USC–SIPI REPORT #111

Hierarchical Texture Segmentation Using Multiple Resolution Operators

by

Allan Guy Weber

July 1987

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 400
Los Angeles, CA 90089-2564 U.S.A.

Acknowledgements

Much of the credit for this piece of research goes to Prof. Alexander Sawchuk, who served as the chairman of my dissertation committee, as my employer, and as my friend. Without his continual guidance this work would have never been attempted or completed. I am also very grateful to the other members of my committee, Prof. Keith Price of the Electrical Engineering Department and Prof. Alan Schumitzky of the Mathematics Department, for their advice, assistance, and friendship during the course of my graduate studies.

Many people on the staff of the Signal and Image Processing Institute have provided me with help of one kind or another and I am grateful to all: to Jerene Brooks, Linda Varilla, Diana Lerner, and Hilda Marti for knowing how to get things done; to Ray Schmidt for his invaluable photographic expertise; and to Toy Mayeda for being able to build or fix anything I ever brought to him.

For keeping all the computers running, and providing interesting lunchtime conversation over the past few years, I would like to thank the staff of the USC University Computing Services: bob, karl, mark, mcooper, mead, smith, tli, valaine, wiedel, wuts, and assorted others.

Lastly, I wish to thank my parents for their love and encouragement during my time at USC. It took longer than I expected, but it was worth it.

Contents

List of Figures

List of Tables

Abstract

Texture information can be used as a basis for segmenting an image into regions. Image texture is basically a local area phenomenon that is sensitive to the size of the area. What appears as a non-textured area at one resolution level can appear as a region with distinctive texture at a different resolution. The performance of texture segmentation schemes is often highly dependent on the size of the local area operator used to generate the classification features. The size of the operators has a major impact on the performance near the boundaries between texture regions. Features based on large operators perform better overall but are highly affected by the mixing of class statistics when the operator overlaps more than texture, such as near the texture boundaries. Features based on small operators show poorer overall performance but are more likely to maintain an acceptable performance level in the boundary areas. The trade-off is between statistical accuracy of the classification versus the final accuracy of the texture region boundaries. The problem being studied here is how to combine information from texture classifiers operating at different resolutions into a segmentation process that gives acceptable performance in all areas of an image.

In this study, the nature of the mixing problem is examined and a solution is proposed based on using multiple resolution features in a hierarchical decision process. A key component of the solution is an analysis of the image data prior

to performing any classification. From this analysis, we determine the expected location in the feature space of the mixture points. Three different methods of isolating the mixture points in the feature space are proposed and tested. During the initial classification phase, image points that are within the mixture areas are left unclassified. Spatial information is incorporated into the segmentation process by the use of a local area cohesion operation. The final segmentation is based on a hierarchical decision process that uses the classification choice at both resolutions and the spatial cohesion data. Several decision processes are tested that use the information in different ways.

Chapter 1

Introduction

The ability to use a machine to interpret pictures has been the focus of a tremendous amount of work in recent years. Whether it is called machine vision or image understanding or scene analysis, the goal is the same: build a machine that can understand a picture in much the same way that a human observer can. It should be able to identify objects in the picture and determine relationships between objects. A human observer is capable of extracting a wealth of information from an image. The information can range from finding objects to avoid while walking across a room to spotting structures in photos returned from a satellite. The fact that a human can so quickly acquire, process, and analyze the visual information constantly being thrown at him gives one a great appreciation for the effectiveness of the human visual system. For a machine to be able to do some of these same tasks typically requires a large amount of computing resources and a significantly longer processing time.

One of the basic requirements for having a machine understand an image is the ability to locate regions in the image that contain pixels with some common

characteristic. This task is usually referred to as segmenting the image and is described in detail in many standard references [1], [2], [3], [4]. The common characteristic used as the basis for the segmentation can be a simple pixel property such as brightness or color. An image can also be segmented according to a a more complex, region based property such as texture. Texture is present to some degree in virtually all images and can be a key feature for use in segmenting the image. Image texture is easily recognizable but difficult to define. It exhibits a variety of properties such as coarseness, regularity, and granularity. Textures can range from the randomness of something like tree bark to the highly structured pattern of a chain-link fence. It is basically a local area phenomenon that is sensitive to the size of the area being observed. A change in resolution can transform an area of the image that appears as a non-textured region with a constant gray level into a region with distinctive texture. Changing the resolution further can make the region return to appearing as a uniform surface.

Virtually all images contain some type of texture that is used by humans to understand the contents of the image. Segmenting an image using the texture information is something a human can do rather easily in most cases. Previous work on using texture as the basis for segmentation by machines has had varying degrees of success. One area of texture segmentation where machines typically do not perform as well as a human observer is in accurately finding the location of the boundaries between textures. Texture segmentation algorithms that work with acceptable error rates in areas of homogeneous texture often perform poorly near the boundary between two or more textures. Conversely, an algorithm that works well near the boundaries often is not acceptable for use over the entire

image. A key difference between the two methods is often the size of the spatial area used to analyze the texture for making the segmentation decision. We are often forced to make a trade-off between overall accuracy of the segmentation versus the final accuracy of the texture region boundaries. Segmenting an image into texture regions is nearly identical to the problem of segmenting an image into regions of similar gray levels. The only additional work needed is the ability to transform the image into a texture domain where each pixel is represented by a number describing the texture that the pixel in the original image belongs to, rather than by a number describing the image brightness at that pixel. Obviously the problem is to find the method that can accurately map the points in the original image into the texture domain.

A key part of any texture classification/segmentation process is the generation of features to use in differentiating between the various textures. Since image texture is an area property rather than a point property, the features must be generated by analyzing the image data in a neighborhood around the pixel being segmented. The spatial area covered by the analysis operation represents the resolution of the features and has a significant effect on the performance of the features. Features generated using the data in a relatively large spatial area are more likely to give a statistically better classification since the features are based on a greater amount of data. Using a large spatial area operation when near the boundary of a texture region can lead to problems. The area will often overlap more than one region and the resulting feature is based on data from two classes. This can lead to an increase in classification errors. At best the result is a ragged boundary between regions. In many cases, a more serious problem occurs in which a false region appears between the two true regions.

To get a smooth and accurate boundary, a smaller window size must be used so that the feature generation operator more often encompasses texture from only one region. This usually makes the boundaries between the regions more accurate but can result in a higher overall error rate since the features are now based on less image data.

The problem being studied here is how to combine the results from two or more texture classification processes done at different resolutions, each with different strengths and weaknesses, into a segmentation method that has acceptable performance both near to and away from texture boundaries. The key to achieving this is in how the information from the different resolution classifiers is used. In combining the results of the classifiers, knowledge about their strengths and weaknesses in classifying near edges is used to give a final segmentation that is better than that achieved with either single resolution classifier working alone. The combining of the multiple resolution information is done in a hierarchical manner. With this approach, the segmentation accuracy is higher than that achievable using the multiple resolution features on an equal or non-hierarchical basis.

A brief overview of previous work image texture classification and segmentation is presented in Chapter 2. Chapter 3 discusses the motivation for using a multiple resolution, hierarchical segmentation process. The texture classification features and the classification algorithm being used are described in Chapters 4 and 5, respectively. Chapter 6 discusses the effects of the mixing of class statistics in generating the classification features and derives a expression for the probability density of the mixture. In Chapter 7, a set of techniques is presented for minimizing the effect of the mixture problem. Chapter 8 describes

the spatial cohesion technique used to enhance the classification accuracy. The hierarchical decision process used to make the final segmentation is discussed in Chapter 9. A summary of the work and conclusions are presented in Chapter 10.

Chapter 2

Background

2.1 Classification vs. Segmentation

Image data can be processed by classification methods or by segmentation methods or by a combination of the two. The dividing line between what is a classification process and what is a segmentation process is somewhat fuzzy. Depending on the definitions used, the methods described in the following chapters can be viewed as one or the other or both.

Pattern classification is the problem of partitioning a set of data items into groups or classes on the basis of one or more features measured from the data. For image data, the features can be something simple like brightness or color, or something more complex such as texture. The number of classes may be determined before classification starts or it may be determined by the classifier as the process proceeds. The resulting classifications are totally dependent on the data, the classification algorithm, the selection of the classes available for assignment, and on the selection of the features to be used as the basis for any

classification decisions. Changing any one of these four generally results in a different partitioning of the data.

In a classification problem, the order in which the data is processed is not highly significant. A classifier based on *a priori* information about the classes present does not change as the data is processed. The results of the classification are the same regardless of the ordering applied to the input data. For a classifier that adapts to the data as the processing proceeds, the results can be different for a different ordering of the data. However, under normal circumstances, the various results should tend to converge to a common solution. This order-independent property implies that the results of a classification of image data is not depend on any spatial information. The absolute or relative location of pixels in the image is not considered when determining the choice of class assignment on the basis of the feature data.

Segmentation is unique to image data and is a process of partitioning image pixels into groups that represent regions in the image with some common characteristic. For image understanding applications, the goal is to have each region be associated with some part of an object in the image. An object can be something physical such as a door or a lake, or it can be something more general such as a blank background behind all the foreground objects. An object may consist of more than one regions if parts of the object differ significantly in appearance or if part of the object is hidden from view, but two objects in the image should not share a single region. By its nature, the goal of segmentation is to result in regions containing pixels that have some type of spatial relationship, regardless of whether or not spatial information was used in making the partitioning decision. If spatial information is used, the distance in the image between two

pixels has a direct effect on whether or not the points should belong to the same region. Unlike classification, two pixels with identical appearance in the image may not be segmented into the same region if they are not in close proximity to each other. If spatial information is used in doing the segmentation, then the order in which the pixels are processed is important since this may imply the location of the pixels in the image.

The key difference between the classification and segmentation is that the goals of segmentation involve a spatial partitioning while classification does not. Classification by itself does not yield meaningful results about the spatial relationships between pixels in the image. Segmentation does result in spatial groupings, however it is not typically used to apply a meaningful interpretation to the grouping. This is left to a higher level process that operates on the results of the segmentation.

Both classification and segmentation are pixel-based processes rather than region-based processes and can not deal directly with image texture. The image texture must first be mapped from the pixel brightness space into a feature space where each pixel contains information about the intensity variations present in a local area around the pixel. The task of segmenting the textures into regions can be viewed as a three step process. First, a feature or set of features are generated for each pixel in the image. This has the effect of transforming the texture information in the local areas into one or more feature values. This is sometimes referred to as a transformation from the pattern space to the feature space [5]. Classification techniques are then used to transform the image from feature values to texture numbers. The classification process analyzes each pixel in the feature space and assigns it to a texture class. This effectively maps the features

space into a classification space where each pixel is a number representing the texture present at that position in the image. In the last stage, a segmentation method is used to incorporate spatial information into the final groupings of the pixels. Pixels that were assigned to the same class by the classifier may or may not be segmented in the same spatial region. Since the input data to the segmentation process is a single number representing the texture class, the image can be segmented into regions of common texture in a manner similar to the way a non-textured image can be segmented using gray levels alone.

In implementing an image texture segmentation scheme, as described above, the three processes can be merged together in varying degrees to make the division of functionality difficult to find. In the multi-resolution, hierarchical segmentation method described in the following chapters, the role of the segmentation stage has been reduced to something that can more accurately be described as a classification post-processor. No attempt is made to divide the pixels in the same texture class into spatial groupings that represent regions in the image. The segmentation process is simply used to combine spatial information with the output of the classification stage to enhance the performance of the texture classification.

2.2 Feature Generation

A great deal of the past work in texture classification and segmentation has used standard pattern recognition techniques. As with most classification tasks, the choice of features to be used is crucial for good results. Since texture is in general a local area characteristic, most features are based on some type of

operator applied to pixels in the local area surrounding the pixel being classified. We will use the term "local area operator" or "LAO" to refer to a wide variety of linear or non-linear operations that are performed in a spatial neighborhood around a pixel. The form of the output from the LAO is dependent on the operation and typically is a scalar value or a vector or a matrix. An example of an LAO is a two-dimensional convolution operation in which the data within a local area is convolved with a fixed set of number resulting in a single output value. Examples of other operations that can be applied to data in a local area include finding maximum(s) or minimum(s), measuring various statistical moments, calculating histograms, etc.

Image texture can be classified using statistical features such as mean or variance. These moments can be measured over a local area surrounding the pixel in question and the resulting features are then used for classification. One disadvantage of these features is that they lack any sense of spatial orientation. A texture with a horizontal orientation can have the same moments as one with a vertical orientation even though they look distinctly different to an observer. Information about the spatial orientation of a texture in a local area can be obtained by using a second order statistics such as the digital autocorrelation function. This can be used to measure how well an image matches a shifted version of itself. For images, this is a two dimensional function with the amount of shift horizontally and vertically as the independent variables. Two properties can usually be determined from the autocorrelation function: the coarseness of the texture and the presence of any periodicity in the texture. Most applications of this method limit the shift to a fixed number of pixels around the central point.

A problem with this technique is that most natural textures have very similar autocorrelation functions [6].

Since textures are often repeating patterns of pixels, a natural feature to use is the spatial frequencies present in the image. This can be measured optically [7], [8] or digitally via an FFT algorithm. Transforms into the frequency domain other than the Fourier such as the slant or Hadamard have also been tried [9], [10], [11]. A problem with this attempt is that to have useful transform results, the transform must be over a large number of pixels. If the transform is only applied to the small area surrounding the pixel, only high frequency information is obtained. This technique also suffer from the same shortcoming as the autocorrelation method in that it tends to perform poorly on textures that are similar. Eklundh [12] examined using only the phase component of Fourier features in classifying textures and determined that without amplitude information, the texture discriminating power of these features was low despite earlier successes in using phase information in areas such as transform coding of images.

The method of gray level cooccurrence matrices [13], [14] (also known as intensity cooccurrence matrices) is similar to the autocorrelation methods in that it works in the spatial domain. The elements of the cooccurrence matrix describe the number of times a pair of pixels with particular gray levels occur a specified distance and direction apart in an image or portion of an image. From these matrices, various features are calculated such as entropy or contrast. The cooccurrence matrix can be viewed as an approximation to the two variable probability density function. As such the features represent *ad hoc* approximations of the true entropy or contrast. If cooccurrence matrices are to

be calculated for a variety of separation distances and directions, the number of features can become rather large. However, this method has proven to be quite successful in texture classification. Results of classification tests are usually better that with autocorrelation [15]. Zucker and Terzopolous [16] used gray level cooccurrence matrices for texture classification by interpreting the matrix data as samples from a random process. Davis, Johns, and Aggarwal [17] extended the concept of the gray level cooccurrence matrix to include information about the distribution of edges in the local area. Terzopolous and Zucker [18] used these generalized cooccurrence matrices along with gray level cooccurrence matrices to classify textures.

Texture often can be broken down into primitive components. These components and the relations between them are referred to as a structural description. The orientation and placement of the primitives can be quite regular as in an image of brick wall or random as in an image of sand or gravel. Structural texture classification involves finding primitives in the image and determining the strength of the interrelationships. These can then be used as features in doing the classification [19], [20].

Laws [21], [22], [23] developed a set of texture features called "texture energy measures" based on measuring the energy in the output of a matched filtering operation on the texture. These are described in more detail in Chapter 4.

The above methods for generating features for texture classification are all similar approaches in one sense. Each of them is concerned with finding a good set of features to use in doing the classification. Once these features are found, classical pattern recognition methods such as nearest mean are used to perform

classification. In doing so, knowledge about the individual characteristics of feature operators is often not used.

2.3 Classification

Once the image texture has been transformed from the pattern space to the feature space, the implementation of the classification algorithm for image texture is not significantly different from any other type of data. Descriptions of the various types of pattern classification algorithms are described in Duda and Hart [24], Tou and Gonzales [25], and Andrews [5]. Methods for doing pattern classification can be divided into several categories depending on how much is known about the statistics of features of the various classes.

