$$\hat{\mu}_1 = \begin{bmatrix} 2186.33 \\ 353.44 \\ 1381.13 \\ 420.09 \end{bmatrix} \quad \hat{\Sigma}_1 = \begin{bmatrix} 113591.00 & 3942.57 & -9470.76 & 4072.05 \\ 3942.57 & 1840.95 & 1570.56 & 282.51 \\ -9470.76 & 1570.56 & 34738.49 & 2645.80 \\ 4072.05 & 282.51 & 2645.80 & 9595.98 \end{bmatrix}$$

Class 2:

$$\hat{\mu}_2 = \begin{bmatrix} 1849.99 \\ 379.64 \\ 1565.02 \\ 907.04 \end{bmatrix} \quad \hat{\Sigma}_2 = \begin{bmatrix} 60625.39 & 561.92 & -10707.75 & 1434.09 \\ 561.92 & 897.83 & -427.33 & 133.29 \\ -10707.75 & -427.33 & 35876.14 & -2724.69 \\ 1434.09 & 133.29 & -2724.69 & 12325.16 \end{bmatrix}$$

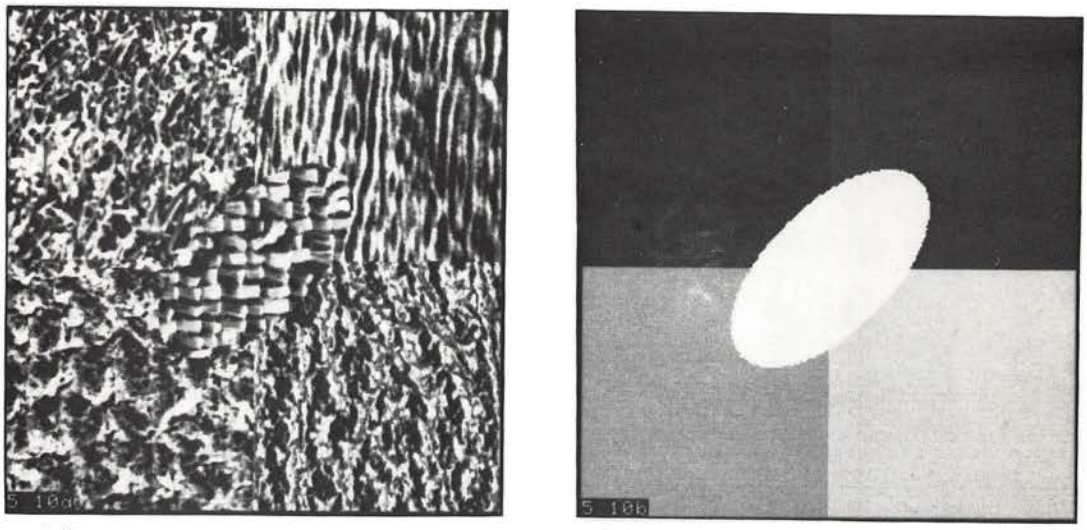
Class 3:

$$\hat{\mu}_3 = \begin{bmatrix} 2117.23 \\ 282.51 \\ 1042.02 \\ 789.73 \end{bmatrix} \quad \hat{\Sigma}_3 = \begin{bmatrix} 87590.06 & 2952.99 & -3855.30 & 3727.11 \\ 2952.99 & 999.11 & 1687.15 & 861.03 \\ -3855.30 & 1687.15 & 37699.60 & 2946.45 \\ 3727.11 & 861.03 & 2946.45 & 8877.83 \end{bmatrix}$$

Class 4:

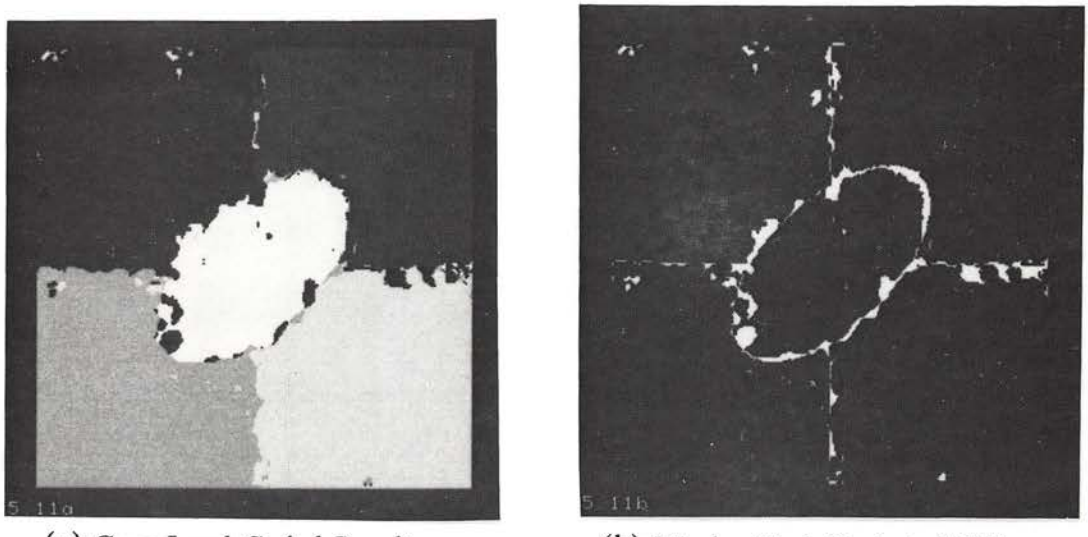
$$\hat{\mu}_4 = \begin{bmatrix} 1095.63 \\ 204.95 \\ 2078.92 \\ 571.01 \end{bmatrix} \quad \hat{\Sigma}_4 = \begin{bmatrix} 38469.51 & 3566.36 & 1468.14 & 903.01 \\ 3566.36 & 1128.25 & 3229.56 & 168.49 \\ 1468.14 & 3229.56 & 205809.3 & 5520.99 \\ 903.01 & 168.49 & 5520.99 & 4664.20 \end{bmatrix}$$

Table 5.1: Mean Vectors and Covariance Matrices for Four Texture Classes.



(a) Texture Mosaic (b) Ground Truth

Figure 5.10: Texture Mosaic 2 for Experimental Work.



(a) Gray Level Coded Result (b) Misclassified Pixels in White

Figure 5.11: Classification Results of Mosaic 2 Using Estimated Statistics.

Class 1:

$$\hat{\mu}_1 = \begin{bmatrix} 2184.95 \\ 354.97 \\ 1378.21 \\ 436.36 \end{bmatrix} \quad \hat{\Sigma}_1 = \begin{bmatrix} 110014.40 & 3933.55 & -5774.52 & 1663.83 \\ 3933.55 & 2019.06 & 1505.99 & 1892.66 \\ -5774.52 & 1505.99 & 33314.60 & 1915.88 \\ 1663.83 & 1892.66 & 1915.88 & 8885.39 \end{bmatrix}$$

Class 2:

$$\hat{\mu}_2 = \begin{bmatrix} 1844.73 \\ 378.74 \\ 1580.03 \\ 913.76 \end{bmatrix} \quad \hat{\Sigma}_2 = \begin{bmatrix} 50350.62 & 750.41 & -4984.10 & 1225.25 \\ 750.41 & 927.46 & -615.09 & 360.04 \\ -4984.10 & -615.09 & 32158.93 & -2620.75 \\ 1225.25 & 360.04 & -2620.75 & 11453.80 \end{bmatrix}$$

Class 3:

$$\hat{\mu}_3 = \begin{bmatrix} 2125.01 \\ 276.91 \\ 1003.04 \\ 792.89 \end{bmatrix} \quad \hat{\Sigma}_3 = \begin{bmatrix} 78745.71 & 3760.30 & 4794.06 & -613.94 \\ 3760.30 & 1236.20 & 1594.83 & 1564.32 \\ 4794.06 & 1594.83 & 24427.44 & 3309.81 \\ -613.94 & 1564.32 & 3309.81 & 6760.91 \end{bmatrix}$$

Class 4:

$$\hat{\mu}_4 = \begin{bmatrix} 1068.34 \\ 202.01 \\ 2089.31 \\ 578.35 \end{bmatrix} \quad \hat{\Sigma}_4 = \begin{bmatrix} 40649.37 & 2853.76 & -3289.22 & 1936.48 \\ 2853.76 & 1150.54 & 1532.33 & 552.44 \\ -3289.22 & 1532.33 & 189317.80 & 5929.95 \\ 1936.48 & 552.44 & 5929.95 & 6130.46 \end{bmatrix}$$

Class 5:

$$\hat{\mu}_5 = \begin{bmatrix} 2331.75 \\ 186.74 \\ 969.30 \\ 356.57 \end{bmatrix} \quad \hat{\Sigma}_5 = \begin{bmatrix} 116683.70 & 375.98 & -11071.40 & -1701.72 \\ 375.98 & 787.28 & 746.62 & 535.06 \\ -11071.40 & 746.62 & 37622.97 & 1207.38 \\ -1701.72 & 535.06 & 1207.38 & 3109.34 \end{bmatrix}$$

Table 5.2: Mean Vectors and Covariance Matrices for Five Texture Classes.

Chapter 6

Unsupervised Segmentation with Spatial Constraints

In chapter 5 we have discussed using clustering as a means to estimate statistics from unlabeled samples. Based upon these estimated statistics, a Bayes classifier may then be devised to classify pixels. One shortcoming of this method, as we mentioned in chapter 4, is that the spatial relationship has been ignored. In this chapter, we attempt to incorporate spatial constraints into our unsupervised segmentation algorithm through the use of probabilistic relaxation.