If the statistics of the data in the feature space are known or can be estimated, a classifier can be constructed based on the probability densities of the data. Estimation of the parameters of the densities using samples of the data is commonly referred to as "supervised learning," supervised implying that the true class of the samples is known. A common example of this is to assume the probability densities are Gaussian and to estimate the first and second order moments from available sample feature points. Once the probability densities have been determined, a classifier can be designed using Bayes decision theory that minimizes the cost of an incorrect classification based on a set of loss functions.

Alternative techniques known as clustering or unsupervised learning are used if the statistics of the classes are not known and no samples from known classes are available to allow estimation of the parameters. A clustering algorithm is an iterative process that attempts to determine the location of the class means

based on the data being classified. A well known example is the Isodata algorithm [26]. This technique makes repeated passes through the data, each time changing the estimate of the class means, until the locations of the class means stabilize.

Most pattern classification methods are based to some extent on using a set of discriminant functions to determine to which class to assign the point. The discriminant functions can be viewed as boundaries in the feature space between one class and another. The number of discriminant functions necessary to construct the complete boundary for a class is dependent on the data. The discriminant functions can be of any shape necessary in the feature space to implement the desired boundary.

2.4 Segmentation

Reviews of many of the techniques for doing image segmentation are provided by Ballard and Brown [1], Haralick and Shapiro [27], Fu and Mui [28], and Zucker [29]. Many image segmentation techniques can be divided into two categories: edge based and region based. Edge based systems include those that operate on lines and edges in an attempt to locate the region by finding the boundary [30], [31]. Region based systems localize the image regions either by combining small regions into larger ones (region growing) or by splitting large regions into smaller ones (region splitting). Region growing methods start with small regions on the order of a few pixels and merge them together according to a similarity criterion [32]. Region splitting methods proceed in the opposite direction, dividing large region according to a splitting criterion into separate regions [33], [34]. The

concepts of region growing and splitting were combined by Horowitz and Pavlidis [35], [36] into a technique that uses both.

Of special interest are segmentation methods that utilize information obtained at different resolutions. Most of the classification and segmentation methods studied either used features generated at the same resolution level or treated features from multiple resolutions the same. Carlton and Mitchell [37], [38] used two sizes of LAO to segment images. The large LAO was used to obtain a first level segmentation of the image. The small LAO was then used to subdivide the regions found by the large operator. This can be considered a hierarchical segmentation method since the boundaries of the first phase of the segmentation were not changed by the second phase.

Other methods of texture segmentation that make use of multiple resolution images are those based on the "pyramid" techniques described by Tanimoto and Pavlidis [39]. They use a set of successively reduced resolution images (Fig. 2.1). The pixels or nodes in the lower resolution image higher in the pyramid are functions of the "children" pixels in the higher resolution image on the next level below (Fig. 2.2). The processing of the images can either proceed in a top-down fashion, moving from the lower to higher resolution images or alternatively in a bottom-up manner going from the higher to lower resolution images.

Tanimoto and Pavlidis used a top-down procedure to isolate objects in the image. The presence of an object is first noted in a node on one level and then its shape is refined by moving to the children nodes on the next lower level where there is more resolution. The procedure is repeated down through the pyramid, refining the shape of the object.

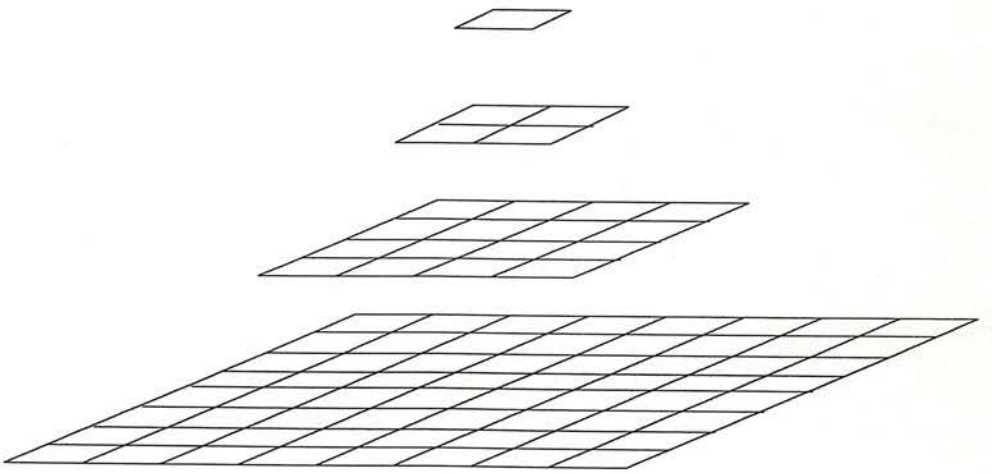


Figure 2.1: Pyramid structure with four levels

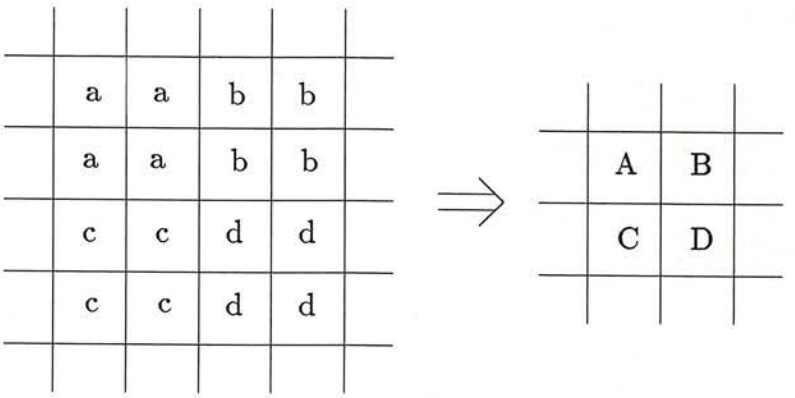


Figure 2.2: Relationship of nodes on adjacent levels

The work by Pavlidis and Tanimoto [40] and Chen and Pavlidis [41] on textured image segmentation use the split-and-merge algorithm with the pyramid data structure. Their methods are based using both the bottom-up approach for merging and the top-down approach for splitting. Each father node is a function of the four children beneath it in the larger image. Pavlidis and Tanimoto used features based on two-dimension Fourier transforms of the image brightness. From these features, a texture uniformity and texture similarity metric is computed to use in making the decision as to splitting a block into four regions or merging a group of four blocks into one. Chen and Pavlidis used features based on a simplified version of the cooccurrence matrix that does not contain directionality information. By examining the feature values of the four children nodes, a decision can be made whether to merge them into one or to split the four into different regions. After doing these operations, the cooccurrence matrices are recomputed. This process repeats recursively up to a up to a predetermined level in the pyramid. The result is an image containing square regions of various sizes. The picture is then operated upon by two other processes to group adjacent regions together using the cooccurrence matrices and to remove small regions by joining them to larger ones.

Burt, Hong, and Rosenfeld [42] performed segmentation on non-textured images with a similar bottom-up pyramid method using gray level information. Their method differs from that of Chen and Pavlidis in that the pyramid structure uses overlapping 4 by 4 blocks of pixels (Fig. 2.3). At each level a pixel is linked to one of the four candidate father nodes in the lower resolution image above it. The father nodes are then recomputed as a function of the children nodes linked to it. This process moves upward through the pyramid and then

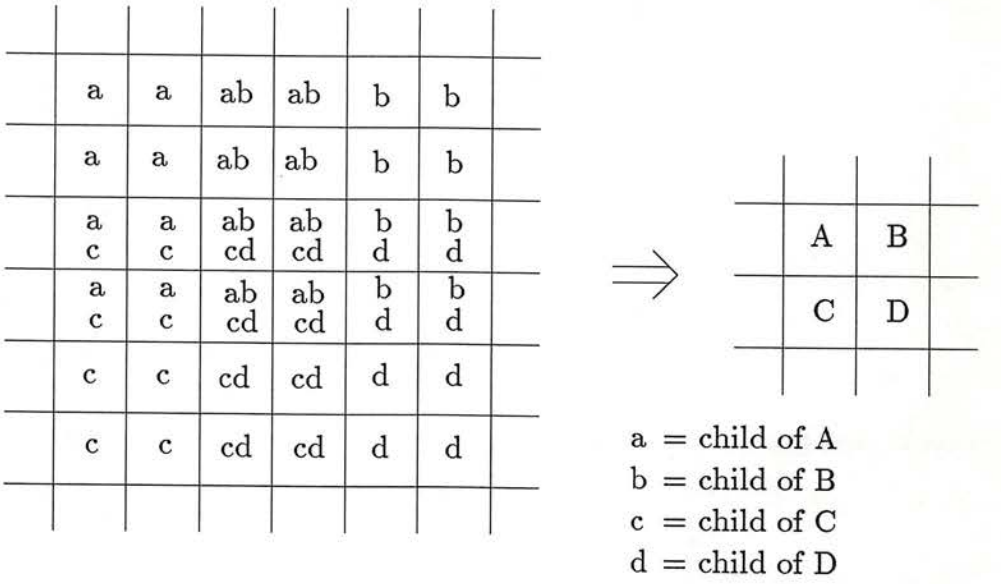


Figure 2.3: Pyramid with overlapping nodes

repeats until the links between nodes stabilize. The trees of links thus give a segmentation of the image.

Pietikäinen and Rosenfeld [43] extended the linked pyramid approach to use texture features. The texture features are based on gray level cooccurrence matrices and are initially measured in non-overlapping local areas of 8×8 pixels. The bottom level of the pyramid thus contains nodes representing each of these blocks. Their method also utilizes both bottom-up and top-down processes to improve the segmentation results.

While the pyramid techniques do make use of multiple resolution information, the process does not vary as it moves from one resolution to another. The decision process for nodes encompassing few pixels is the same as that for nodes in the low resolution image encompassing many pixels. The method for obtaining the low resolution image in the pyramid method discards a great deal of data. Reducing the resolution by a factor of 4 reduces the the number of pixels

by a factor of four. This is not the same as operating on an image with a large local area operator at each pixel to produce a new image.

Chapter 3

Multiple Resolution Segmentation

The primary objective of this research is to develop a technique for finding texture boundaries through a classification and segmentation method that utilizes feature operators of multiple sizes in ways that take advantage of their particular characteristics. The use of local area operators (LAOs) of differing size is important because the spatial size of the operator has a direct effect on the classification accuracy in various parts of an image. Thompson [44] examined the relationship between the size of the texture granularity and the size of the local area operator used to analyze it. He compared techniques for determining the minimum size of the LAO with the results of human perception of the textures using various size viewing regions. He found that "there is a well defined trade off between spatial resolution of a texture boundary and the ability to distinguish between visually similar textures." Besides using multiple size operators, a classifier should make use of the available information in an intelligent manner.

Simply using information from various size LAOs as just more features without taking into regard the size of the LAO does not generally improve performance. Our knowledge of how the various size operators respond in the different areas of a picture should be put to use. The concept of using local area operators of multiple sizes is not new. However, in the past the results were always combined without regard to the relative strengths and weaknesses of the two operators under various conditions. The work described in Chapter 2 involving the pyramids of successively reduced resolution versions of the same image does make use of different resolutions [41], [39]. However the basic algorithm is not a function of the resolution and it does not make use of the information from more than one resolution simultaneously. At any one level in the pyramid, the image data is processed at that resolution level and then the resolution is changed.

The spatial size of the local area operator used to generate the features affects the classification accuracy in two opposing manners. In an area consisting of a single, homogeneous texture, an LAO covering a larger spatial area generates more accurate features than a smaller LAO assuming the same pixel resolution was used. This is due to the nature of image texture, as discussed in Chapter 2, in that it can not be measured at a single point but must be measured over a local neighborhood. To generate feature points that can be used to accurately classify the texture, the features should be based on as much data as possible. The variations for the feature data from a 5×5 LAO are greater than from a 15×15 LAO covering a spatial area nine times as large. Accurate classification is only possible for those pixels that generate feature points that can be separated in the feature space from the feature points of other classes.

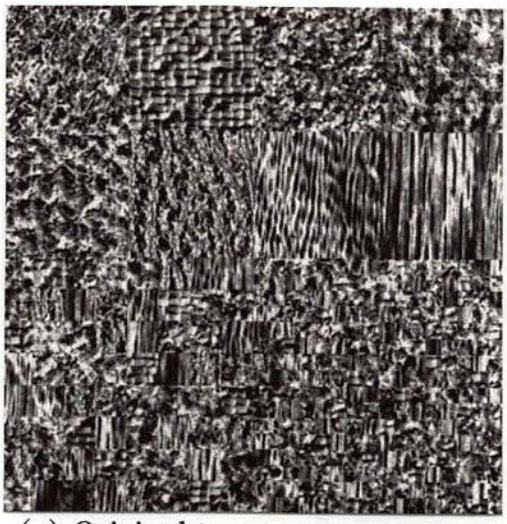
If the variance of the feature values is too large, a significant percentage of the features representing the pixels of one class are closer to the mean of another class. This results in an incorrect class assignment when using a classification algorithm based on measuring the distance from a feature point to each class mean and assigning the point to the nearest class. For this reason, the larger local area operator is preferred. If we can assume that the LAO is guaranteed to always be over an area of the image that contains a single texture, then the larger LAO will perform better than a small one. However, in a image with multiple textures, and the inherent boundaries between them, many of the possible positions of the LAO cause it to be over more than one texture. The spatial size of the LAO is directly related to the likelihood that it encompasses more than one texture. If the features for an image point are generated from data belonging to more than one texture class, the underlying statistical model is no longer that of a single class, but instead is a mixture of the statistics of the classes in present. This is discussed in more detail in Chapter 6. The probability of a classification error for points with features based on a mixture of the class statistics are higher than for points with single class features This implies that the smaller LAO should be used since it is less affected by the mixture problem. These two opposing effects result in a trade-off between high overall classification accuracy with poorly defined texture boundaries, and lower overall classification accuracy with better defined texture boundaries.

The effect of the different sizes of LAOs for generating features can be seen in the texture segmentation work by Laws [22]. The texture mosaic he used (Fig. 3.1a) contains 8 textures with identical first and second order moments in

regions that are of pixel dimensions 128×128, 32×32, and 16×16. The classification using a 31×31 LAO (Fig. 3.1b) resulted in few errors in the interiors of a region but the boundaries are not well determined. The 15×15 LAO (Fig. 3.1c) leaves many errors in the interior of the region but did a much better job of finding the region boundaries.

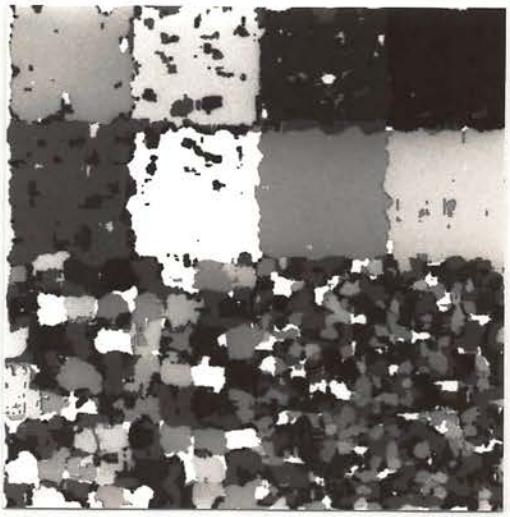
The obvious solution to this problem is to classify the points away from the texture boundaries with features generated using large LAOs, and to classify the points near the texture boundaries with features generated using small LAOs. Unfortunately, this can not be done directly since the texture boundaries are one of the things we are attempting to locate and they are not known in advance.

The alternative solution being examined is based on using the large LAO features to construct a test for detecting pixels near the texture boundaries. Pixels that are determined to have a high likelihood of not being near a texture boundary are classified normally with the large LAO features to take advantage of the higher classification accuracy possible with these features. The remaining pixels are assumed to be near a texture boundary and are classified with the small LAO features. By using the features in this manner, we are implementing a hierarchical classification. The classifier phase using the large LAO features has priority over the classifier phase using the small LAO features. In addition, the two stage process allow the results of the large LAO classification to be used in performing the second stage classification.



(a) Original texture mosaic

(b) Classification results – 31×31 features



(c) Classification results – 15×15 features

Figure 3.1: Texture mosaic used by Laws

Chapter 4

Classification Features

In any pattern classification problem, the proper selection of the features to be used in the classification algorithm is very important for obtaining the best possible results. Substantial work has been done by others to develop ways to select the best possible set of classification features for the available data. The features that we use are similar to those developed by Laws known as "texture energy measures" [21], [22], [23]. His features have been shown to work as well or better than most others on a particular set of relatively fine-grained textures (see Section 4.2) and are also relatively easy to calculate. In the work by Laws, efforts were made to use an optimum set of the features. A statistical selection procedure was used to select a set of features with the best discrimination power for a given set of input textures. For our purposes, finding an optimum set of features is of secondary importance in comparison to developing a technique for combining the results from the multi-resolution features at hand. There is some justification in using a non-optimum set of features in that any real world

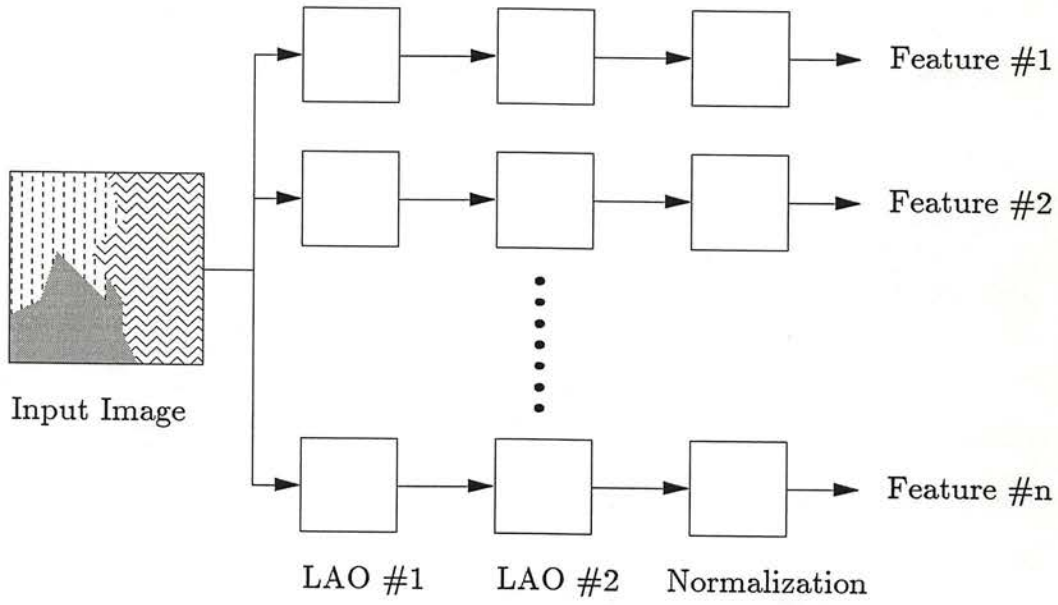


Figure 4.1: Block diagram of feature generation process

application of a texture classifier will almost assuredly be working with features that are less than perfect for the data being classified.

4.1 Texture Energy Measures

The texture energy features we use are based on a using a sequence of two local area operations (LAOs) on the image data as shown in Fig. 4.1. The first LAO to be applied to the original image is a two-dimensional convolution. The output of this step is processed by the second LAO that calculates an estimate of the standard deviation. The output of the second step is normalized to become one component of the feature set. This multi-stage process is repeated for each component of the feature set.

The first LAO convolves the image data with a fixed set of convolution masks having fixed, integer-valued weights. The term "mask" is used here and in the discussion below to signify a matrix of values used in performing the convolution. The two-dimensional convolution is performed for each pixel over the full input image. The result of the convolution at any one position in the image is used as the output image value for that position, as expressed by

$$F(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} f(x+i, y+j)h(i,j), \tag{4.1}$$

where the f represents the input image, h the convolution mask of size $2n+1$ by $2n+1$, and F the output image. In order to make this convolution technique work near the edges of the image, a border of n pixels is added to the image. The pixels in the border area are set to the mean value of the image in order to minimize the effect on the convolution output.

The convolution masks are designed to act as matched filters for certain types of quasi-periodic variations commonly found in textured images. The pixel dimensions of these masks are typically 7×7 or smaller. In most cases the sum of the elements of the mask is zero, which results in the output image having a mean of zero. The convolution masks are intended to be sensitive to visual structures such as edges, ripples, and spots. Because the micro-texture features in the image 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. Convolution masks that are not matched to the texture have smaller output variations. Thus the relevant information for texture discrimination is present in the convolution output image as the local variance of the output values.

$$
\begin{array}{lcl}
\text{E5} & = & [\ -1 \quad -2 \quad 0 \quad 2 \quad 1\ ] \\
\text{L5} & = & [\ 1 \quad 4 \quad 6 \quad 4 \quad 1\ ] \\
\text{R5} & = & [\ 1 \quad -4 \quad 6 \quad -4 \quad 1\ ] \\
\text{S5} & = & [\ -1 \quad 2 \quad 0 \quad 2 \quad -1\ ]
\end{array}
$$

Figure 4.2: Vectors used to generate convolution masks

In Laws' work, the first stage LAO for calculating the texture energy features used several different convolution mask at sizes of 7×7, 5×5, and 3×3. The output of this operation was processed by the second stage LAO, described below, and normalized. A subsequent feature selection process resulted in using features based on only four of the first stage LAOs, all of size 5×5. These were referred to as the E5L5, R5R5, E5S5, and L5S5 masks. The E5, R5, S5, and L5 terms refer to the 5-element vectors shown in Fig. 4.2 that are used to create the 5×5 LAOs by taking direct (outer) products of the two vectors. In addition, the L5L5 mask is used for the final normalizing process. Figure 4.3 shows the convolution masks used. The four LAOs selected were found by Laws to be the most important for classifying the Brodatz textures he was using. We are using the same four masks for the features generated in this study.

The second step in calculating the features involves applying a larger LAO to the first stage data to measure the sample variance (or some approximation) within local areas, either at each pixel or over a sub-block of the image. It is the size of this LAO that is changed to give the multiple resolution data for the final classification. Typically, the spatial size of the LAO for measuring the variance is on the order of 15×15 or 31×31 pixels.

The variance in the local area of the filtered image can be measured in a variety of ways. One method of computing local variance within a $2n + 1$ by

-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

(a) E5L5

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

(b) R5R5

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

(c) E5S5

-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

(d) L5S5

Figure 4.3: Convolution masks

$2n+1$ pixel area centered at point (x,y) is given by

$$\sigma^2(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} (f(x+i,y+j) - m(x,y))^2, \tag{4.2}$$

where the local mean, m, at point (x,y) is given by

$$m(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} f(x+i,y+j). \tag{4.3}$$

There are also variations on this equation that provide different expressions for the local variance. Kuan [45], [46] used a method in which the local mean, m, in Eq. (4.2) is that from the area centered on the point $f(x+i,y+j)$ rather than around the point $f(x,y)$. The local variance equation thus becomes

$$\sigma^2(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} (f(x+i,y+j) - m(x+i,y+j))^2. \tag{4.4}$$

If the convolution calculation is performed using LAOs having 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 local area

$$\sigma^2(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} f(x+i,y+j)^2. \tag{4.5}$$

Experimental examination of the statistics of some filtered images show that this zero mean assumption is justified.

The final feature values use the standard deviation, σ, rather than the variance of the data. This implies that the features can be calculated with an approximation for the standard deviation such as averaging the absolute values of the points in the local area

$$\sigma(x,y) = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} |f(x+i,y+j)|. \tag{4.6}$$

This method produces some savings in computation over Eqs. (4.2) and (4.5). The differences in the final classification performances using the various measurements in Eqs. (4.2), and (4.4)-(4.6) have turned out to be essentially negligible. Since we are mainly interested in the relative performance at multiple resolutions, the performance differences from one method to another are of no great consequence. We have used the sample standard deviation method (Eq. 4.2) in the work to be described.

The output from the second LAO is a non-normalized feature image. The final step in creating the feature set is to normalize the feature data. This makes the feature data independent of changes in brightness and contrast in the input image. The normalizing factors are derived in much the same way as the feature points described above using two LAOs. The first LAO is a convolution using a small (5×5) mask. However in this case the mask is not zero sum, resulting in a output image that does not have a zero mean. The output of this operation is similar to that of a low-pass filter. A second LAO is again used to measure the standard deviation, σ, of the output image. Since the image does not have a zero mean, Eqs. (4.2) or (4.4) must be used instead of Eqs. (4.5) or (4.6). The output of the second LAO is a processed image that is used to normalize the data in the feature images on a pixel-by-pixel basis. The pixel values in the feature image are normalized by dividing them by the corresponding pixel values in the normalizing image. Any difference in multiplicative gain from one image to another cancels since the final feature images are now a ratio of standard deviations. As mentioned before, the mean of all the features is zero because the convolution masks used by the first stage LAO to produce the features are

all zero sum. This has the desired side effect of canceling any additive bias or brightness change in the input image.

The texture energy features calculated by Laws were based on using several different first stage LAO convolution masks and one size of second stage LAO standard deviation measurement. In order to reduce the dimensionality of the classification process, a principal component transformation [3] was then used by Laws to allow selection of the a working subset of feature data containing the most significant transformed features. This process has been simplified in our case. Based on Laws' conclusion, a subset of the possible convolution masks were selected beforehand (Fig. 4.3) and used without performing the principal component transformation.

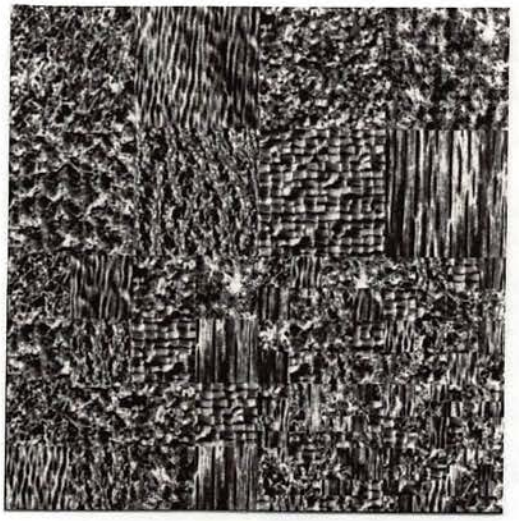
As described above, the four masks selected were those indicated by Laws to be the most important for classifying the Brodatz textures he was using. However, the performance that can be obtained using only these convolution masks is certain to be inferior to that obtained using an equal number of features that are optimum linear combinations of many features. For the purpose of this research, the relative performance of the classification at various resolutions is more important than the absolute classification accuracy. It is important to remember that, unlike Laws' work, this is not a study in doing feature selection but rather a study in how to best utilize existing features from various resolutions. It is very likely that the results that follow are not as good as they might be if more effort went into selecting an optimum feature set for the classification.

4.2 Test Data

The primary test image used in this study is a texture mosaic (Fig. 4.4a) composed of eight different texture samples taken from the Brodatz texture book [47]. The Brodatz book consists of photographs of a variety of textures, both small and large grained. Photographic prints of several of the textures were obtained from the author. These were digitized as 512×512 pixels images with 256 gray levels per pixel, followed by a histogram equalization process.

The mosaic is a 512×512 image with 8-bits (256 gray levels) per pixel. It consists of eight textures which are numbered from one through eight and listed for reference in Table 4.1. Rectangular regions of size 128×128 pixels were extracted from the 512×512 scanned images for these eight textures and combined into a single mosaic image following the diagram in Fig. 4.4b. All eight textures are present in the image in squares of size 128×128, 64×64, 32×32, and 16×16. The texture data in the smaller regions was obtained by extracting data from the larger 128×128 regions. This mosaic is slightly different than the mosaic used by Laws (Fig. 3.1a) in that is has regions of four different sizes rather than only three. The mosaic Laws used did not have 64×64 regions.

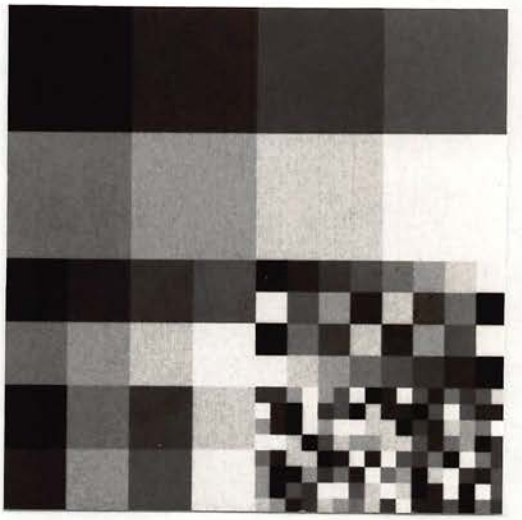
In the segmentation results that follow, the individual textures are shown in the images with each class assigned a gray level. Figure 4.4c shows the gray levels used to display each of the eight textures.



(a) Texture mosaic image

1		2		3	4
5		6		7	8
1	2	3	4		
5	6	7	8		
1	5	3	7		
2	6	4	8		

(b) Numerical texture map



(c) Gray-level texture map

Figure 4.4: Texture mosaic image and reference maps

Reference number	Texture	Brodatz page number
1	Grass	D9
2	Water	D38
3	Sand	D29
4	Wool	D19
5	Pigskin	D92
6	Leather	D24
7	Raffia	D84
8	Wood	D68

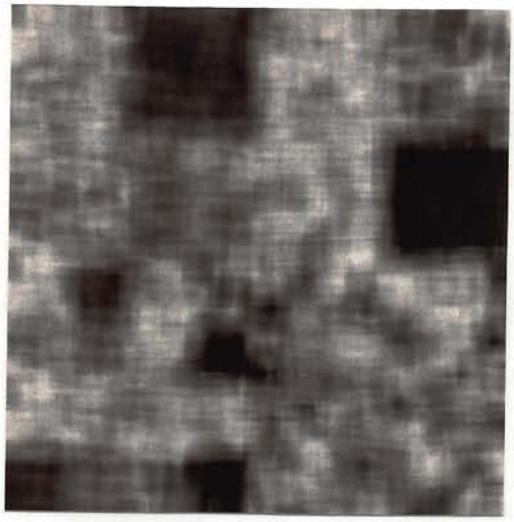
Table 4.1: Components of the texture mosaic

4.3 Experimental Results

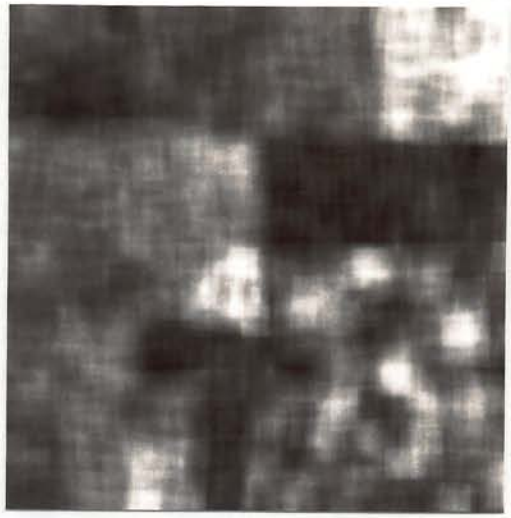
The resulting features generated for the texture mosaic are shown in Figs. 4.5 and 4.6. These represent the results of using the four first stage LAO convolution masks (Fig. 4.3) described in the previous section, with the 31×31 and 15×15 standard deviation operation of Eq. 4.2, followed by the normalizing process. The images shown have been scaled to an eight-bit output range (0-255) for viewing. For the purposes of doing the computations in this study, all the feature data used for the classification are stored as floating point numbers.