In this chapter, another area of interest is to study the effect of limiting the range of possible patterns for classification to a smaller subset of classes. From the initial probability labeling of each pixel, we will see that many pixels have a very small probability of assignment to certain classes. One way to limit this uncertainty is through the use of probability thresholding. If we let T_P be a probability threshold value, we then set label probabilities to zero if they are less than T_P. Based upon this concept, an algorithm is developed and its effectiveness is evaluated.

In section 6.1 we describe the algorithm used to enforce spatial constraints under the assumption that no training prototypes are available. Section 6.2 discusses the issues related to limiting probability labels to a smaller subset of classes and describes an algorithm to implement it. In section 6.3 we present the simulation results of the aforementioned two algorithms.

6.1 Incorporating Spatial Constraints into Unsupervised Segmentation Algorithms

Because there are no training prototypes available, the first step is to learn the statistics from the unlabeled samples. The clustering algorithm described in chapter 5 may be used to form clusters in the feature space. Once the clusters are formed then the statistics such as mean vectors and covariance matrices for each cluster can be estimated. We use these estimated cluster statistics as class statistics for our initial probability labeling.

Knowing the cluster statistics and assuming that each cluster is multivariate Gaussian distributed, then the probability of assigning pixel with feature vector $\vec{x}$ to cluster λ_i is given by

$$P(\lambda_i|\vec{x}) = \frac{p(\vec{x}|\lambda_i)P(\lambda_i)}{\sum_{\lambda_j \epsilon \Lambda} p(\vec{x}|\lambda_j)P(\lambda_j)}\,, \tag{6.1}$$

where $P(\lambda_i)$ is the *a priori* probability of each cluster. These *a posteriori* probabilities of each pixel may be used as the initial labeling probability $P_i^{(0)}(\lambda)$ and compatibility coefficients may then be calculated.

Based upon the study we did in chapter 4, the probabilistic relaxation scheme

developed by Peleg converges much faster than the original scheme. Hence we will concentrate our study using Peleg's algorithm. Let us summarize Peleg's algorithm here for convenience.

1. For each one of the 8-connected neighbors we use Eqs. (6.2)–(6.4) to calculate the compatibility coefficients. The coefficients $r_{i,i+\delta}(\lambda,\lambda')$ are fixed throughout the iteration process. The *a priori* probabilities $P(\lambda)$ are estimated by

$$\hat{P}(\lambda) = \frac{1}{n}\sum_{i=1}^{n} P_i^{(0)}(\lambda) \tag{6.2}$$

The estimates for the joint probabilities are

$$\hat{P}_{i,i+\delta}(\lambda,\lambda') = \frac{1}{n}\sum_{i} P_i^{(0)}(\lambda) P_{i+\delta}^{(0)}(\lambda') \quad \delta\epsilon\Delta\,, \tag{6.3}$$

where n is the number of points, pixel $i+\delta$ is a specific neighbor of pixel i, and again Δ defines the neighborhood about the particular pixel being considered. The compatibility coefficients are then defined as

$$r_{i,i+\delta}(\lambda,\lambda') = \frac{\hat{P}_{i,i+\delta}(\lambda,\lambda')}{\hat{P}(\lambda)\hat{P}(\lambda')} \quad \delta\epsilon\Delta\,. \tag{6.4}$$

2. The iterative updating process follows by using Eqs. (6.5)–(6.6). The $q_{i,i+\delta}^{(k+1)}(\lambda)$ is the pairwise effect of one of the 8-connected neighborhood.

$$S_{i,i+\delta}^{(k+1)}(\lambda) = \sum_{\lambda'\epsilon\Lambda} \hat{P}_i^{(k)}(\lambda)\hat{P}_{i+\delta}^{(k)}(\lambda') r_{i,i+\delta}(\lambda,\lambda') \quad \delta\epsilon\Delta\,, \tag{6.5}$$

and

$$q_{i,i+\delta}^{(k+1)}(\lambda) = \frac{S_{i,i+\delta}^{(k+1)}(\lambda)}{\sum_{\lambda'\epsilon\Lambda} S_{i,i+\delta}^{(k+1)}(\lambda')} \quad \delta\epsilon\Delta\,, \tag{6.6}$$

here $q_{i,i+\delta}^{(k+1)}$ is the new probability estimate for the labels at pixel a_i based upon the previous estimates at a_i and $a_{i+\delta}$.

3. Because every pixel has eight neighbors, the new $P_i^{(k+1)}(\lambda)$ is the average of these eight estimates as in Eq. (6.7).

$$P_i^{(k+1)}(\lambda) = \frac{1}{8} \sum_{\delta \epsilon \Delta} q_{i,i+\delta}^{(k+1)}(\lambda) \quad \lambda \epsilon \Lambda \,. \tag{6.7}$$

4. Repeat steps (2)–(3) until the stopping criteria is met. In our case we specified the number of iterations in advance.

In section 6.3 we will present the simulation results of applying the previously described algorithm to our two texture mosaics.

6.2 Limiting the Probability Labels by Probability Thresholding

One observation which we can make from the initial probability labeling process is that many pixels have very low probabilities of assignment to some of the classes. This suggests that it may be useful to limit the pixel classification only to those classes whose probabilities exceed a given threshold. To be more specific, we may define T_P to be a probability threshold, and let those initial probabilities vanish if their values are less than T_P, thus restricting the choice of possible pixel labels.

One immediate advantage of probability thresholding is that the amount of data to be processed during the following relaxation iterations decreases as a function of the threshold settings. The higher the threshold setting is, less data needs to be processed. On the other hand, once certain classes are suppressed, they will never reappear through the use of probabilistic relaxation. This becomes a trade-off problem. Our goal is to

study how different probability threshold settings affect the amount of data to be processed and its classification rates.

The algorithm we use to study the effects of limiting the decision space by probability thresholding is described as follows:

1. Choose the value of probability threshold T_P.

2. Use the clustering algorithm described in chapter 5 to form clusters in the feature space. Once the clusters are formed then the statistics such as mean vectors and covariance matrices for each clusters can be estimated. We use these unbiased cluster statistics as class statistics for our initial probability labeling purpose.

3. Based upon the estimated cluster statistics and normal distribution assumption, the cluster labeling probabilities are assigned to each pixel according to:

$$P(\lambda|\vec{x}) = \frac{p(\vec{x}|\lambda)P(\lambda)}{\sum_{\lambda'} p(\vec{x}|\lambda')P(\lambda')}\,, \tag{6.8}$$

   where $p(\vec{x}|\lambda)$ is the cluster conditional density function

$$p(\vec{x}|\lambda) = \frac{1}{(2\pi)^{1/2}|\Sigma_\lambda|^{1/2}} \exp(-\frac{1}{2}(\vec{x}-\vec{\mu}_\lambda)^t \Sigma_\lambda{}^{-1}(\vec{x}-\vec{\mu}_\lambda))\,. \tag{6.9}$$

   We use the unbiased sample estimates $\hat{\mu}_\lambda$ and $\hat{\Sigma}_\lambda$ for $\vec{\mu}_\lambda$ and Σ_λ respectively.

4. If any of the $P(\lambda|\vec{x})$ is less than T_P, we set it to zero and then normalize the nonzero probabilities.

5. From these thresholded initial probability labelings, a table of compatibility coefficients may be generated.

6. A mask for each class which indicates those pixels with probability zero or one is then generated. This mask can be used to guide the relaxation process to work

only on those pixels whose probabilities are not zero or one. In addition, the amount of data needs to be processed can also be calculated from this mask.

7. With the initial probability labeling and compatibility coefficients on hand, we can start our probabilistic relaxation process.

8. After each iteration or a few iterations the classification rate may be calculated. We repeat the relaxation process until the stopping criteria is met. In our case we specified the number of iterations in advance.

In the next section we present the results of applying this aforementioned algorithm with different probability thresholds to our two texture mosaics. By doing so, we can quantitatively understand the effect caused by different probability thresholds, and the results may help us in choosing the adequate threshold setting.

6.3 Simulation Results

Figure 6.1(a) and Figure 6.4(a) are the texture mosaics for this experiment. Each consists of four and five textures respectively and they are identical to those used in Figures 4.1 and 4.6. Figure 6.2(a) shows the initial segmentation result of the first mosaic by method mentioned in chapter 5. The error rate is about 3.68 percent. Figure 6.2(b) is the segmentation result after we applied 25 iterations of probabilistic relaxation to the first mosaic. The error rate went down from 3.68 percent to 2.5 percent. The classification rates were recorded every 5 iterations and were plotted in Figure 6.3.

Figure 6.5(a) is the initial segmentation result of the second mosaic. The initial

error rate is 4.23 percent. After we applied 25 iterations of relaxation to the second mosaic the result is shown in Figure 6.5(b). The error rate went down from 4.23 percent to 3.02 percent. Similarly, the classification rates were recorded every 5 iterations and were plotted in Figure 6.6.