The same feature data was also used to estimate the parameters for the *a priori* statistics for each individual texture. The first and second order statistics for each of the eight textures was measured using the feature data in the largest (128×128) regions of the mosaic. The exact area used to compute the statistics within the 128×128 regions was selected to be the largest portion of each region that was not affected by the boundary effects of the neighboring regions. Thus, if the two LAOs are a 5×5 convolution operation and a 31×31 standard deviation operation, a 94×94 region in the center of the 128×128 region is not

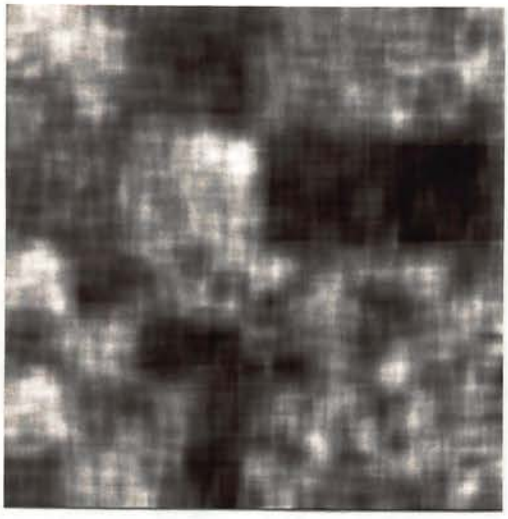
affected by the edges of the images or the neighboring regions. The data in this area is used as the sample data for training the classifier. This results in a 4-dimensional mean vector and a 4×4 covariance matrix for each of the eight textures. These are used as the first and second order moments in the *a priori* statistics for the classification algorithm described in the next chapter.



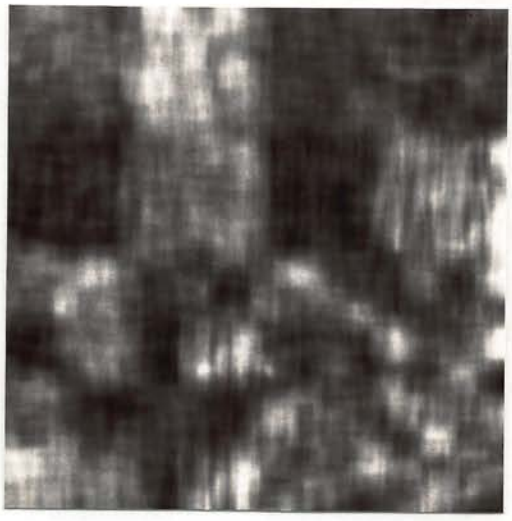
(a) E5L5



(b) R5R5

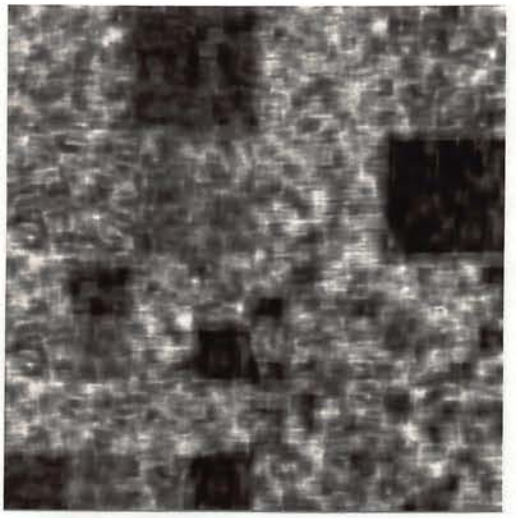


(c) E5S5



(d) L5S5

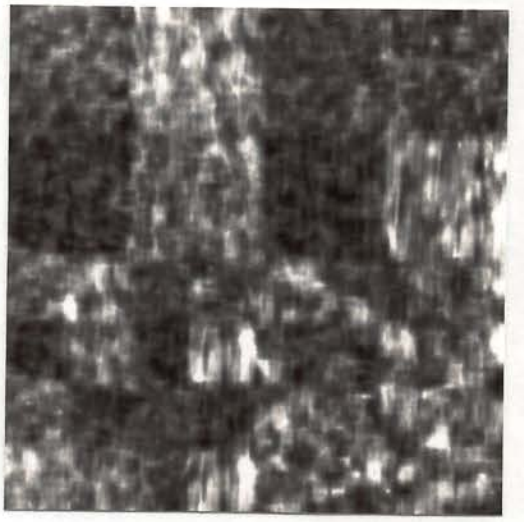
Figure 4.5: 31×31 normalized features



(a) E5L5

(b) R5R5

(c) E5S5



(d) L5S5

Figure 4.6: 15×15 normalized features

Chapter 5

Classification Algorithms

A key component of the texture segmentation process is the technique for doing pattern classification. Numerous algorithms for pattern classification are available. The selection of an algorithm is determined to a large extent by the amount and type of information that is known about the data to be classified. Classification techniques may be divided into two groups: ones that rely on some amount of prior knowledge about the data to be classified and those that operate without any prior knowledge. The amount of information available determines the type of classifier that can be used. The ideal situation is to know *a priori* the complete statistics for all classes, either directly or by estimating the parameters. If this is the case, the classification algorithm may be based on Bayes decision theory which ensures minimum misclassification error [5].

5.1 Bayes Classifier

For the case in which the *a priori* probabilities of each class and the probability densities of the data from each class are known, an optimum statistical classifier

based on Bayes decision theory can be designed that minimizes the cost of making an error. The cost is based on a set of values that describe the loss for making each potential type of error. For a classification problem involving N classes and one-dimensional data, the classes are denoted by $\omega_i, i = 1, \ldots, N$. In the equations that follow, we assume that i takes on values from 1 to N unless otherwise specified. For each data point, x, we must decide which of the N classes to assign the point to. The *a priori* probabilities of each class, $P(\omega_i)$, and the conditional densities, $p(x|\omega_i)$, of each class are known. Using Bayes rule [48], we can relate the *a priori* and conditional probabilities to the *a posteriori* probability:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \tag{5.1}$$

where

$$p(x) = \sum_{j=1}^{N} p(x|\omega_j)P(\omega_j). \tag{5.2}$$

The Bayes classification is performed by calculating the value of $P(\omega_i|x)$ for each of the N classes and assigning the point x to the class with the highest *a posteriori* probability. When the *a posteriori* probability, $P(\omega_i|x)$, is used in this fashion for doing classification, it is referred to as a discriminant function. For a N-class problem, we need N discriminant functions. The classification consists of assigning the point to the class with the largest discriminant value [24], [25].

The Bayes classifier algorithm can be extended to multi-dimensional data. In this case, Eq. (5.1) is replaced by

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{5.3}$$

where

$$p(\mathbf{x}) = \sum_{j=1}^{N} p(\mathbf{x}|\omega_j)P(\omega_j). \tag{5.4}$$

In Eq. (5.3), the data is represented by the d-dimensional vector, $\mathbf{x}$. The equation can be simplified considerably if we assume equal *a priori* probabilities ($P(\omega_i) = P$) and if we assume that the conditional densities are multivariate Gaussian

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)} \tag{5.5}$$

where

$$\boldsymbol{\mu}_i = E[\mathbf{x}|\omega_i] \tag{5.6}$$

$$\Sigma_i = E[(\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^t|\omega_i]. \tag{5.7}$$

Under these assumptions, Eq. (5.3) can be written as

$$P(\omega_i|\mathbf{x}) = \left(\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)}\right)\frac{P}{p(\mathbf{x})}. \tag{5.8}$$

Since the values of P and $p(\mathbf{x})$ are the same for all classes, those terms can be removed from the expression for $P(\omega_i|\mathbf{x})$. Taking the logarithm of each side yields the discriminant function

$$g_i(\mathbf{x}) = -ln(P(\omega_i|\mathbf{x})) \tag{5.9}$$

$$= \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \frac{1}{2}ln|\Sigma_i| + \frac{d}{2}ln(2\pi). \tag{5.10}$$

Constant terms can be removed from the above expression since they are present in all the discriminant functions and do non affect the result of any comparisons between the conditional densities of the classes. This reduces the discriminant function to

$$g_i(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_i)^t\Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + ln|\Sigma_i|. \tag{5.11}$$

This results in a set of discriminant functions (one for each class) that can be used to perform the Bayes classification of the unknown points. Note that by relating $g_i(\mathbf{x})$ to the negative log of the *a posteriori* density, we have reversed the sense of the test applied to the discriminant functions. Classification of an unknown point is done by evaluating $g_i(\mathbf{x})$ for each class and assigning the point to the class that results in the lowest value of $g_i(\mathbf{x})$.

In Eq. (5.11), the first part of the expression is the square of a general distance function called the Mahalanobis distance, defined by

$$D_M^2(\mathbf{x}, \boldsymbol{\mu}_i) = (\mathbf{x} - \boldsymbol{\mu}_i)^t \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i). \tag{5.12}$$

The Mahalanobis distance is a distance function that compensates for different variances of the data in the d dimensions. If d is one, the Mahalanobis distance is in units of standard deviations of the data. The same concept applies for higher dimensions.

If we assume that the covariance matrices for all classes are proportional to the identity matrix ($\Sigma_i = \sigma^2 I$) then the Mahalanobis distance is proportional to the Euclidean distance and the expression for the discriminant functions reduces to the Euclidean distance from the point to the class mean. This implies that the point is being classified to the nearest mean in the d-dimensional Euclidean feature space. Without the above assumption on the covariance matrices, the feature space is no longer Euclidean but is warped by the differing covariances of each component of the feature vector. However the fundamental classification operation can still be viewed as a comparison of the distances between the point and the various class means. In the more general case of Eq. (5.11), the distance is essentially the Mahalanobis distance plus a class dependent bias given by

$ln|\Sigma_i|$. This distance can be referred to as the "Bayes distance", D_B where

$$D_B^2(\mathbf{x}, \boldsymbol{\mu}_i) = g_i(\mathbf{x}) \tag{5.13}$$

$$= D_M^2(\mathbf{x}, \boldsymbol{\mu}_i) + ln|\Sigma_i|. \tag{5.14}$$

This measure of how close the point $\mathbf{x}$ is to the mean of class i is not a true distance metric since $D_B(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i) \neq 0$, due to the additive bias.

5.2 Classification Results

The Bayes classifier described above is applied to each individual pixel in the image using the the texture mosaic features generated with the 31×31 and 15×15 standard deviation LAO shown in Figs. 4.5 and 4.6, respectively. The results are shown in Table 5.1. Also shown are the results of classifying the mosaic using both the 31×31 and 15×15 features simultaneously as 8-dimensional feature data. The values in the table indicate the percentage of correctly classified points in the image, for both the full image and for the sub-images consisting of various sized mosaic blocks.

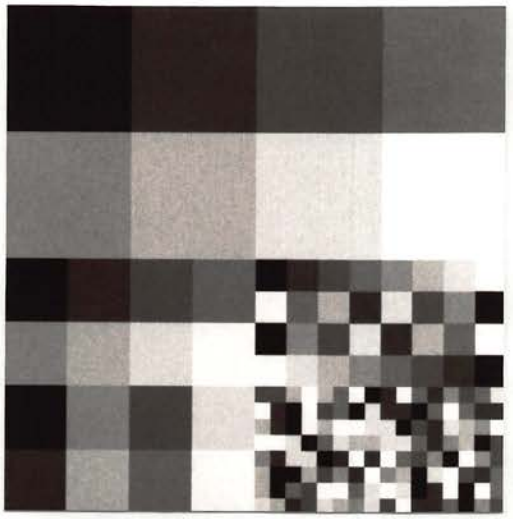
The pictorial classification results are shown in Fig. 5.1. A key point to notice is that using both the 31×31 and 15×15 features simultaneously did not increase the accuracy of the classification significantly over that achieved with just the 31×31 features.

While image textures can be classified using the above techniques, the results indicate that there are a few inherent problems. The most significant problem is the presence of an incorrect class along the boundary between two other classes. This is most obvious in Fig. 5.1b in the areas between classes 1 and 2 and between

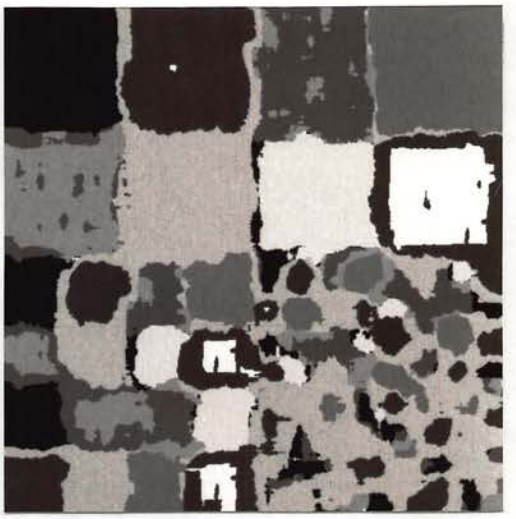
	31×31	15×15	31 × 31+ 15×15
Overall	68.1	69.9	68.5
128×128	83.2	78.2	83.4
64×64	70.2	71.6	70.8
32×32	48.4	58.1	48.5
16×16	23.5	44.9	24.2

Table 5.1: Results of Bayes classification

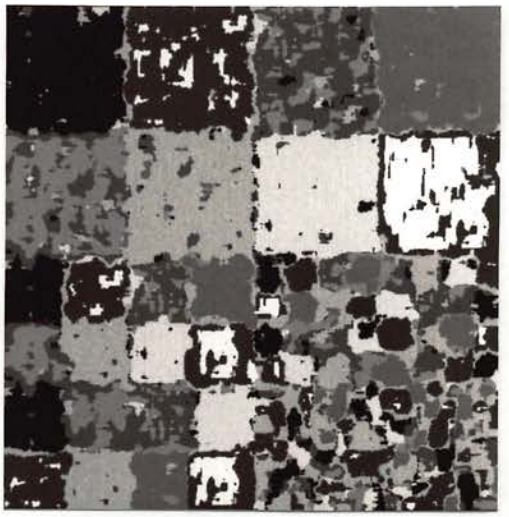
classes 7 and 8. This is a more serious problem than if the points in these areas were incorrectly classified to the wrong class of one of the two on either side of the boundary. If this false class did not appear, our main concern would be to find the boundary between the two classes. However, with the presence of the third class, we can never be sure whether or not the class is legitimate or an artifact of the classification technique. A secondary problem concerns the distribution of the incorrectly classified pixels. In the above example, the incorrectly classified points should have been distributed into all 8 classes in approximately equal number, but it is apparent that most of the errors resulted in a pixels being assigned to class 6. This problem can be seen in the lower right portion of Fig. 5.1b where the texture blocks are of size 15×15. In this part of the image, a large number of pixels have been classified to class 6 (leather). This indicates that class 6 tends to be a default class for points classified incorrectly. In the next chapter we examine the origin of this phenomenon.



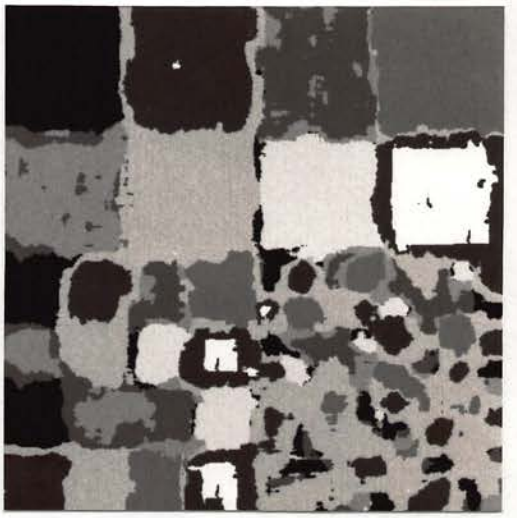
(a) Gray-level texture map



(b) Classification results – 31×31 features



(c) Classification results – 15×15 features



(d) Classification results – 31×31 and 15×15 features

Figure 5.1: Classification results for texture mosaic image

Chapter 6

Mixture Densities

The success of any pattern classification system depends greatly on the features available. No pattern classification algorithm can compensate for features that cannot separate the classes. A basic assumption in most statistical pattern classification schemes is that each unknown sample point data point is a member of exactly one of the possible classes. This implies that the features for each data point are based on exactly one statistical model and that the features for all points in the class are based on the same statistical model. Given a set of features calculated from the data points, a pattern classifier tries to match them to the features of the known classes. Using some type of similarity measure, the most likely class for membership is decided on and the sample point is classified to that class. An exception to this is the classification methods based on fuzzy sets [49], [50], [51]. A fuzzy set is one whose members have an associated probability describing the likelihood of membership in the class.

6.1 Why the Mixture Problem Exists

The assumption about data points being from only one class breaks down somewhat when classifying image texture. Image texture is not a point phenomenon; it is a local area phenomenon in which the texture at a point in the image can only be determined by analyzing a neighborhood surrounding the point in question. For this reason, the features used in texture classification schemes are based on a measurement made within a local area. An inherent problem with this approach in analyzing an unknown image is that the local area may encompass a single texture or multiple textures. The local area operators (LAOs) used to calculate the features often overlap more than one texture in the image. In this situation, the resulting feature values do not have statistics matching just one of the classes. Instead, the feature statistics are a combination or mixture of the statistics of the classes that are present in the area encompassed by the LAOs. This implies that sample points close to the boundaries between regions of different textures generate features that are spatially nonstationary and a function of spatial position in the image. This is true even though the statistics of the individual classes may be spatially stationary.

The statistical model for these nonstationary features is a mixture density as described by Duda and Hart [24],

$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{N} p(\mathbf{x}|\omega_j, \Theta_j) P(\omega_j), \tag{6.1}$$

where N is the number of classes, $\Theta = (\Theta_1, \ldots, \Theta_N)$. Θ_i is the parameter vector for the corresponding component density and $P(\omega_j)$ are mixing parameters that

satisfy

$$\sum_{j=1}^{N} P(\omega_j) = 1. \tag{6.2}$$

Here the mixture density is a weighted sum of the component densities. The fact that the mixing parameters change as we move from one part of the image to another is what makes the feature statistics nonstationary.

The presence of a mixture density has a significant effect on texture classification accuracy. The degree to which this problem affects classification accuracy depends mostly on how much a mixture of two or more classes resembles another class. For example, assume we are attempting to classify the textures shown in Fig. 6.1 by using two features: 1) a measure of the number of horizontal edges and, 2) a measure of the number vertical edges within a window. Class 1 consists of only horizontal edges, class 2 consists of only vertical edges, class 3 contains both horizontal and vertical edges, and class 4 has neither type of edges. The features that are generated along the boundary between classes 1 and 2 show both horizontal and vertical edges, much the same as the features generated for class 3. We can expect that many of the points in this boundary region will be incorrectly classified to class 3. Other boundaries, such as between classes 3 and 4 are not affected in this way because that mixture does not resemble any other class.

The mixture problem described above affects the generation of feature data from the image data. For natural images the problem is also present in the original digitizing process. The digitization of the image involves measuring the average brightness level within small, local areas. If a local area overlaps more than one texture, the resulting pixel value is based on a mixture of the textures present in the area. In the analysis that follows, the presence of the mixture

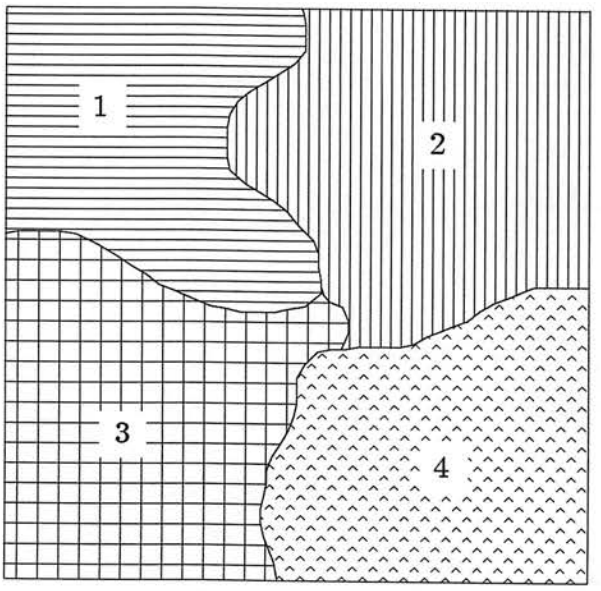


Figure 6.1: Sample texture image - 4 classes

problem in the digitizing process is ignored. This is justified since the size of the LAO used to generate the features is significantly larger than the size of the aperature used to digitize the image. With the sizes of LAOs we are using, only a small fraction of the pixels in the LAO are affected by the mixture problem from the digitizing process. The mixture problem in the digitizing process is not present in an artificially constructed mosaic image, such as the test image shown in Fig. 4.4a, since the individual texture regions are all extracted from larger images containing a single texture.

6.2 Derivation of the Mixture Density

The approximate statistics of the feature points from a mixture of classes can be derived in the following manner. The first step involves determining the

conditional statistics of an individual feature point for a given proportion of classes present in the local area used to calculate the feature value. The second step is to determine the statistics of the collection of all features points in image as a function of the proportion of the total feature points that are affected by the mixture phenomenon.

6.2.1 Density of an Individual Feature Point

Assume we have a second stage LAO of the type described in Chapter 4 that encompasses M feature points that have been processed by the first stage LAO. Also assume that the values calculated in the first stage are derived using a point process or with an LAO small enough that any mixing of pixels from different classes can be ignored. This means that the second stage LAO only contains points from one of two classes. For purposes of calculation, assume that the feature values from the first stage LAO are of dimensionality one and Gaussian distributed with means μ_i, and covariance σ_i^2

$$\begin{aligned} x_1 &\in N(\mu_1, \sigma_1^2) \\ x_2 &\in N(\mu_2, \sigma_2^2). \end{aligned} \tag{6.3}$$

The following derivation of the density of the feature points can easily be extended to multi-dimensional features.

The second stage LAO is used to measure the sample variance of the processed image points. The sample variance values are the feature values that are used for doing the classification. Let the sample variance be given by

$$y = \frac{1}{M-1} \sum_{i=1}^{M} (x^{(i)} - \bar{x})^2 \tag{6.4}$$

where $\bar{x}$ is the sample mean

$$\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x^{(i)}. \tag{6.5}$$

Let m_1 and m_2 be the number of points in the LAO from classes 1 and 2 respectively. Since there are only these two classes in the LAO, then $m_1 + m_2 = M$. The points from the two classes are denoted as

$$\begin{aligned} x_1^{(i)}, i &= 1, \ldots, m_1 \\ x_2^{(j)}, j &= 1, \ldots, m_2 \end{aligned} \tag{6.6}$$

Assume the means of the two classes are equal ($\mu_1 = \mu_2$), implying

$$E(x_1^{(i)}) = E(x_2^{(j)}) \tag{6.7}$$

for $i = 1, \ldots, m_1$, and $j = 1, \ldots, m_2$. Then the sample mean of class 1 points, $\bar{x}_1$, and the sample mean of class 2 points, $\bar{x}_2$, are both be approximately the same as $\bar{x}$, the sample mean of all the points in the LAO.

The output of the second LAO that is measuring the sample variance of the feature points within the LAO is then given by

$$\begin{aligned} y &= \frac{1}{M-1}\left(\sum_{i=1}^{m_1}(x_1^{(i)} - \bar{x}_1)^2 + \sum_{j=1}^{m_2}(x_2^{(j)} - \bar{x}_2)^2\right) \\ &= \frac{m_1 - 1}{M-1}\left(\frac{1}{m_1 - 1}\sum_{i=1}^{m_1}(x_1^{(i)} - \bar{x}_1)^2\right) + \frac{m_2 - 1}{M-1}\left(\frac{1}{m_2 - 1}\sum_{j=1}^{m_1}(x_2^{(j)} - \bar{x}_2)^2\right) \\ &= \frac{m_1 - 1}{M-1}y_1 + \frac{m_2 - 1}{M-1}y_2 \end{aligned} \tag{6.8}$$

where y_1 and y_2 are the sample variances of the points within the LAO from the classes 1 and 2, respectively. This implies that the sample variance of all the data in the LAO is a weighted sum of the individual sample variances.

Since the distributions of x_1 and x_2 are Gaussian, then the distributions of the sample variances, y_1 and y_2, are Chi-squared (χ^2) with m_1 and m_2 degrees of freedom, respectively [52]. The mean of the sample variances is thus the variance of the samples

$$E(y_i) = \mu_{y_i} = \sigma_i^2 \tag{6.9}$$

and

$$E(y) = \frac{m_1 - 1}{M - 1}\sigma_1^2 + \frac{m_2 - 1}{M - 1}\sigma_2^2. \tag{6.10}$$

The variance of the sample variance is given by

$$\begin{aligned}
\sigma_y^2 &= E(y - E(y))^2 && (6.11)\\
&= E\left(\frac{m_1 - 1}{M - 1}y_1 + \frac{m_2 - 1}{M - 1}y_2 - \left(\frac{m_1 - 1}{M - 1}\sigma_1^2 + \frac{m_2 - 1}{M - 1}\sigma_2^2\right)\right)^2 && (6.12)\\
&= E\left(\frac{m_1 - 1}{M - 1}(y_1 - \sigma_1^2) + \frac{m_2 - 1}{M - 1}(y_2 - \sigma_2^2)\right)^2 \\
&= \left(\frac{m_1 - 1}{M - 1}\right)^2 E(y_1 - \sigma_1^2) + \left(\frac{m_2 - 1}{M - 1}\right)^2 E(y_2 - \sigma_2^2) \\
&\quad + \frac{(m_1 - 1)(m_2 - 1)}{(M - 1)^2}E((y_1 - \sigma_1^2)(y_2 - \sigma_2^2)). && (6.13)
\end{aligned}$$

Since the points in class 1 and class 2 are independent, the last term is equal to zero. The resulting expression for the variance of the feature points is

$$\sigma_y^2 = \left(\frac{m_1 - 1}{M - 1}\right)^2 \sigma_{y_1}^2 + \left(\frac{m_2 - 1}{M - 1}\right)^2 \sigma_{y_2}^2. \tag{6.14}$$

For $m_1 \gg 1$ and $m_2 \gg 1$, the χ^2 distributions tend to be Gaussian due to the central limit theorem. Let α represent the proportion of points in the mixture from class 1

$$\begin{aligned}
\alpha &= \frac{m_1}{M} \approx \frac{m_1 - 1}{M - 1} && (6.15)\\
1 - \alpha &= \frac{m_2}{M} \approx \frac{m_2 - 1}{M - 1}.
\end{aligned}$$

The distribution of the output from second stage LAO is thus given by

$$p(y|\alpha) = N(\alpha\mu_{y_1} + (1-\alpha)\mu_{y_2}, \alpha^2\sigma_{y_1}^2 + (1-\alpha)^2\sigma_{y_2}^2) \tag{6.16}$$

where $\mu_{y_1} = \sigma_1^2$ and $\mu_{y_2} = \sigma_2^2$. This represents the conditional distribution of the individual feature values given the relative proportions of the two classes present in the second stage LAO.

6.2.2 Density of a Collection of Feature Points

From the density of the individual feature points derived above, we can determine the density of the collection of feature values in the image for a particular configuration of regions in the image. Let us assume we have applied the window described above to various positions in an image that contains two textures separated by a boundary. This has the effect of calculating the feature values, y, for a variety of values of α.

Let K be the total number of positions the LAO is placed in to calculate a value of y. Depending on the position of the LAO, it may or not be affected by the boundary. Let k_1 be the number of positions in which only class 1 is in the LAO and let k_2 be the number of positions in which only class 2 is in the LAO. This implies that $k_1 + k_2 \leq K$. Let β_1 and β_2 represent the fraction of the positions in which only one of the classes is in the LAO

$$\begin{aligned} \beta_1 &= \frac{k_1}{K} \\ \beta_2 &= \frac{k_2}{K} \end{aligned} \tag{6.17}$$

where $\beta_1 + \beta_2 \leq 1$.

The value of α in the expression for the variance density can be expressed as a deterministic function of a discrete random variable, i. Let $\alpha(i)$ be the

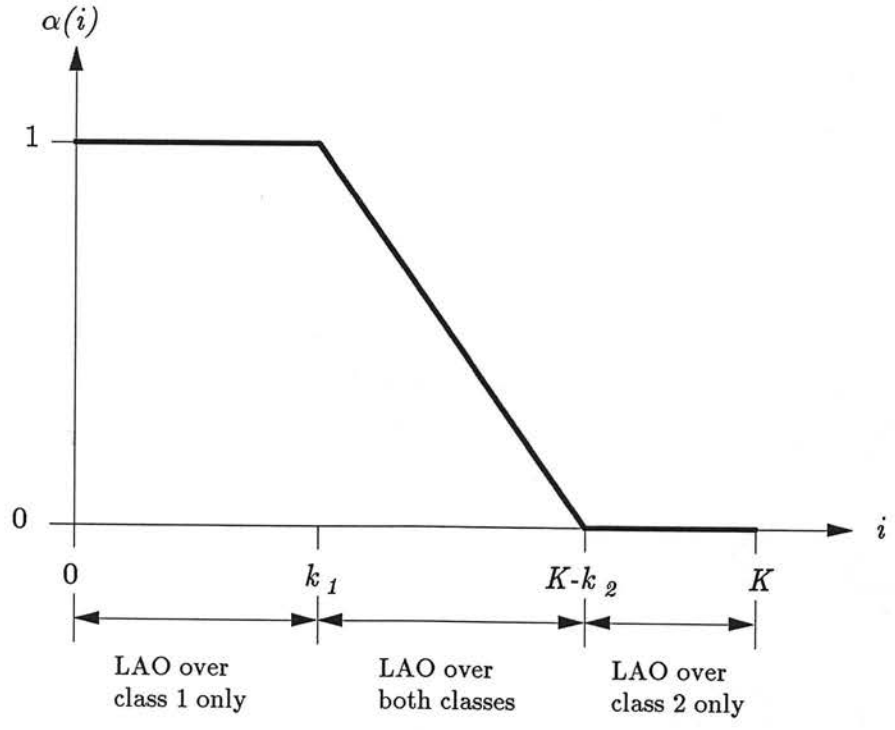


Figure 6.2: Typical plot of $\alpha(i)$ vs. i

fraction of the LAO covering class 1 for LAO position $i, i = 1, \ldots, K$. The random variable, i, is uniformly distributed between 1 and K and

$$P(i) = \frac{1}{K}, \; i = 1, \ldots, K. \tag{6.18}$$

The density of y can now be written as

$$p(y|\alpha(i)) = N(\alpha(i)\mu_{y_1} + (1 - \alpha(i))\mu_{y_2}, \alpha(i)^2\sigma_{y_1}^2 + (1 - \alpha(i))^2\sigma_{y_2}^2). \tag{6.19}$$

Without loss of generality, we can order the K feature values, $y^{(i)}, i = 1, \ldots, K$, such that the contribution of class 1 to $y^{(i)}$ is at least as large as it is for $y^{(i+1)}$. This implies that $\alpha(i)$ is a monotonically non-increasing function. For a typical image, $\alpha(i)$ may be shown in Fig. 6.2.