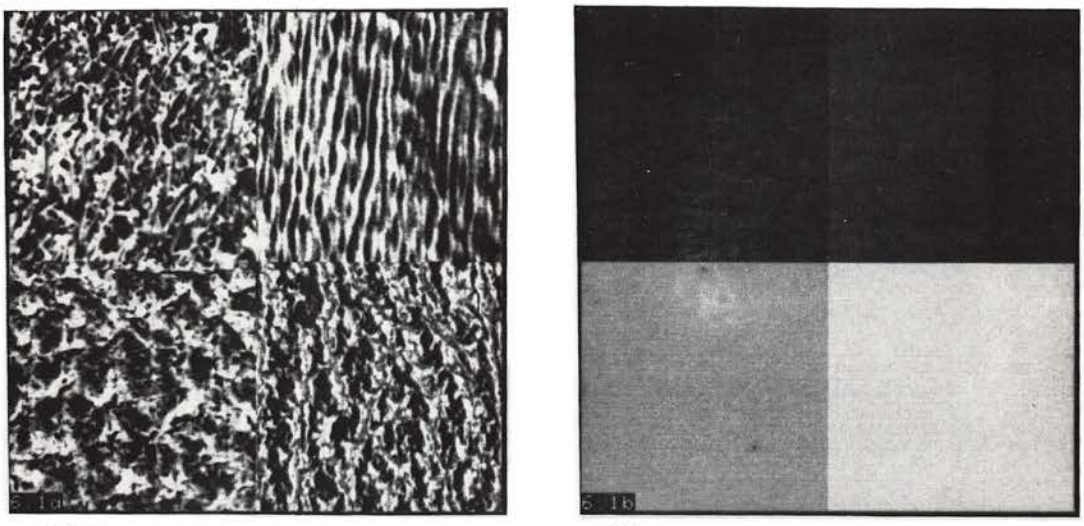
As a comparison, we also process the second test image using Laws' feature extraction method as described in chapter 3. The window size used to estimate macro statistical features is 15×15. Figure 6.7(a) shows the initial classification result with an error rate of 7.55 percent. After 25 iterations, the classification result is shown in Figure 6.7(b). The error rate went down from 7.55 percent to 6.66 percent. From this comparison, we can clearly see the performance is improved by using our proposed method.

Figure 6.8(a) shows the initial segmentation result of the first mosaic when the probability threshold was set to 0.5 percent and after applying 25 iterations of relaxation to the result shown in Figure 6.8(b). The total number of pixels to be processed is about 8.5 percent of the total processed if we do not set a threshold. This is a tremendous saving of computation time, however, at the expense of higher misclassification rate as shown in Table 6.1. If we increase the threshold to 1 percent, the initial segmentation and the segmentation after 25 iterations are shown in Figure 6.9(a) and Figure 6.9(b) respectively. The number of pixels that needs to be processed decreased from 8.5 percent to 7 percent. The results for first mosaic are summarized in Table 6.1.

For the second mosaic we did the same experiments and the results are shown in Figure 6.10 to Figure 6.11. Figure 6.10(a) and Figure 6.10(b) show the results of initial segmentation and the segmentation after 25 iterations respectively for 0.5

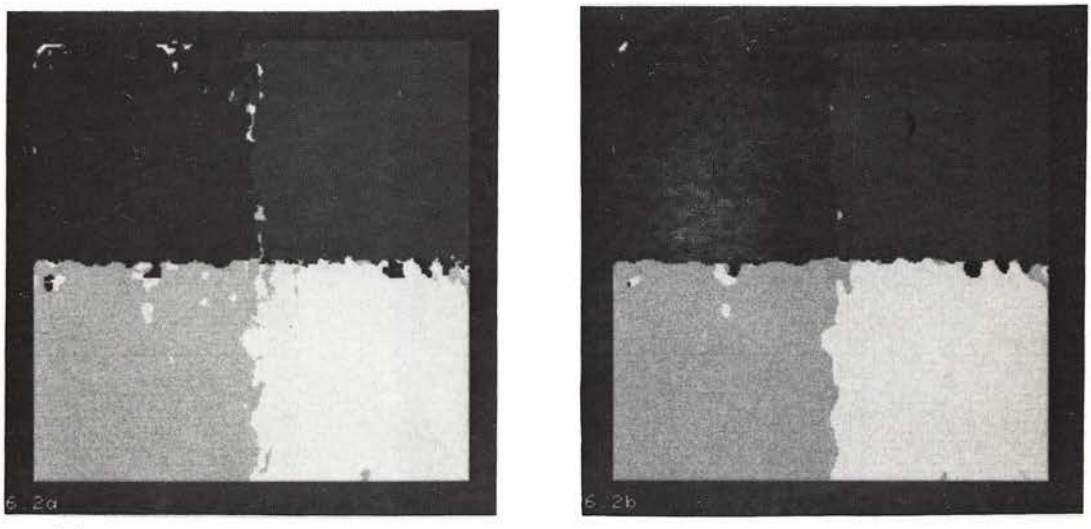
percent threshold. The number of pixels that must be processed is about 5 percent of the total pixels. Similarly, the results of initial segmentation and the segmentation after 25 iterations for 1 percent threshold are shown in Figure 6.11(a) and Figure 6.11(b) respectively. The number of pixels to be processed decreased from 5 percent to 4 percent. The results for second mosaic are summarized in Table 6.2.

From the above results we can draw some conclusions about the probability threshold setting. First, even a very low threshold such as 1 percent reduces the computation significantly and causes limited degradation of performance. Second, under the circumstances that the computation load is of little concern or that the clusters are not well separated in feature space then we should set the the probability threshold very low (perhaps zero) in order to take full advantage of the spatial constraints. In other words, the setting of probability threshold is a data dependent problem, therefore, data analysis is needed before we can make a proper decision. For our data, the 1 percent threshold seems to be a good choice.



(a) Texture Mosaic (b) Ground Truth

Figure 6.1: Texture Mosaic 1 for Experimental Work.



(a) Initial Segmentation Result (b) After 25 Relaxation Iterations

Figure 6.2: Classification Results of Mosaic 1 Using Estimated Statistics.

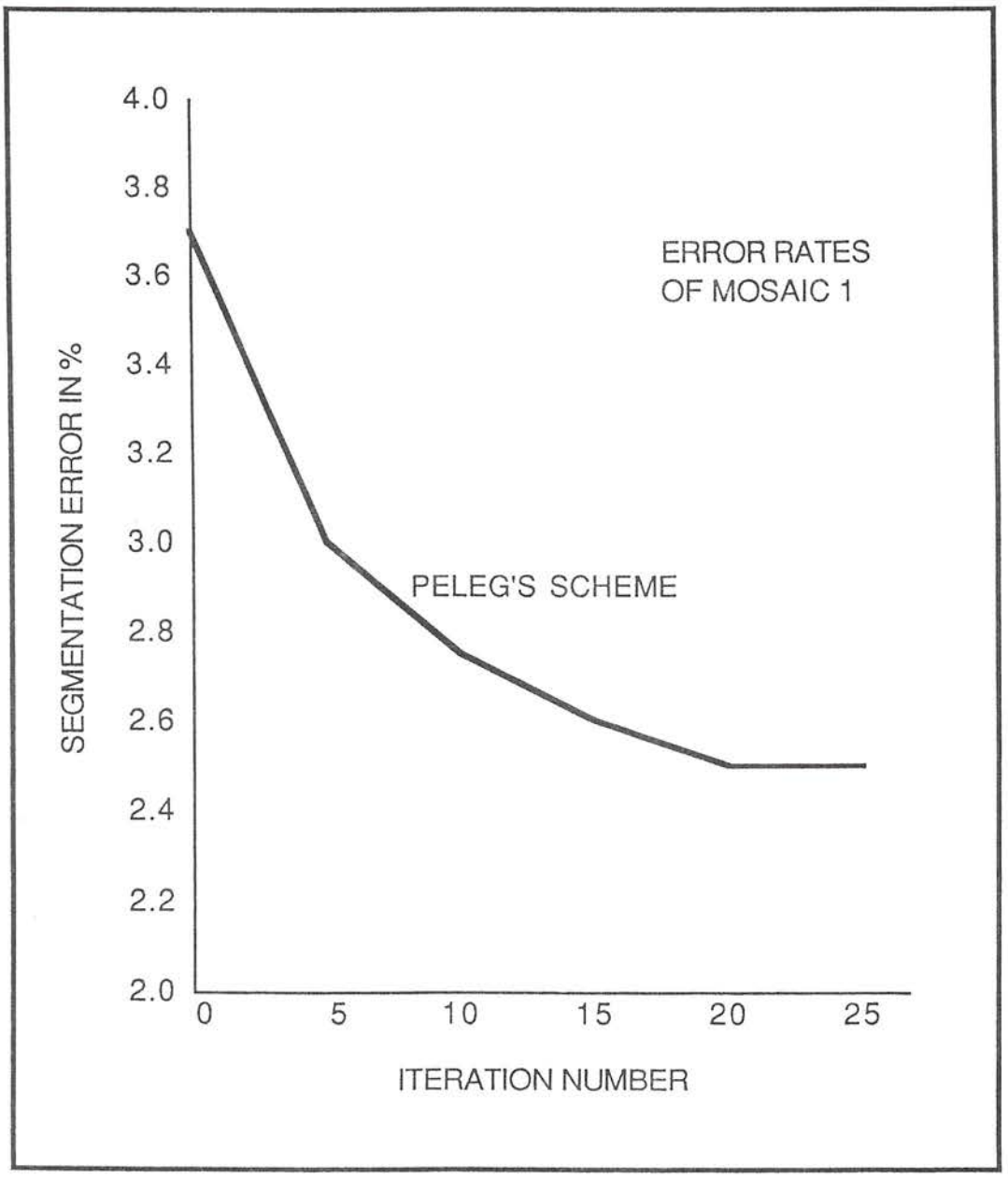
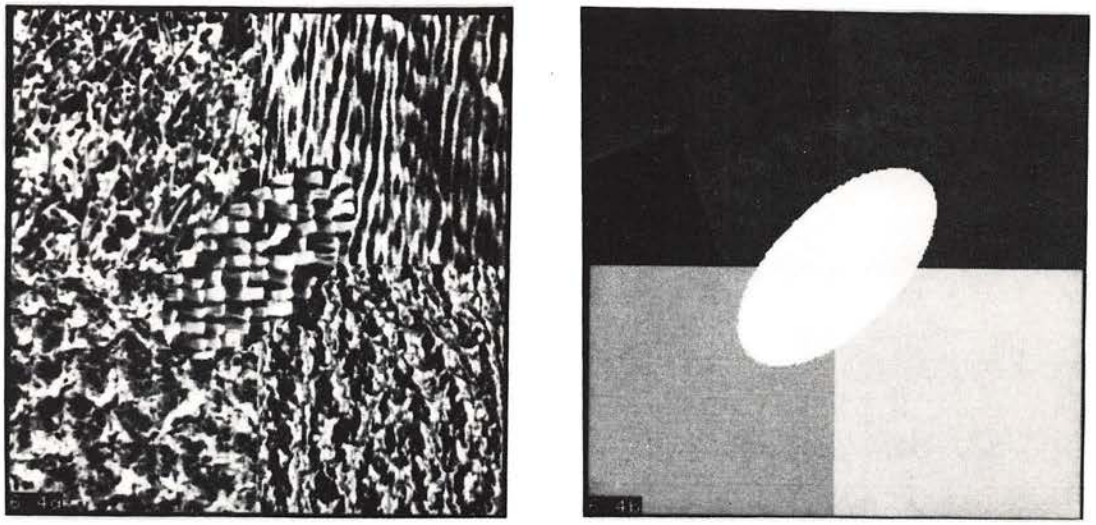
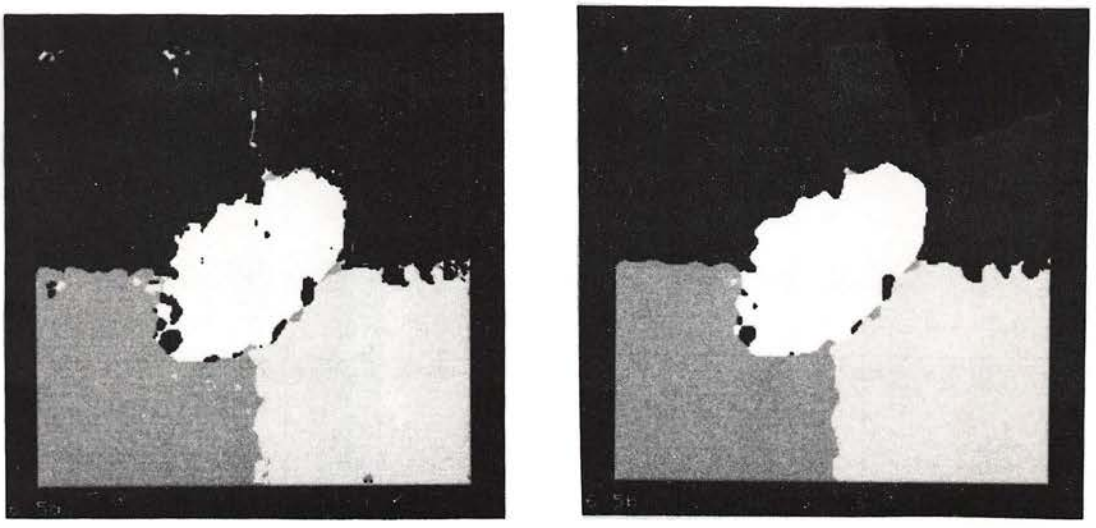


Figure 6.3: The Error Rates of Mosaic 1.



(a) Texture Mosaic (b) Ground Truth

Figure 6.4: Texture Mosaic 2 for Experimental Work.



(a) Initial Segmentation Result (b) After 25 Relaxation Iterations

Figure 6.5: Classification Results of Mosaic 2 Using Estimated Statistics.

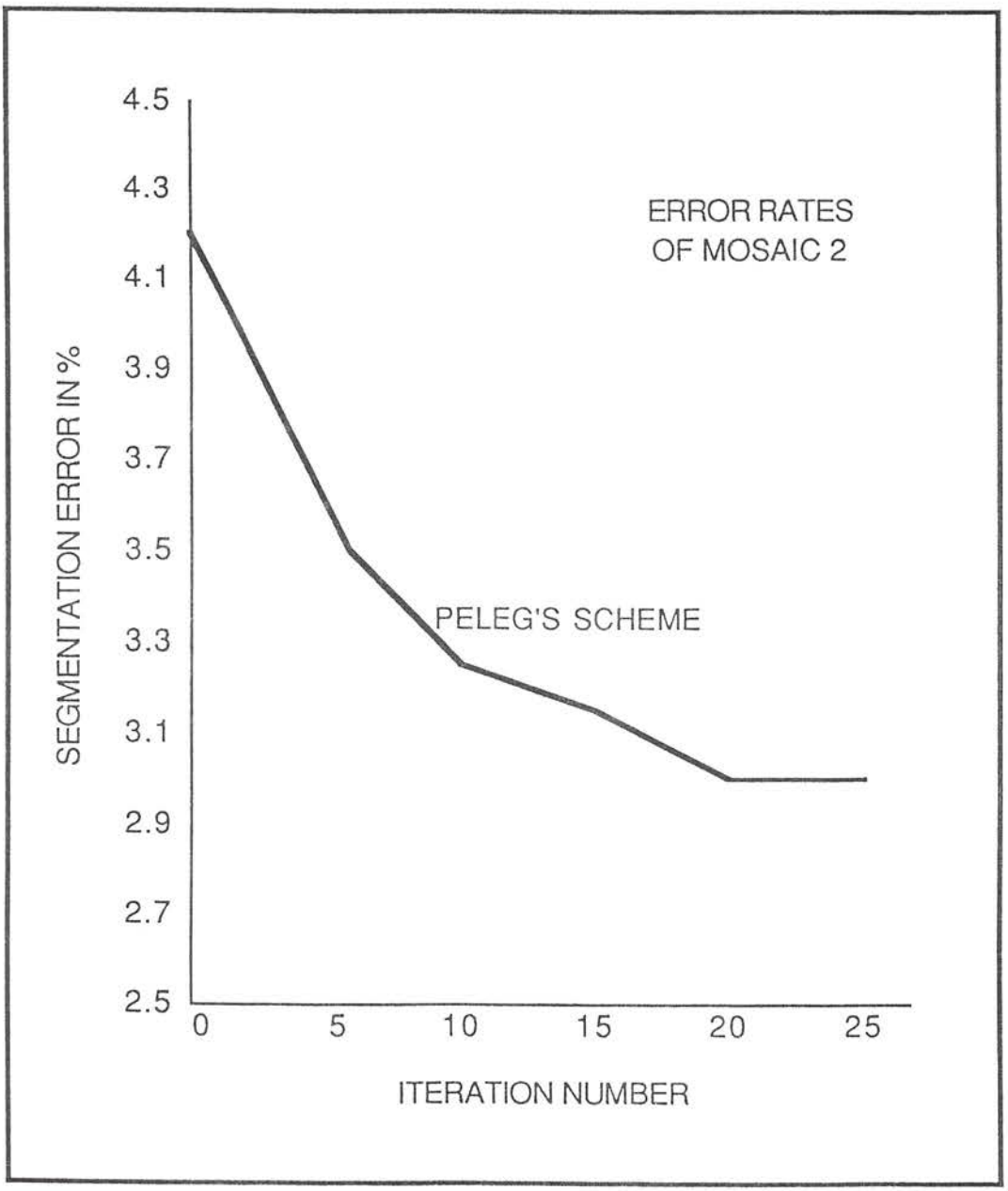
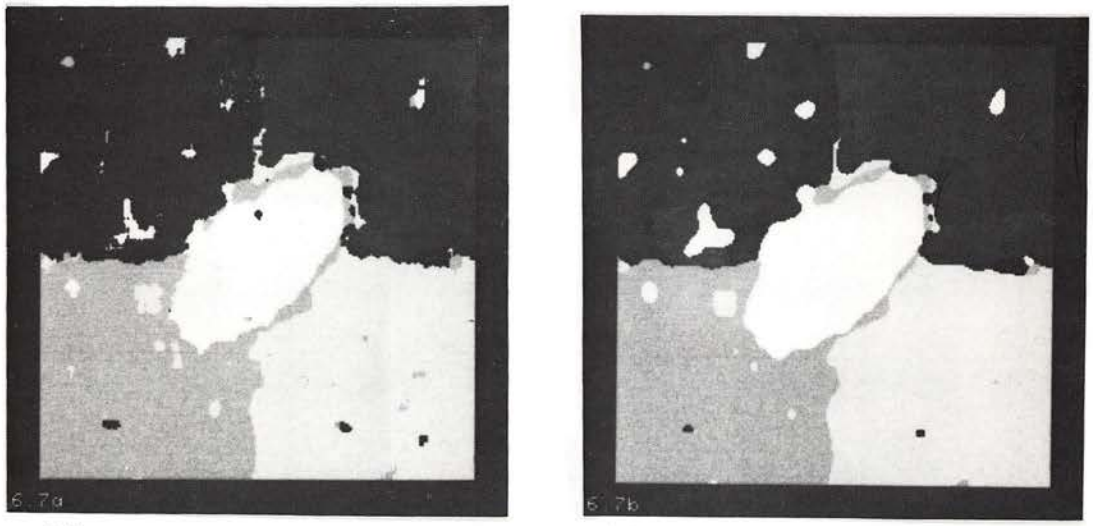
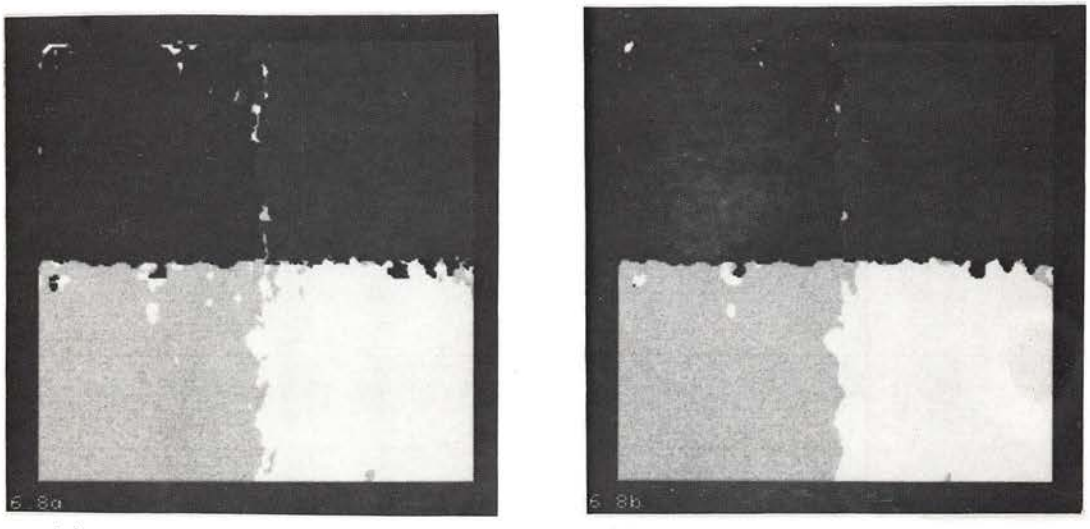