The probability density of y is then given by

$$\begin{aligned} p(y) &= \sum_{i=1}^{K} p(y|\alpha(i))P(i) \\ &= \frac{1}{K}\sum_{i=1}^{K} p(y|\alpha(i)) \\ &= \frac{1}{K}\left(\sum_{i=1}^{k_1} p(y|\alpha(i)) + \sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) + \sum_{i=K-k_2+1}^{K} p(y|\alpha(i))\right) \\ &= \frac{k_1}{K}p(y_1) + \frac{1}{K}\sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) + \frac{k_2}{K}p(y_2) \\ &= \beta_1 p(y_1) + \frac{1}{K}\sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) + \beta_2 p(y_2) \end{aligned} \tag{6.20}$$

where

$$\begin{aligned} p(y_1) &= p(y|\alpha(i)=1) \\ p(y_2) &= p(y|\alpha(i)=0). \end{aligned}$$

The resulting density has three components

1. A Gaussian part at $y = \mu_{y_1}$ due to class 1.
2. A Gaussian part at $y = \mu_{y_2}$ due to class 2.
3. A non-Gaussian part between μ_{y_1} and μ_{y_2} due to the mixture of classes 1 and 2.

Two special cases can be easily examined. If $\beta_1 + \beta_2 \to 1$, this implies that the overlap region is small in relation to the total area being examined. In this case Eq. (6.20) reduces to

$$p(y) = \beta_1 p(y_1) + \beta_2 p(y_2). \tag{6.21}$$

If $\beta_1 = \beta_2 = 0$, this implies that all positions of the window are affected by the boundary and Eq. (6.20) reduces to

$$p(y) = \frac{1}{K}\sum_{i=1}^{K} p(y|\alpha(i)). \tag{6.22}$$

Equation (6.20) can be simplified under certain circumstances. First, assume $p(y|\alpha(i))$ is Gaussian with the same second order moments for all values of i. Secondly, assume that the the means of the $p(y|\alpha(i))$ terms in the summation are linearly distributed between μ_{y_1} and μ_{y_2}. This implies that the function $\alpha(i)$ is linear between $i = k_1 + 1 \approx k_1$ and $i = K - k_2$, and is defined over that domain as

$$\begin{aligned} \alpha(i) &= \frac{K - (i + k_2)}{K - (k_1 + k_2)} \\ &= \frac{-i}{K(1 - (\beta_1 + \beta_2))} + \frac{1 - \beta_2}{1 - (\beta_1 + \beta_2)}. \end{aligned} \tag{6.23}$$

The linearity of the function $\alpha(i)$ is a reasonable assumption since in most case the action of the texture boundary on the mixture density is symmetric. This means that for every position of the window that encompasses some proportions of texture 1 and texture 2, there is another position in which the proportions are reversed.

If the number of terms in the summation is large ($K - (k_1 + k_2) \gg 1$), then the summation term can be approximated by an integration

$$\begin{aligned} \lim_{K\to\infty} \frac{1}{K}\sum_{i=k_1}^{K-k_2} p(y|\alpha(i)) &= \frac{-K(1 - (\beta_1 + \beta_2))}{K} \\ &\quad \times \lim_{K\to\infty} \sum_{i=k_1}^{K-k_2} p(y|\alpha(i)) \frac{-1}{K(1 - (\beta_1 + \beta_2))} \\ &= -(1 - (\beta_1 + \beta_2)) \int_1^0 p(y|\alpha) d\alpha \\ &= (1 - (\beta_1 + \beta_2)) \int_0^1 p(y|\alpha) d\alpha \end{aligned} \tag{6.24}$$

where $p(y|\alpha)$ is a Gaussian distribution with the mean a function of α

$$p(y|\alpha) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{1}{2}\left(\frac{y-\mu(\alpha)}{\sigma_y}\right)^2}. \tag{6.25}$$

The mean is described by the function

$$\mu(\alpha) = \alpha\mu_{y_1} + (1-\alpha)\mu_{y_2}. \tag{6.26}$$

Changing variables of integration from α to μ changes Eq. (6.24) to

$$\begin{aligned}\lim_{K\to\infty} \frac{1}{K} \sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) &\approx (1-(\beta_1+\beta_2)) \int_0^1 p(y|\alpha) d\alpha \\ &= \frac{(1-(\beta_1+\beta_2))}{\mu_{y_1}-\mu_{y_2}} \\ &\quad \times \int_{\mu_{y_2}}^{\mu_{y_1}} \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma_y}\right)^2} d\mu. \end{aligned} \tag{6.27}$$

Another change of variables from μ to z where

$$z = \frac{y-\mu}{\sqrt{2}\sigma_y} \tag{6.28}$$

results in

$$\begin{aligned}\lim_{K\to\infty} \frac{1}{K} \sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) &\approx \frac{(1-(\beta_1+\beta_2))}{\mu_{y_1}-\mu_{y_2}} \left(\frac{1}{\sqrt{2\pi}\sigma_y}\right) \left(-\sqrt{2}\sigma_y\right) \int_{\frac{y-\mu_{y_2}}{\sqrt{2}\sigma_y}}^{\frac{y-\mu_{y_1}}{\sqrt{2}\sigma_y}} e^{-z^2} dz \\ &= \frac{(1-(\beta_1+\beta_2))}{2(\mu_{y_2}-\mu_{y_1})} \frac{2}{\sqrt{\pi}} \int_{\frac{y-\mu_{y_2}}{\sqrt{2}\sigma_y}}^{\frac{y-\mu_{y_1}}{\sqrt{2}\sigma_y}} e^{-z^2} dz \\ &= \frac{(1-(\beta_1+\beta_2))}{2(\mu_{y_2}-\mu_{y_1})} \\ &\quad \times \left(erf\left(\frac{y-\mu_{y_1}}{\sqrt{2}\sigma_y}\right) - erf\left(\frac{y-\mu_{y_2}}{\sqrt{2}\sigma_y}\right)\right) \end{aligned} \tag{6.29}$$

where erf is the Gaussian error function defined as

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz. \tag{6.30}$$

The probability density of y for the simplified case just described is then given by

$$p(y) = \beta_1 p(y_1) + (1 - (\beta_1 + \beta_2)) \frac{erf\left(\frac{y-\mu_{y_1}}{\sqrt{2}\sigma_y}\right) - erf\left(\frac{y-\mu_{y_2}}{\sqrt{2}\sigma_y}\right)}{2(\mu_{y_2} - \mu_{y_1})} + \beta_2 p(y_2). \quad (6.31)$$

Figures 6.3 and 6.4 are contour plots of a sequence of mixture densities for the model just derived (Eq. (6.31)), and for the data present in the texture mosaic features. Figure 6.4 represents an ensemble average of the distributions of the feature data in a collection of classes that mix at a joint boundary and in a collection of all feature planes. The mixing parameter, $\beta_1 + \beta_2$, ranges from zero to one in both plots. The y variable is plotted from lower left to upper right and $p(y)$ is plotted vertically. Each slice though the figure from lower left to upper right represents a different value of $\beta_1 + \beta_2$. At the lower right side of the contour plot is the case of $\beta_1 + \beta_2 = 1$ which implies no mixing. At the upper left is the case of $\beta_1 + \beta_2 = 0$ which implies full mixing. For Fig. 6.4, the actual distributions of the two classes present in the mixture have been normalized and scaled along the y axis to move the means to locations that correspond to those in the plot of the model results.

The mean of the random variable y using the probability density function of Eq. (6.20) is given by

$$\begin{aligned} E(y) &= \int_{-\infty}^{\infty} y p(y) dy \\ &= \int_{-\infty}^{\infty} y \left(\beta_1 p(y_1) + \frac{1}{K} \sum_{i=k_1+1}^{K-k_2} p(y|\alpha(i)) + \beta_2 p(y_2) \right) dy \\ &= \beta_1 E(y_1) + \frac{1}{K} \sum_{i=k_1+1}^{K-k_2} E(y|\alpha(i)) + \beta_2 E(y_2) \\ &= \beta_1 \mu_{y_1} + \frac{1}{K} \sum_{i=k_1+1}^{K-k_2} E(y|\alpha(i)) + \beta_2 \mu_{y_2}. \end{aligned} \quad (6.32)$$

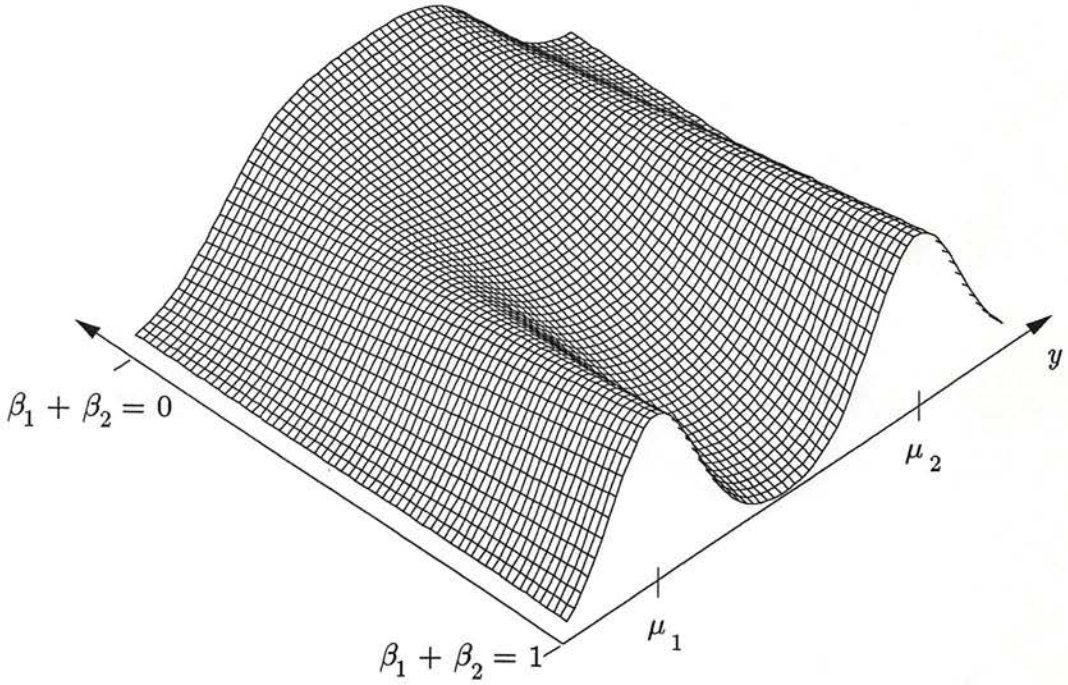


Figure 6.3: Plots of model results ($p(y)$ vs. y) for various degrees of mixing

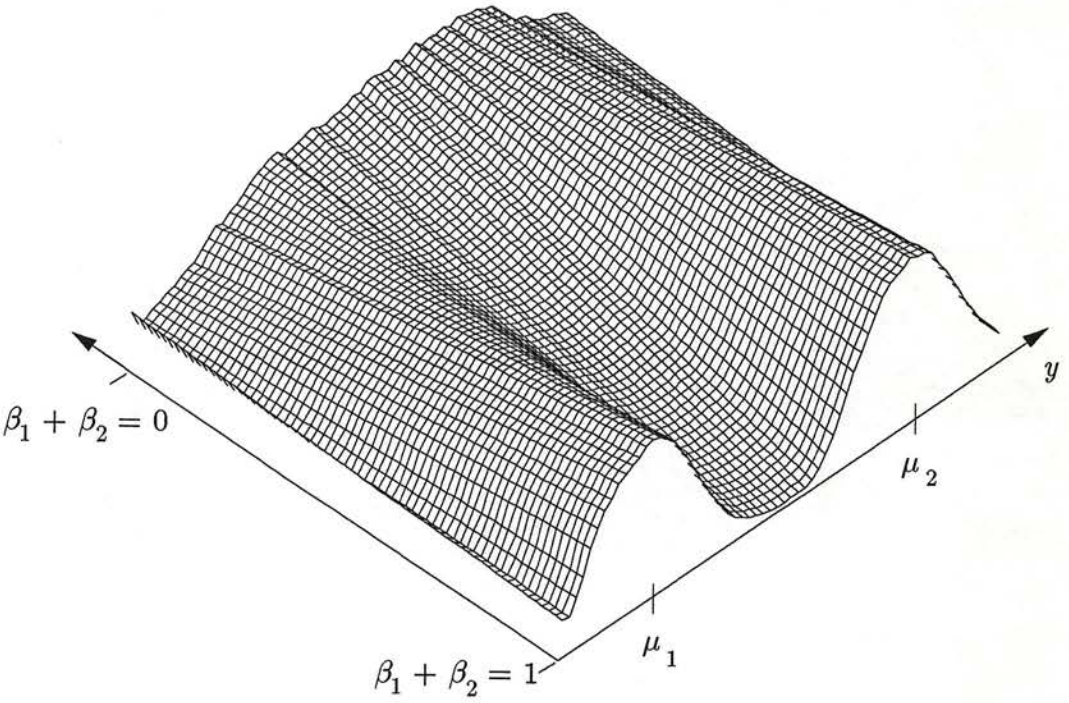


Figure 6.4: Plots of feature data ($p(y)$ vs. y) for various degrees of mixing

By Eq. (6.19), the expected value of the $p(y|\alpha(i))$ is

$$E(y|\alpha(i)) = \alpha(i)\mu_{y_1} + (1-\alpha(i))\mu_{y_2}. \tag{6.33}$$

The middle term in Eq. (6.32) can then be written as

$$\begin{aligned}
\frac{1}{K}\sum_{i=k_1+1}^{K-k_2} E(y|\alpha(i)) &= \frac{1}{K}\mu_{y_1}\sum_{i=k_1+1}^{K-k_2}\alpha(i) + \frac{1}{K}\mu_{y_2}\sum_{i=k_1+1}^{K-k_2}(1-\alpha(i)) \\
&= \frac{K-(k_1+k_2)}{K}\mu_{y_2} + \frac{\mu_{y_1}-\mu_{y_2}}{K}\sum_{i=k_1+1}^{K-k_2}\alpha(i) \\
&= (1-(\beta_1+\beta_2))\mu_{y_2} + \frac{\mu_{y_1}-\mu_{y_2}}{K}\sum_{i=k_1+1}^{K-k_2}\alpha(i). \qquad (6.34)
\end{aligned}$$

The summation term in the above expression is essentially averaging $\alpha(i)$ over the domain k_1+1 to $K-k_2$. If we again assume that the function $\alpha(i)$ is linear over this domain and ranges from one to zero, then the average value of α in this domain is simply $1/2$. The summation of the $\alpha(i)$ terms becomes

$$\begin{aligned}
\frac{1}{K}\sum_{i=k_1+1}^{K-k_2}\alpha(i) &= \frac{1}{K}\sum_{i=k_1+1}^{K-k_2}\frac{1}{2} \\
&= \frac{K-(k_1+k_2)}{2K} \\
&= \frac{1}{2}(1-(\beta_1+\beta_2)) \qquad (6.35)
\end{aligned}$$

and Eq. (6.34) reduces to

$$\begin{aligned}
\frac{1}{K}\sum_{i=k_1+1}^{K-k_2} E(y|\alpha(i)) &= (1-(\beta_1+\beta_2)\mu_{y_2} + (\mu_{y_1}-\mu_{y_2})\frac{1}{2}(1-(\beta_1+\beta_2)) \\
&= (1-(\beta_1+\beta_2))\frac{\mu_{y_1}+\mu_{y_2}}{2}. \qquad (6.36)
\end{aligned}$$

The expected value of y is thus given by

$$\begin{aligned}
E(y) &= \beta_1\mu_{y_1} + (1-(\beta_1+\beta_2))\frac{\mu_{y_1}+\mu_{y_2}}{2} + \beta_2\mu_{y_2} \\
&= \frac{\mu_{y_1}+\mu_{y_2}}{2} + \frac{(\beta_2-\beta_1)(\mu_{y_2}-\mu_{y_1})}{2}. \qquad (6.37)
\end{aligned}$$

The expected value of the output of the second stage LAO is thus a weighted sum of the means that contribute to the mixture. The weights are determined by the proportion of each class that is present in the LAO. This implies that if $\beta_1 = \beta_2$, meaning that there are equal amounts of both textures in the area being examined, then the mean of the mixture is midway between the individual class means. If $\beta_2 > \beta_1$, then the mean of the mixture is shifted towards the mean of class 2, and if $\beta_1 > \beta_2$, then the mean of the mixture is shifted towards the mean of class 1.