Figure 6.6: The Error Rates of Mosaic 2.



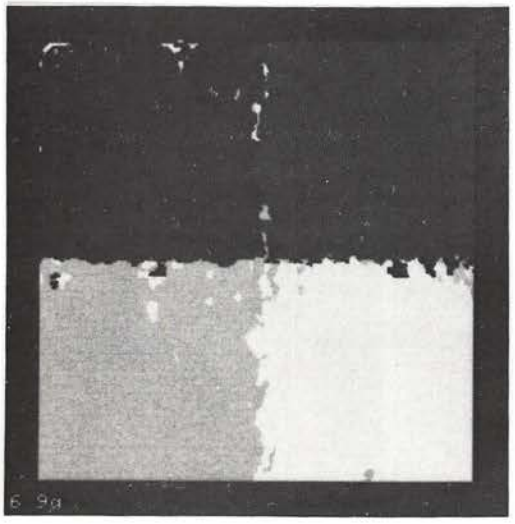
(a) Initial Segmentation Result (b) After 25 Relaxation Iterations

Figure 6.7: Segmentation Results of Mosaic 2 Using Laws' Method.

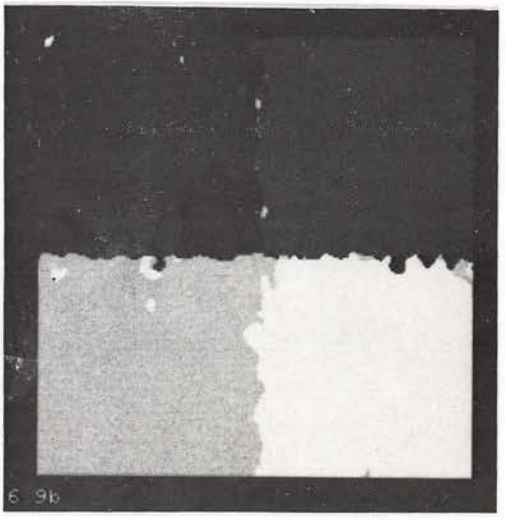


(a) Initial Segmentation Result (b) After 25 Relaxation Iterations

Figure 6.8: Segmentation Results of Mosaic 1 with 0.5 Percent Threshold.

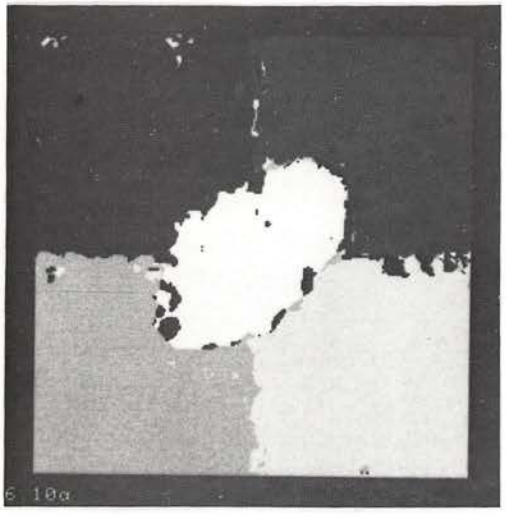


(a) Initial Segmentation Result

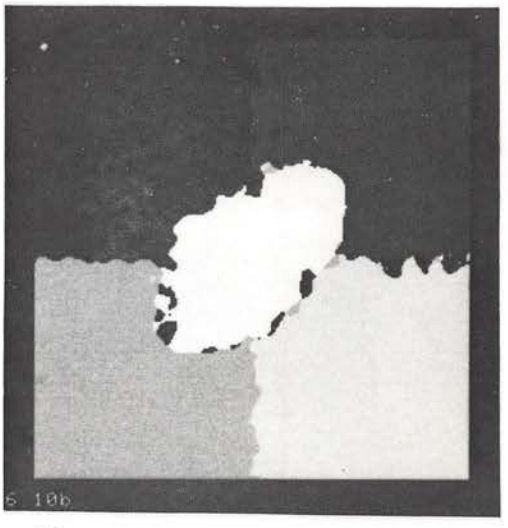


(b) After 25 Relaxation Iterations

Figure 6.9: Segmentation Results of Mosaic 1 with 1 Percent Threshold.

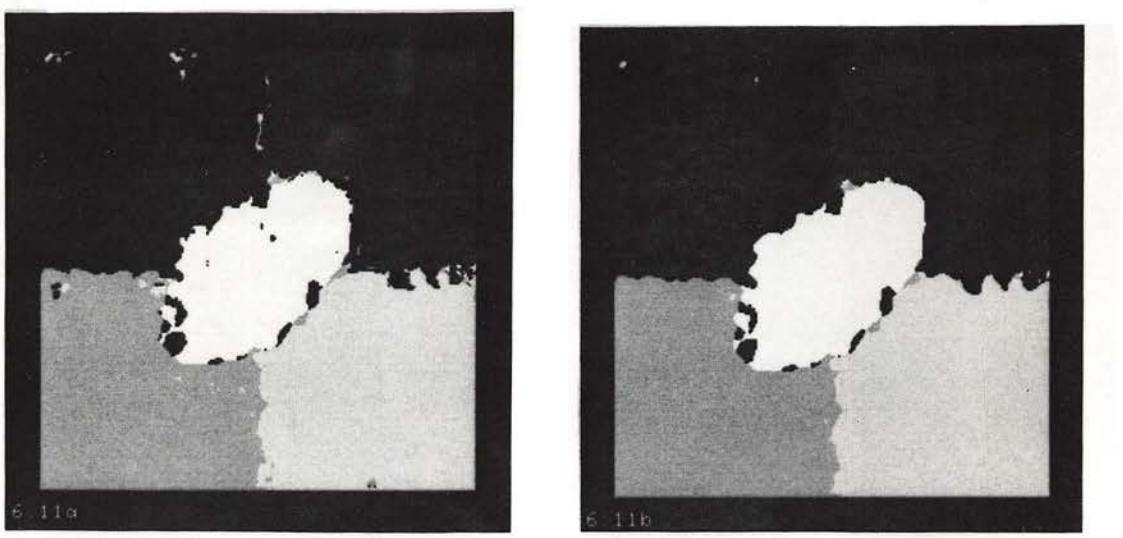


(a) Initial Segmentation Result



(b) After 25 Relaxation Iterations

Figure 6.10: Segmentation Results of Mosaic 2 with 0.5 Percent Threshold.



(a) Initial Segmentation Result (b) After 25 Relaxation Iterations

Figure 6.11: Segmentation Results of Mosaic 2 with 1 Percent Threshold.

Threshold setting (in %)	Percent of total pixels to be processed	Error rate after 25 iterations (in %)
0	100	2.5
0.5	8.5	2.67
1.0	7.0	2.74

Table 6.1: Summary of Probability Threshold Settings (in %) for First Mosaic

Threshold setting (in %)	Percent of total pixels to be processed	Error rate after 25 iterations (in %)
0	100	3.02
0.5	5.0	3.45
1.0	4.0	3.49

Table 6.2: Summary of Probability Threshold Settings (in %) for Second Mosaic

Chapter 7

Overview of Textured Image Segmentation System

In previous chapters, we have addressed issues related to different stages of a textured image segmentation system and have described algorithms for each of the stages. The purpose of this chapter is to tie these algorithms together to form a complete segmentation system.