For a multi-dimensional feature space, the above derivation can be extended to show that the expected value of the feature vector is a weighted sum of the mean vectors of the classes within the LAO. In the feature space, the interpretation is that the expected value of the feature vector $\mathbf{y}$ lies on the line segment connecting the two class means.

The presence of data whose statistics are described by a mixture of probability densities can cause problems when used with a classification method that assumes that each data point is based on a single statistical model. We have seen that the expected location in feature space of the mixture points is somewhere between the classes contributing to the mixture. This is likely to cause misclassification errors if points in this region of the feature space are closer to the mean of a class not in the mixture than they are to the mean of any of the classes contributing to the mixture. In this situation, a classifier method based solely on measuring distance to means will incorrectly assign the sample point to nearest class. For example, Fig. 6.5 is the feature space representation of the class means from Fig. 6.1. In this example, the number of classes is four ($N = 4$) and the dimensionality of the features is two. The line segment

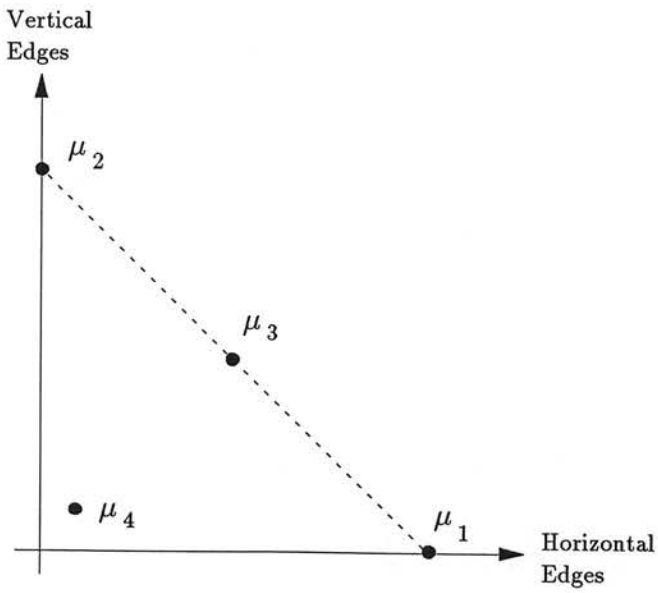


Figure 6.5: Location of class means in feature space

connecting the means of classes 1 and 2 passes very close to the mean of class 3. Feature points on the boundary of classes 1 and 2 are likely to be closer in feature space to the class 3 mean than to either of the means of classes 1 and 2. This is a problem that cannot be solved by a coordinate transformation such as rotation or warping of the feature space. A rotation or warping may change the direction or shape of the line connecting the means but can not solve the problem since the distance functions used to do the classifying are also altered by the transformation.

6.3 Ideal Situations Which Minimize the Mixture Density Problem

It is likely that the mixture density problem could be avoided if we were free to select the dimensionality of the feature space and the location of the class means. In many respects this is a similar problem to that encountered in the design of a communication system. In the selection of signals to be used in a digital communication system, one tries to pick a signal set that minimizes the probability of error. This is commonly done by using orthogonal or antipodal signals [53]. However, for a pattern classification problem the signals (class means) are predetermined.

The mixture density problem is minimized if the class means were located such that each class mean can be separated from the others in the feature space with a single hyperplane. The hyperplane has to satisfy the condition that all points lying in the plane are closer to at least one of the class means on one side of the plane than they are to the single class mean on the other side. Under these conditions we are guaranteed that no linear combination of the mean vectors of the other classes can result in the expected location of a point being on the same side of the separating plane as the single class. This assures us that the resulting expected location of the point is always closer to one of the classes that contributed to the linear combination.

For example, consider a case of four classes with three dimensional feature vectors. The conditions described above are satisfied if the means are made to lie equal distances apart on the vertices of a tetrahedron (Fig. 6.6). In this situation, the expected location of a feature point resulting from the mixture of

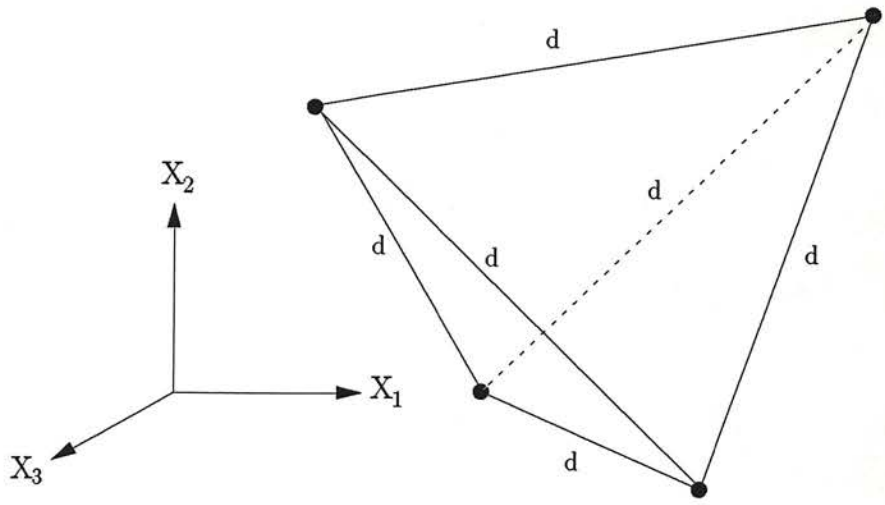


Figure 6.6: Optimum location for 4 classes in 3 dimensions

two or three classes is in one of the plane faces of the tetrahedron. A point in one of the faces is always within $\frac{d}{\sqrt{3}}$ of one of the class means that contributed to it, where d is the distance in the feature space between the vertices where the class means are located. Regardless of where the point is in the plane face, the distance to the class mean at the opposite vertex is at least $d\sqrt{\frac{2}{3}}$. We are thus assured that the point is always be closer to the mean of one of the classes that was part of the mixture density.

It is important to remember that while the expected location in the feature space of the mixture point may lie in one of these plane surfaces, the actual data is distributed around the expected locations due to the variance of the data. This means that some of the feature points may still be closer to an incorrect class mean, even with the class means in the ideal locations as described above.

6.4 Proposed Approximate Solution

The mixture density problem is inherent in any method of feature extraction that uses an LAO for calculating features since the LAO can overlap more than one class when analyzing the data. A proposed solution to this problem is to use a multiple-pass algorithm rather than a single-pass algorithm for doing the classification and segmentation. When using a single-pass classification method, all points in the image must be assigned to one of the classes based on the data available during that pass. There is no opportunity to leave a point unassigned due to a lack of confidence in the potential assignments. By doing the classification using a multiple-pass classifier, we have the option of delaying a decision on the classification of a point. Points for which we do not feel confident about making an assignment are temporarily placed into a "null-class" and are classified by the second stage of the classification process. The ability to avoid making a final decision based on weak evidence allows us to delay the classification until we have reason to be more confident about the decision. Delaying the classification makes it possible to gather more information about the point in question. By combining the information from several passes, the final decision can be made based on a hierarchy of data including spatial information such as the proximity of other points in the potential classes.

Use of a null-class during the preliminary passes does not increase the number of correct classifications, but it decreases the number of incorrect classifications. Eventually we must assign all points in the image to a class. The hierarchical structure is proposed in hope that the points which are not classified at first can be classified later with features that are more immune to the effects of

the mixture density. The choice of features that exhibit some immunity to the mixture density problem depends on the nature of the data being classified. For image texture, natural candidates are features generated using a smaller window which are less likely to overlap more than one texture. It is important to keep in mind that while we have been referring to this procedure as a classification process, it is also a segmentation process. Spatial information is being used indirectly at this stage since the features are dependent on location in the image.

At the stage in the classification process in which the null-class region is used, we are faced with what is essentially an $N + 1$ class problem. Either the point should be classified to one of the classes in question or it should be classified to the null-class.

Ideally, a Bayes decision surface that minimizes the cost of making a misclassification error can be used if the statistics of the mixture of two classes can be determined. Not using the Bayes decision surface has several advantages. As the derivation of the mixture density in Section 6.2 showed, even with several simplifying assumptions the statistics of the mixture are not Gaussian, and the decision surface can not, in general, be solved for in a closed form. Determining the position of a point relative to the decision surface has to be approximated and results in a higher computational cost. In addition, a Bayes classifier requires that we determine the costs of making classification errors. The costs of a misclassification error are not all equal. A point that is assigned to an incorrect texture class does more damage than one that is assigned to the null-class since there is a chance that the point can be classified correctly during the second phase of the classification. We must therefore decide on relative costs to apply to the two types of errors if we use a Bayes decision surface.

For these reasons, a trade-off must be made between the degree of optimality of the null-class region boundary and the speed of computation. The closer to an optimum boundary that is used, the greater is the computation cost. The following chapter describes some non-optimum null-class regions that have been tested.

Chapter 7

Null-Class Methods

Three different techniques have been examined for implementing the null-class concept described in the previous chapter for doing hierarchical classification and segmentation. While each technique has advantages and disadvantages in terms of classification and computational performance, all of the methods are basically similar in that they implement the null-class concept in the following manner:

1. Prior to classifying the data, the *a priori* statistics of the data is used to determine the size and location in the feature space of the null-class regions for each class, if any. If they exist, the null-class regions are defined differently for each class according to the data.
2. Using the classification algorithm described in Chapter 5, the image data points are temporarily assigned to one of the classes.
3. The location of the point in the feature space is compared to the location of the null-class region for that class. If the point is inside the null-class

region, it is assigned to the null-class. If it is not in the null-class, it is left in the class determined by step 2.

The primary differences between the following methods are in the shape and size of the null-class region that is used for each class. These methods by no means represent an exhaustive list of possible methods, but do illustrate a relatively wide range of the type of null-class regions that can be implemented. The methods described are all designed to guard against a mixtures of pairs of classes. The concepts can be extended to higher dimensionality if we wish to have a null-class that tests for mixtures of three or more classes.

7.1 Hypersphere Method

One of the simplest techniques for implementing hierarchical decisions is to surround each class mean with a spherical neighborhood enclosing the area in which we can be reasonably confident of a proper classification. Points that fall within these hyperspheres are classified immediately to that class. Points which are outside of all the hyperspheres are assigned to the null-class. The radius of each hypersphere is class dependent and is selected so that most of the points that may be from other classes or from mixtures fall outside the sphere. Obviously, the radius of the hypersphere should be such that the sphere is entirely contained within the normal Bayesian classification region as determined by the discriminant functions. In the absence of any other restrictions, this constrains the radius of the spheres to be the minimum of the distances from the mean to the normal Bayesian boundaries. To try to protect against the effects of the mixture density, an additional condition is placed on the radius of

the spheres. The radius must be small enough that no line segment connecting two other means passes through the sphere.

These ideas are illustrated in Fig. 7.1 which is an example of a four-class, two dimensional feature space. The class means are denoted by the points μ_1, μ_2, etc. The solid lines indicate some of the pairwise Bayesian class boundaries which lie between each pair of classes. The dashed lines indicate some of the line segments between means. For simplicity, the Bayesian class boundaries are shown as straight lines perpendicular to the lines between pairs of class means. In practice this is only be true if the covariance matrices were equal for all the classes.

By examining the relative locations of the means and the line segments we can see that μ_2 and μ_4 are likely to be unaffected by any mixture of the other classes. For instance, none of the points along the lines between the pairs μ_1–μ_2, μ_2–μ_3, and μ_1–μ_3, are closer to μ_4 as they are to one of the class means on the endpoints of the line segments. The same applies for μ_2 and the line segments between the other means. However, some of the points along the line segment μ_1–μ_2 are closer to μ_3 than they are to either μ_1 or μ_2. Therefore the size of the region around μ_3 is determined, in part, by the distance from μ_3 to this line segment. The same may also be true for the region around μ_1 and must be checked to determine the radius of the classification region.

The circles show the extent of the spherical regions after applying the two criteria described above. Note that the maximum radius of each circle is determined by the closest of the class boundaries and the line segments between the means. By letting the hypersphere radius be such that both conditions are satisfied, we are assured that (1) the hypersphere does not contain any part of

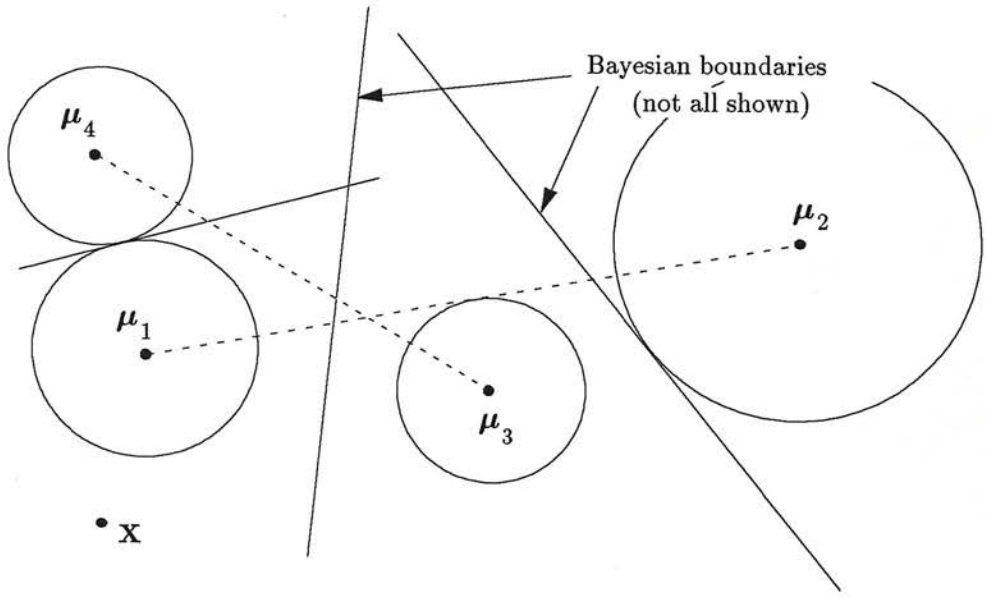


Figure 7.1: Hyperspheres for implementing null-class

the feature space that was not originally a part of the region using the normal discriminant functions, and (2) the expected position of any point resulting from the mixture of two classes falls outside the sphere, placing it in the null-class. It is important to remember that the line segments only represent the location of the expected values of the points generated by the mixture of the class statistics. The individual distributions of the points around each of the means assures us that the actual features points cover a region around both of the means and also around the line segment. The distribution of the points is determined by the probability density of the mixture.