Section 7.1 provides an overview of the supervised textured image segmentation system based upon the assumption that training samples for each texture are available. If these training samples are not available then the problem becomes an unsupervised one which is addressed in section 7.2. In sections 7.1 and 7.2 we also summarize the observations and guidelines we made about the parameters setting for the supervised and unsupervised textured image segmentation systems. Section 7.3 contains some additional discussion of these segmentation systems along with some additional issues to be considered.

7.1 An Overview of Supervised Textured Image Segmentation System

Figure 7.1 shows a block diagram of the proposed supervised textured image segmentation system. The textured image is first convolved with a set of 5×5 micro-texture feature extraction masks. Four micro-texture masks are selected beforehand for our study. No principal component rotation or feature selection was performed because our main concern is with segmentation given a set of features rather than selection of an ideal set of features for a given type of image. A set of statistics which are the local average of the absolute values of each of the feature statistics within a 7×7 window is then estimated for each pixel. These estimated local statistics are further processed by the EPNSQ filter to reduce their variance.

The EPNSQ filter implicitly uses the assumption that the local statistics image follows a nonstationary mean, nonstationary variance (NMNV) image model [57]. Basically, the NMNV model treats a 2-D image as a random field with nonstationary mean and nonstationary variance. Thus, the EPNSQ smoothing algorithm attempts to take the nonstationary nature of the problem into consideration by spatial selection of the averaging region.

Based upon our simulation results, using the proposed feature estimation scheme can provide us not only better correct classification rates but also more accurate region borders for region sizes larger than 16×16. For large region sizes of our test mosaic, i.e. 128×128, the improvement on the correct classification rate is about 7 percent.

For our test image, the chosen window sizes, i.e., 5×5 for micro-texture masks,

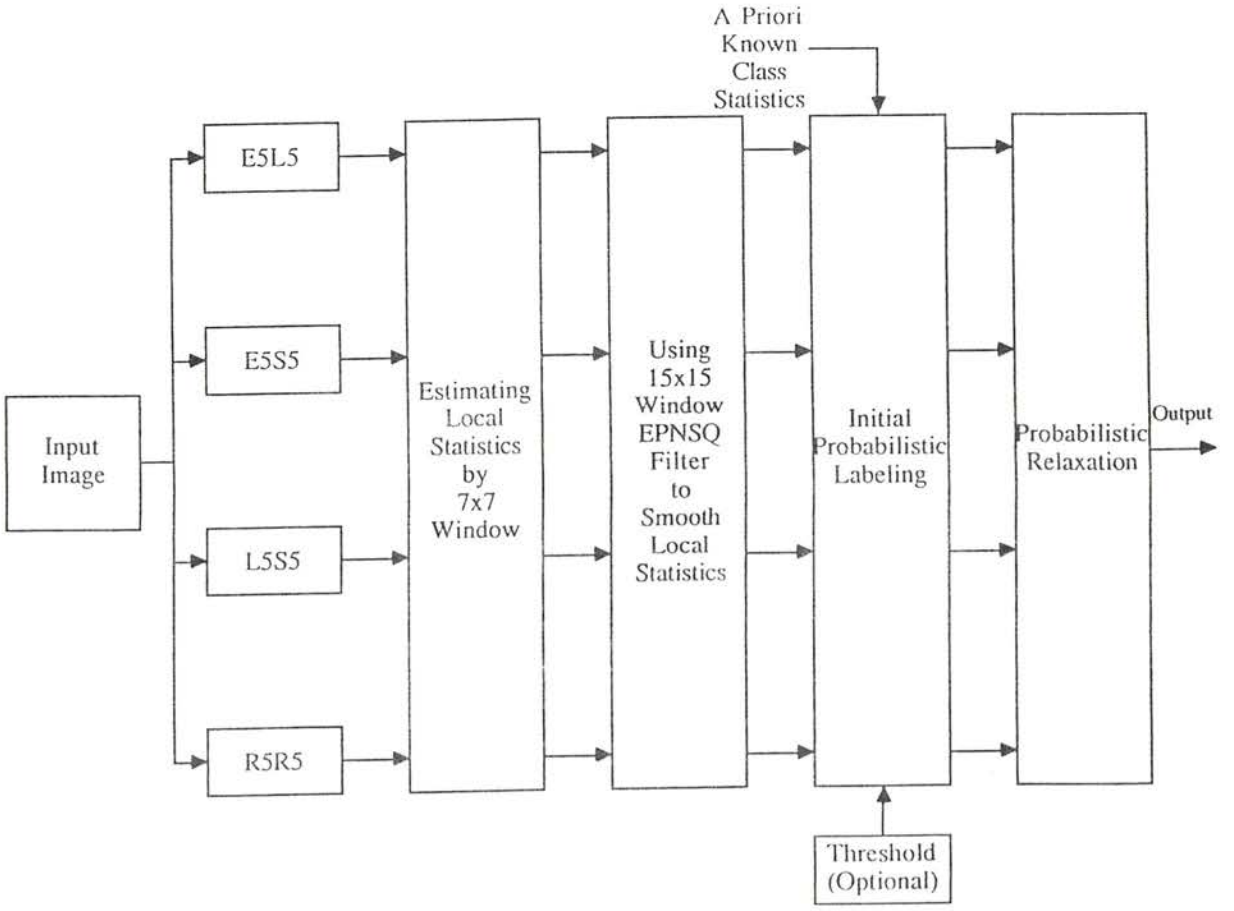


Figure 7.1: Block Diagram of Supervised Textured Image Segmentation System.

7×7 for macro statistics estimation, and 15×15 for EPNSQ filter, worked very well. This sequence of window sizes may not be a good choice for all images. In other words, to choose a appropriate sequence of window sizes is a data dependent problem. In the process of choosing window sizes, the following guidelines must be considered:

- The window sizes for micro-texture masks are typically 3×3, 5×5, or 7×7, depending on the spatial scale size of the micro-texture features.
- The window size used for macro statistic estimation should be large enough to include a representative sample of the image texture. The coarser the texture the larger the window size should be.
- The window size used for EPNSQ filter should be smaller than the smallest region size in which we expect to have a good segmentation.

In order to further improve the segmentation performance, we postpone our classification of each pixel until spatial constraints are incorporated. We use probabilistic relaxation to enforce the spatial constraints. Each pixel is first labeled probabilistically based upon the similarity measure obtained in the feature space. From the initial probabilistic labeling, the compatibility coefficients are determined. The probabilistic relaxation is then used to update the labeling of each pixel iteratively until the stopping criteria is met. Based upon our simulation results, after applying 25 iterations of probabilistic relaxation about 30 percent of the misclassified pixels can be corrected.

In the process of using probabilistic relaxation techniques to enforce the spatial constraints, we made the following observations. First, probabilistic relaxation methods are an effective means to incorporate spatial constraints into segmentation algorithms.

Second, probabilistic relaxation provides most of the error reduction in the first 25 iterations or so, after that the error decreases slowly. Third, Peleg's scheme is usually converges much faster than RHZ's scheme. Finally, the square quadrant window shape works fairly well on regions with arbitrary shape provided that the region size is much larger than the size of the quadrant window.

Probability thresholding may be used to reduce the computation time of each relaxation iteration. The proper setting of the probability threshold is data dependent and represents a trade-off between classification accuracy and efficiency. For our data, a 0.5 percent probability threshold can reduce the amount of pixels need to be processed over 90 percent with a slightly higher misclassification rate.

From the simulation results we can draw some conclusions about the probability threshold setting. First, even a very low threshold such as 1 percent reduces the computation significantly and causes limited degradation of performance. Second, under the circumstances that the computation load is of little concern or that the clusters are not well separated in feature space then we should set the the probability threshold very low (perhaps zero) in order to take full advantage of the spatial constraints. In other words, the setting of probability threshold is a data dependent problem, therefore, data analysis is needed before we can make a proper decision. For our data, the 1 percent threshold seems to be a good choice.

7.2 An Overview of Unsupervised Textured Image Segmentation System

Figure 7.2 shows a block diagram of the unsupervised textured image segmentation system described in this thesis. As stated in chapter 2, we view the unsupervised textured image segmentation as a problem of clustering with spatial constraints incorporated. The statistics needed by the classifier may be learned from the unlabeled samples after the clusters are formed.

The K-means clustering algorithm is used to form clusters in the feature space. In order to weight each feature equally, the raw feature data is first normalized as we described in chapter 5. In our study, the clustering algorithm actually only works on a subset of the data consisting of every fourth line and every fourth pixel of the feature planes. The initial seed points were chosen based upon Ball and Hall's approach. Once the clustering process converges, the sample mean vector and sample covariance matrix for each cluster are estimated. These statistics are then used by the classifier to assign initial labeling probabilities. We applied the K-means clustering algorithm to the same test images used for supervised segmentation, and found that the supervised and unsupervised classification results are comparable. This probably occurs because the feature data sets we have on hand are fairly well-separated, thus the statistics estimated from the clustering algorithm are as good as those estimated from the training samples.

We also noted from the scatter plots shown in Chapter 5 that even though classes are reasonably well separated in some of the feature pairs, they still partially overlap in the feature space. This overlap causes mislabeling of pixels. Another question concerns

the effect of subsampling of patterns used in the cluster analysis on the class statistic estimates. We think it should not result in any noticeable differences of class statistic estimates, since neighborhood pixels are highly correlated as a result of the large size quadrant window smoothing operation.

After the class statistics are estimated by the clustering algorithm, a Bayes classifier is then used to generate initial probabilistic labeling for each pixel. Similar to the supervised segmentation system, we use the probabilistic relaxation technique to iteratively incorporate spatial constraints into our unsupervised segmentation system. Based upon our simulation results, 97.5 percent and 97 percent correct classification rates were achieved for the first and second texture mosaics, respectively.

The probability threshold procedure may also be applied to speed up the probabilistic relaxation process for the unsupervised segmentation system. The conclusions we drew for the supervised segmentation system regarding to the probability threshold setting also hold true for the unsupervised segmentation system.

7.3 Discussion

As we mentioned in chapter 2, our texture analysis approach is basically a statistical one. Roughly speaking, statistical methods are suitable for micro-textures, i.e., textures with short correlation distance. On the other hand, structural methods are more suitable for macro-textures. In the previous chapters we assumed the statistical approach matches the nature of the textures under study. However, if we want to develop a more general textured image segmentation system this assumption may not always be true. Therefore, some kind of analysis needs to be done even before we can invoke

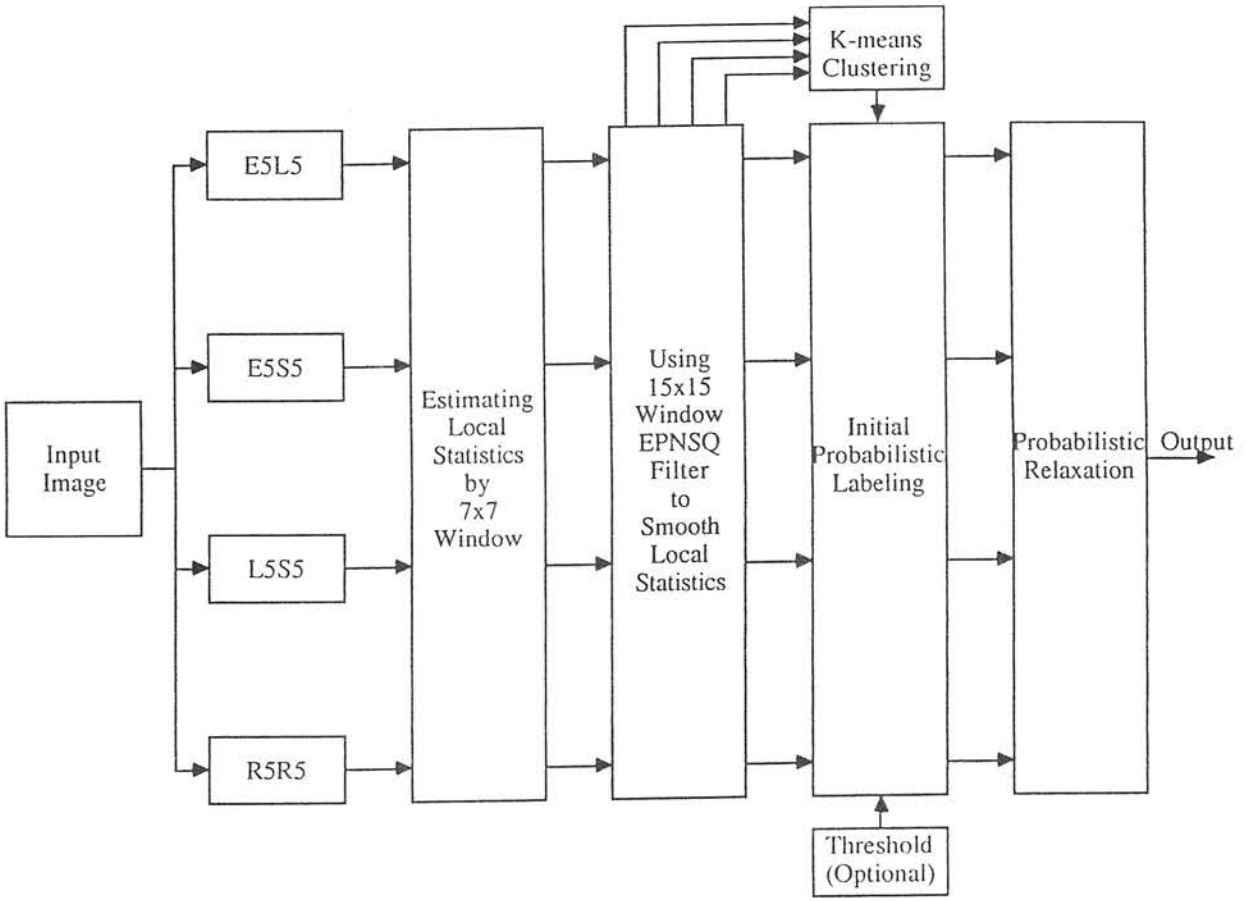


Figure 7.2: Block Diagram of Unsupervised Textured Image Segmentation System.

our algorithms. One possible way is to perform a global statistical analysis first to determine if the textures present are suitable for segmentation by a statistical approach. If not, then a structural or a structural-statistical approach may be more effective.

Although Laws' micro-texture masks are powerful in texture classification performance, they were developed empirically. More study needs to be done to see if these masks can be derived analytically, perhaps based on statistical models, or if there is a even better set of masks. Ade [58] has proposed a set of "eigenfilters" which are derived theoretically to characterize and classify textures. The performance of eigenfilters in texture classification and segmentation remains to be seen. Similar work along this line may provide us an opportunity to analytically derive a set of masks with comparable or better performance.

In our work, the classification is based upon four texture energy features selected beforehand instead of the four principal components derived from a higher dimensional feature space. In other words, the feature selection step was omitted. The degradation of performance, if any, that is caused by this omission is not clear. For the particular texture samples used in this study, the degradation may be insignificant since they are the same as those studied by Laws. We believe that the particular set of features used in this work are robust in the sense that they should produce good discrimination among other micro-texture samples having similar spatial scale size. For other types of input data, other texture features (a different set of feature extraction masks) and other features in general (gray-level, color, etc.) may be needed. Therefore, more study is needed in this area.

Chapter 8

Summary and Conclusions

The main objective of this research, as we mentioned in chapter 1, is twofold. First, to improve segmentation results; especially along the borders of regions. Second, to take into account the spatial relationship of pixels in the process of produced textured image segmentation.

8.1 Summary

In order to achieve these objectives, our approach to the textured image segmentation problem is based upon the following principles:

1. The problem of estimating texture features without destroying the boundaries between regions is similar to the problem of smoothing a noisy image. Thus, techniques used to smooth noise which do not blur edges are extended to the textured image segmentation problem to improve the accuracy along region boundaries.

2. Both global information in the feature space and the spatial organization of this data in the image space must be used.

3. The unsupervised textured image segmentation problem may be viewed as one of clustering with spatial constraints incorporated. Since there are no training samples available, the clustering technique are used to organize the patterns into clusters and then the cluster statistics are estimated. The spatial information is then enforced to improve the segmentation performance.

In response to the first principle, we described an algorithm which has taken the nonstationary nature of the problem into account during the feature extraction stage. We have shown that the EPNSQ filter can be applied to smooth texture feature estimates. We implemented this algorithm and tested it on a texture mosaic which consists of eight classes of textures. Based upon our simulation results, using the proposed feature estimation scheme can provide us not only better correct classification rates but also more accurate region borders for region sizes larger than 16×16. For large region sizes of our test mosaic, i.e. 128×128, the improvement on the correct classification rate is about 7 percent.

In response to the second principle, we described a supervised segmentation system which consists of two stages. The first stage assigns probabilistic labeling using the global information provided in the feature space. We realized that the weakness of classifying pixels based solely upon feature space distribution is that the formation of clusters in the feature space does not take into consideration the spatial distribution of points in the image. Therefore, we explored the use of probabilistic relaxation to reduce local ambiguities in the second stage. To the best of the author's knowledge, no previous attempts have been made in using probabilistic relaxation to solve the textured image segmentation problem. The proposed system has been implemented

and tested on two texture mosaics. Based upon our simulation results, after applying 25 iterations of probabilistic relaxation about 30 percent of the misclassified pixels can be corrected.

In response to the third principle, we described a unsupervised segmentation system which consists of three stages. The first stage learns class statistics from the clusters formed in the feature space. A K-means algorithm is used to form the clusters. The second stage assigns probabilistic labeling based upon the class statistics estimated from the feature space. At this stage we only use the global information provided in the feature space. The probabilistic relaxation is then used to enforce the spatial constraints in the third stage. Good experimental results for unsupervised segmentation are presented; in cases when the texture features are well-separated in feature space they are comparable to supervised results. Based upon our simulation, 97.5 percent and 97 percent correct classification rates were achieved for the first and second texture mosaics, respectively.

We also proposed the use of a probability threshold to speed up the relaxation iterations. The proper probability threshold setting is a trade off between segmentation performance and speed. With some analysis of the data, the probability threshold may be set with significant improvement in speed yet with limited degradation of performance.

8.2 Possible Future Work

Some ideas for future work were presented in chapter 7. Let us briefly reiterate here:

- For a more general textured image segmentation system, some kind of global statistical analysis needs to be done in order to determine if the nature of textures under study are suitable for segmentation by a statistical approach. If not, then a structural or a structural-statistical approach may be necessary.

- More study needs to be done to see if micro-texture masks can be derived analytically or if there is a even better set of masks. Ade [58] has proposed a set of "eigenfilters" which are derived theoretically to characterize and classify textures. Similar work along this line may provide us an opportunity to analytically derive a set of masks with comparable or better performance.

- Some investigation of the feature selection step omitted in our study is needed. For an arbitrary input textures this feature selection step and/or feature extraction step may be indispensable.

- Robustness of the proposed supervised and unsupervised segmentation algorithms needs to be investigated. In other words, we need to know how good it would work on other textured images of similar spatial scale size.

8.3 Conclusions

The main objectives of this research have been accomplished. To improve the segmentation results along the borders of regions, we have put our effort into the feature extraction stage. To utilize the information from both feature space and image space, we devised an algorithm that accomplishes it in two steps, i.e., assigning initial probabilistic labeling based upon the global information provided in the feature space and

reducing local ambiguities by probabilistic relaxation based upon the local information provided in the image space. The unsupervised textured image segmentation problem is solved by clustering with spatial constraints incorporated.

References

[1] A. Rosenfeld and A. C. Kak, *Digital Picture Processing, Vol. 2.* New York: Academic Press, 1982.

[2] R. Chellappa and A. A. Sawchuk, *Digital Image Processing and Analysis, Vol. 2.* New York: IEEE Computer Society Press, 1985.

[3] R. M. Haralick and L. G. Shapiro, "Image segmentation techniques," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 100–132, 1985.

[4] L. V. Gool, P. Dewaele, and A. Oosterlinck, "Texture analysis anno 1983," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 336–357, 1985.

[5] S. W. Zucker, "Region growing: childhood and adolescence," *Computer Vision, Graphics, and Image Processing*, vol. 5, pp. 382–399, 1976.

[6] H. Kaizer, "A quantification of textures on aerial photographs," Tech. Rep. Tech. Note 121, AD 69484, Boston Univ. Research Laboratories, Boston University, 1955.

[7] A. Rosenfeld and E. Troy, "Visual texture analysis," Tech. Rep. Tech Rep. 70-116, University of Maryland, College Park, MD, 1970.

[8] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Trans. Compt.*, vol. C-20, pp. 562–569, 1971.

[9] S. G. Carlton and O. R. Mitchell, "Image segmentation using texture and gray level," in *Proc. IEEE Conference on Pattern Recognition and Image Processing*, June 1977.

[10] L. S. Davis and A. Mitiche, "Mites: a model-driven, iterative texture segmentation algorithm," *Computer Graphics and Image Processing*, vol. 19, pp. 95–110, 1982.

[11] K. S. R. M. Haralick and I. Dinstein, "Texture features for image classification," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-3, pp. 610–621, 1973.

[12] K. I. Laws, "Rapid texture identification," *Proceedings of the SPIE*, vol. 238, pp. 376–380, 1980.

[13] K. I. Laws, *Textured Image Segmentation.* PhD thesis, University of Southern California, Los Angeles, California, 1980. USCIPI Report 940.

[14] M. Hassner and J. Sklansky, "The use of Markov random fields as models of texture," *Computer Graphics and Image Processing*, vol. 12, pp. 357–370, 1980.

[15] R. L. Kashyap, R. Chellappa, and A. Khotanzad, "Texture classification using features derived from random field models," *Pattern Recognition Letters*, vol. 1, pp. 43–50, 1982.

[16] S. Chatterjee and R. Chellappa, "Maximum likelihood texture segmentation using gaussian Markov random field models," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1985.

[17] C. W. Therrien, "Linear filtering models for texture classification and segmentation," in *Proc. 5th International Conference on Pattern Recognition*, 1980.

[18] F. S. Cohen and D. B. Cooper, "Real-time textured image segmentation based on non-causal Markovian random field models," in *Proc. SPIE 449, 3rd Intl. Conf. on Robot Vision and Sensory Control*, 1983.

[19] R. Bajcsy, "Computer description of textured surfaces," in *Proc. Third International Joint Conference on A. I.*, 1973.

[20] T. Pavlidis and S. L. Tanimoto, "Texture identification by a directed split and merge procedure," in *Proc. Conference Comp. Graphics, Pattern Recognition and Data Structure*, 1975.

[21] T. Pavlidis and P. C. Chen, "Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm," *Computer Graphics and Image Processing*, vol. 10, pp. 172–182, 1979.

[22] T. Pavlidis and P. C. Chen, "Image segmentation as an estimation problem," *Computer Graphics and Image Processing*, vol. 12, pp. 153–172, 1980.

[23] T. Pavlidis and P. C. Chen, "Segmentation by texture using correlation," in *Proc. 5th International Conf. on Pattern Recognition*, 1980.

[24] O. R. Mitchell, S. P. Lutton, and S. P. Su, "Texture image segmentation using local extrema," in *Proc. IEEE Conference on Pattern Recognition*, 1979.

[25] H. Knutsson and G. H. Granlund, "Texture analysis using two-dimensional quadrature filters," in *Proc. ICASSP 83, IEEE Conf. on Acoust., Speech, Signal Proc.*, 1983.

[26] B. J. Schachter, L. S. Davis, and A. Rosenfeld, "Some experiments in image segmentation by clustering of local feature values," *Pattern Recognition*, vol. 11, pp. 19–28, 1979.

[27] K. Deguchi and I. Morishita, "Texture characterization and texture-based image partitioning using two-dimensional linear estimation techniques," *IEEE Trans. on Computers*, vol. C-27, pp. 739–745, 1978.

[28] G. B. Coleman and H. C. Andrews, "Image segmentation by clustering," *Proceedings of the IEEE*, vol. 67, pp. 773–785, 1979.

[29] S. W. Zucker, A. Rosenfeld, and L. S. Davis, "Picture segmentation by texture discrimination," *IEEE Trans. on Computers*, vol. C-24, pp. 1228–1233, 1975.

[30] M. Pietikainen and A. Rosenfeld, "Image segmentation by texture using pyramid node linking," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-11, pp. 822–825, 1981.

[31] A. G. Weber and A. A. Sawchuk, "Multiple resolution image texture segmentation," Tech. Rep. USC Report ISG 104, University of Southern California, 1983.

[32] J. E. Bevington and R. M. Mersereau, "A maximum-likelihood approach to image segmentation by texture," in *Proc. ICASSP 84, IEEE Conf. on Acoust., Speech, Signal Proc.*, 1984.

[33] C. W. Therrien, "An estimation-theoretic approach to terrain image segmentation," *Computer Vision, Graphics, and Image Processing*, vol. 22, pp. 313–326, 1983.

[34] R. W. Conners, M. M. Trivedi, and C. A. Harlow, "Segmentation of a high-resolution urban scene using texture operators," *Computer Vision, Graphics, and Image Processing*, vol. 25, pp. 273–310, 1984.

[35] H. Derin and W. S. Cole, "Segmentation of textured images using Gibbs random fields," *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 72–98, 1986.

[36] S. Jiang and A. A. Sawchuk, "Noise updating repeated wiener filter and other adaptive noise smoothing filters using local image statistics," *Applied Optics*, vol. 25, pp. 2326–2337, 1986.

[37] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* London: Prentice-Hall International, 1982.

[38] M. J. Norušis, *Advanced Statistics Guide.* New York: McGraw-Hill, 1985.

[39] A. A. Afifi and V. Clark, *Computer-aided Multivariate Analysis.* Belmont, California: Lifetime Learning Publications, 1984.

[40] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic Press, 1972.

[41] R. Ohlander, K. Price, and D. R. Reddy, "Picture segmentation using a recursive region splitting method," *Computer Vision, Graphics, and Image Processing*, vol. 8, pp. 313–333, 1978.

[42] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-7, pp. 165–177, 1985.

[43] F. Vilnrotter, R. Nevatia, and K. Price, "Structural description of natural textures," in *Proceedings of the Fifth International Pattern Recognition Conference*, 1980.

[44] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover, 1966.

[45] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 420–433, 1976.

[46] S. Peleg and A. Rosenfeld, "Determining compatibility coefficients for curve enhancement relaxation processes," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 548–555, 1978.

[47] S. W. Zucker, E. V. Krishnamurthy, and R. L. Haar, "Relaxation processes for scene labeling: convergence, speed, and stability," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 41–48, 1978.

[48] S. Peleg, "A new probabilistic relaxation scheme," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-2, pp. 362–369, 1980.

[49] R. M. Haralick, J. C. Mohammed, and S. W. Zucker, "Compatibilities and the fixed points of arithmetic relaxation processes," *Computer Graphics and Image Processing*, vol. 13, pp. 242–256, 1980.

[50] J. O. Eklundh, H. Yamamoto, and A. Rosenfeld, "A relaxation method for multispectral pixel classification," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-2, pp. 72–75, 1980.

[51] M. R. Anderberg, *Cluster Analysis for Applications.* New York: Academic Press, 1973.

[52] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: John Wiley & Sons, 1973.

[53] G. H. Ball and D. J. Hall, "Isodata, a novel method of data analysis and pattern classification," Tech. Rep. AD 699616, Stanford Res. Inst., 1965.

[54] F. R. Fromm and R. A. Northouse, "Class: a nonparametric clustering algorithm," *Pattern Recognition.*, vol. 8, pp. 107–114, 1976.

[55] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications (abstract)," *Biometrics*, vol. 21, p. 768, 1965.

[56] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Symp. Math. Statist. and Probability, 5th, Berkeley*, 1967.

[57] D. T. Kuan, *Nonstationary 2-D Recursive Restoration of Images with Signal-Dependent Noise with Application to Speckle Reduction.* PhD thesis, University of Southern California, Los Angeles, California, 1982.

[58] F. Ade, "Characterization of textures by "eigenfilters"," *Signal Processing*, vol. 5, pp. 451–457, 1983.