As previously described in Section 5.1 (Eq. (5.13)), the distance from a mean to a point in the feature space can be measured by using the the Bayes classifier discriminant function as an approximate distance metric. Under the assumption of equal *a priori* probabilities, the Bayes discriminant function is equal to the

Mahalanobis distance plus a class-dependent offset. The Bayes squared distance is thus given by

$$D_B^2(\boldsymbol{\mu}_i, \mathbf{x}) = g_i(\mathbf{x}) = D_M^2(\boldsymbol{\mu}_i, \mathbf{x}) + K_i \tag{7.1}$$

where

$$D_M^2(\boldsymbol{\mu}_i, \mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_i)^t \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \tag{7.2}$$

is the Mahalanobis distance and

$$K_i = log|\Sigma_i|. \tag{7.3}$$

The covariance matrix, Σ_i, is that associated with mean $\boldsymbol{\mu}_i$. The presence of the constant, K_i, causes $D_B(\boldsymbol{\mu}_i, \mathbf{x})$ to violate the requirements of a true distance function since $D_B(\boldsymbol{\mu}_i, \boldsymbol{\mu}_i)$ is not zero. By using the Bayes discriminant function for the distance metric we are assured that all points on a boundary between two classes are the same "distance" from both classes.

When we use this technique for measuring distance, the shape of each resulting neighborhood is spherical only in the sense that the radius is constant in all directions from a mean when using the corresponding Bayes distance. The presence of the non-identity covariance matrix in the distance function causes the actual neighborhoods to be ellipsoids rather than spheres when viewed in Euclidean space. This also has the side effect that the Bayesian boundaries between the classes are plane surfaces in the Mahalanobis space but curved surfaces when viewed in Euclidean space. Again, the covariance matrices for each pair of classes determine the shape of the boundary between the classes. This creates an unusual situation in which each mean is surrounded by a "hypersphere" that is not spherical in the Euclidean sense. However, since this technique is simply

an extension of the process that is used to do the classification, it appears to be valid.

The boundary between two classes is determined by a pair of discriminant functions. The distance from one of the class means to the point on the boundary that lies on the line segment between the two means is found by first finding the equation of the line segment. Assuming we have two class means, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ as in Fig. 7.1, a point on the line between them is given by

$$\mathbf{x} = \alpha\,\boldsymbol{\mu}_1 + (1-\alpha)\boldsymbol{\mu}_2 \tag{7.4}$$

where $0 < \alpha < 1$. At the class boundary, the distance from $\boldsymbol{\mu}_1$ to the point is the same as the distance from the $\boldsymbol{\mu}_2$ to the point,

$$D_B(\boldsymbol{\mu}_1, \mathbf{x}) = D_B(\boldsymbol{\mu}_2, \mathbf{x}). \tag{7.5}$$

Both of these distances are functions of the covariance matrices of the respective classes. The Bayes distance for class 1 is used to measure the distance from $\boldsymbol{\mu}_1$ to point $\mathbf{x}$ on the boundary, and the Bayes distance for class 2 is used to measure the distance from $\boldsymbol{\mu}_2$ to $\mathbf{x}$. The distance from $\boldsymbol{\mu}_1$ to $\mathbf{x}$ is not necessarily equal to the distance from $Mean[2]$ to $\mathbf{x}$ if the Bayes distance function for class 1 is used for both distances. The same applies to using the Bayes distance function for class 2 for both distances.

Setting the discriminant functions (Eq. (7.1)) equal and substituting the parametric expression for $\mathbf{x}$ (Eq. (7.4)) into it yields the quadratic equation

$$\alpha^2(\Delta\boldsymbol{\mu}_1 - \Delta\boldsymbol{\mu}_2) - \alpha 2\Delta\boldsymbol{\mu}_1 + (\Delta\boldsymbol{\mu}_1 + log|\Sigma_1| - log|\Sigma_2|) = 0 \tag{7.6}$$

where

$$\Delta\boldsymbol{\mu}_1 \;=\; (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t \Sigma_1^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{7.7}$$

$$\Delta\mu_2 \;=\; (\mu_1 - \mu_2)^t \Sigma_2^{-1} (\mu_1 - \mu_2). \tag{7.8}$$

If the covariances of the two classes are equal ($\Sigma_1 = \Sigma_2$) then $\Delta\mu_1$ and $\Delta\mu_2$ are equal. The solution for α is then $\alpha = \frac{1}{2}$, implying the boundary is midway between the means, as expected. In general the solution to the quadratic equation is given by

$$\alpha = \frac{\Delta\mu_1 \pm \sqrt{\Delta\mu_1\Delta\mu_2 - (\Delta\mu_1 - \Delta\mu_2)(log|\Sigma_1| - log|\Sigma_2|)}}{\Delta\mu_1 - \Delta\mu_2} \tag{7.9}$$

This equation has two solutions, one of which is between zero and one and can be used to determine the boundary point lying between μ_1 and μ_2 on the line connecting them. Once the location of this point has been determined, the distance from the class mean to the point is found using the corresponding distance function (Eq. (7.1)).

The distance from a class mean to a line segment between two other means is found in much the same manner. Again referring to Fig. 7.1, we wish to find the value of α that determines the point on the line between μ_1 and μ_2 that is closest to the mean of a third class, μ_3. A point on the line between μ_1 and μ_2 is given by Eq. (7.4). As before, the the distance from μ_3 to the point $\mathbf{x}$ on the line between μ_1 and μ_2 is given by the Bayes classifier "distance"

$$D_B(\mu_3, \mathbf{x}) = (\mathbf{x} - \mu_3)^t \Sigma_3^{-1} (\mathbf{x} - \mu_3) + log|\Sigma_3| \tag{7.10}$$

where the covariance matrix, Σ_3, is that associated with mean μ_3. Substituting the expression for $\mathbf{x}$ into the distance function results in a quadratic equation in α

$$\begin{aligned} D_B(\mu_3, \mathbf{x}) = \quad & \alpha^2 (\mu_1 - \mu_2)^t \Sigma_3^{-1} (\mu_1 - \mu_2) \\ + \quad & 2\alpha (\mu_2 - \mu_3)^t \Sigma_3^{-1} (\mu_1 - \mu_2) \end{aligned}$$

$$+ \quad (\boldsymbol{\mu}_2 - \boldsymbol{\mu})^t \Sigma_3^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_3) + log|\Sigma_3|. \tag{7.11}$$

We wish to find the value of α that minimizes this distance. The derivative of the distance with respect to α is given by

$$\frac{dD_B}{d\alpha} = 2\alpha(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t \Sigma_3^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + 2(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_3)^t \Sigma_3^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \tag{7.12}$$

Setting the derivative to zero and solving for α gives the result

$$\alpha = \frac{(\boldsymbol{\mu}_3 - \boldsymbol{\mu}_2)^t \Sigma_3^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)}{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t \Sigma_3^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)}. \tag{7.13}$$

In many cases, the mean of the third class is closer to one of the means on the ends of the line segment than to any of the points between the means. This is the case of $\boldsymbol{\mu}_4$ in Fig. 7.1 which is closer to $\boldsymbol{\mu}_1$ than it is to the points between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. For these situations, the mixture density problem described above is not serious and we ignore it. This situation can be detected by noting whether the value of α lies outside the range of zero to one. In Eq. (7.13), if α is between zero and one, then there is a point on the line segment closer to $\boldsymbol{\mu}_3$ than either $\boldsymbol{\mu}_1$ or $\boldsymbol{\mu}_2$ and the coordinates of the nearest point are given by Eq. (7.4) using the value of α just determined. The distance from $\boldsymbol{\mu}_3$ to the point is found using Eq. (7.11). This value is used in conjunction with the distance to the nearest class boundary to determine the radius of the hypersphere around $\boldsymbol{\mu}_3$.

7.1.1 Experimental Results

Table 7.1 shows the distances (in units of Bayes distances) from each mean to the closest boundary and the class on the other side of that boundary for the texture mosaic image. From the table we can see that classes 3 and 5 are apparently very similar since their means are quite close together in the feature space. As

Class	Distance	Class on other side of boundary
1	17.1	3
2	9.6	8
3	3.8	5
4	27.1	6
5	3.8	3
6	22.9	5
7	23.6	1
8	9.6	2

Table 7.1: Distance to nearest class boundary (31×31 features)

a result, the hyperspheres around the means of these classes are quite small and this causes many of the points from those classes to remain unclassified after the first classification pass.

Table 7.2 shows the results of the test for the proximity of a line segment between two other means. The table lists the distances from each mean to the nearest line segment which satisfies the mixture density problem criterion described above.

The radius of the local area hypersphere is the minimum of the nearest boundary distance and the nearest line segment distance. By comparing the results shown in Tables 7.2 and 7.1 we can see that only class 6 has a line segment closer to its mean than a class boundary. For all other classes, the radius of the hypersphere was determined by the distance to the nearest class boundary.

Class	Distance	Classes on line segment
1	31.5	3-7
2	11.1	3-8
3	4.5	1-5
4	52.6	5-6
5	7.5	3-4
6	15.4	1-4
7	109.9	1-5
8	340.1	1-4

Table 7.2: Distance to nearest line segment connecting two classes (31×31 features)

Figure 7.2 shows the result of classifying the texture mosaic using the Bayes classifier on the 31×31 features with the null-class implemented using the hyperspheres around each class mean. The areas of the image that are black are the areas that were not classified due to the feature point not being inside any of the hyperspheres. The areas of the image along the boundaries between textures have been left unclassified in most cases. Note that the regions corresponding to classes 3 and 5 have a large portion of the points not classified as a result of the small sizes of those hyperspheres. Table 7.3 lists the results of this classification. As expected, the portions of the image consisting of small texture regions have a greater number of unclassified points. The most promising result is that the number of incorrectly classified points have been kept to acceptable levels even in the small regions. While it is preferable to classify these points to the correct class, leaving them unclassified is better than having them classified incorrectly. The basic idea is to minimize the damage done by the large-feature classification operation since the points can always be classified later.

Figure 7.2: Classification of 31×31 features using hypersphere neighborhoods

Region	Correct	Incorrect	Not classified
Overall	38.7	1.9	59.4
128x128	54.1	1.0	45.0
64x64	38.6	1.3	60.2
32x32	12.7	2.1	85.2
16x16	3.7	6.8	89.4

Table 7.3: Hypersphere classification results (31×31 features)

7.2 Hyperplane Method

The major drawback of implementing the null-class region using hyperspheres is that they are overly restrictive. An example of this is shown in Fig. 7.1. Consider the point **x** which is significantly closer to the mean of class 1 than it is to the mean of any other class. There appears to be little doubt as to which class the point should be assigned. However by implementing the null-class test as described above, the point is left unassigned. The spherical shape of the region forces points to be left unclassified that are on the opposite side of the mean from the mixture density line and are almost certain not to be the result of mixing classes. To avoid this problem, it is necessary to use a region that is not omnidirectional but instead conforms better to the surrounding class mixtures. One way to do this is by to create an arbitrarily shaped classification region for each class using hyperplanes in much the same way that the Gaussian decision surfaces are formed. By using plane surfaces we are free to have either a closed (finite volume) or open (infinite volume) region depending on the distribution of the class means. The hypersphere regions were, by nature, closed regions. One significant difference between the two methods is that is that for some classes no hyperplanes are needed if all the expected locations of mixture points are outside of the normal class boundary.

Finding an acceptable method for determining the placement of the hyperplanes is important for making this technique work. When doing the classification, the hyperplanes are used in exactly the same way as the normal Bayesian boundaries in that a point must be the correct side of all the surfaces in order to be classified to a particular class. The classification problem has now grown to

where we wish to be able to separate each class from the other classes and from all the pairs of mixtures that may be present. Thus the mixture density points may be considered as members of a "pseudo-class" which must be separated from the true class. As with the normal decision surfaces, the placement of the hyperplane in the feature space is such that the mean of the mixture density pseudo-class and the mean of the true class are on the opposite sides of the plane. A separate hyperplane must be used for each mixture density class that may affect a certain true class.

Whether or not a class may be affected by a mixture density can be determined prior to classification by examining test points from the various mixture classes. For each class, we must test mixtures of all possible pairs of the other class means to determine whether the mixture of any two classes may be mistaken for the class in question. This is done by classifying a sample of points along the line segment between the two means. Depending on the proportion of each class in the mixture, the expected location of the mixture point is somewhere along this line segment. If the number of points on the line segment that are classified to the class in question is below some threshold, then we can assume that that particular mixture is not a significant problem and a hyperplane to separate that particular mixture from the true class is not needed. The number of separating hyperplanes for each class is very data-dependent. A class that lies in the midst of several other classes in feature space may need several, while an outlying class away from all others may not need any. In many cases some of the hyperplanes may be redundant in that one or more planes are not really needed because of the presence of another. Once we know which hyperplanes

are needed to separate a true class from a mixture, the location and orientation of the planes can be determined.

This idea is illustrated in Fig. 7.3. In a three-class, two dimensional feature space shown, we wish to separate class 3 from the mixture of classes 1 and 2. While it may be possible to determine the optimum position of the separating hyperplane relative to the $\boldsymbol{\mu}_1$–$\boldsymbol{\mu}_2$ line segment based on the density of the mixture, for now we simply assume that the plane is located a fraction ϵ of the distance from the $\boldsymbol{\mu}_1$–$\boldsymbol{\mu}_2$ line segment to the mean $\boldsymbol{\mu}_3$ as shown in Fig. 7.3. A value of zero for ϵ implies that the line segment lies in the separating hyperplane. In the absence of any information about the shape of the mixture density, the natural orientation for the hyperplane is to make it normal to the line between $\boldsymbol{\mu}_3$ and the nearest point on the $\boldsymbol{\mu}_1$–$\boldsymbol{\mu}_2$ mixture line as shown. This assures us that the mixture line and the plane do not intersect and that the entire length of the mixture line is on the opposite side of the plane from $\boldsymbol{\mu}_3$.

Determining whether or not a point should be classified or left in the null-class is done in two steps. First, the normal Bayesian classification is done to determine the class to assign the point to if the null-class were not being used. If any hyperplanes have been define for this class then the position of the point is examined to see if it is within the restricted classification region for the suggested class. If it is, it is assigned to that class. If it is not, it is assigned to the null-class. The first part of this method is done in exactly the same manner as a normal, unrestricted classification. In a normal classification, the means are determined from the data and the quadratic surfaces that make up the pairwise class boundaries are the locus of points equidistant from the pair of means using the Bayesian distance described previously in Eq. (7.1).

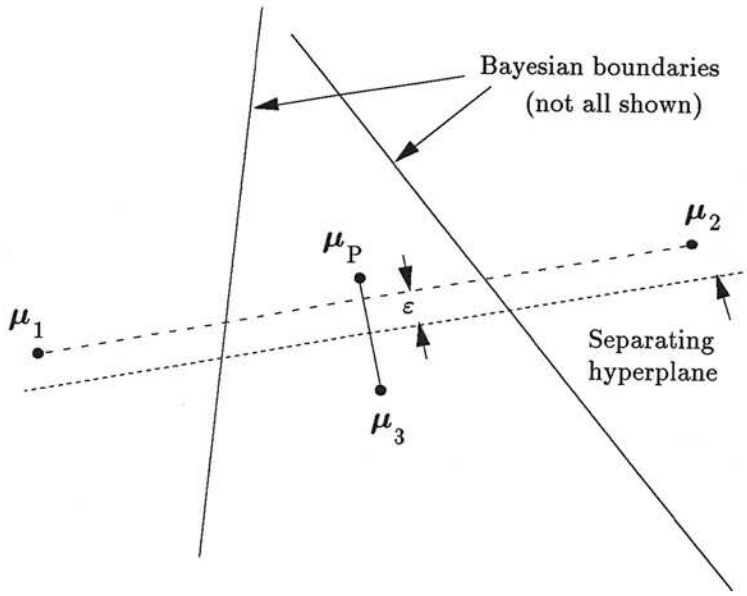


Figure 7.3: Sample feature space for hyperplanes

The second part of the null-class test is done in much the same manner. We know the location of the mean in question and the desired location and orientation of each of the hyperplanes. We can therefore work backwards to determine the location of a mean for the pseudo-class, which when tested against the mean in question, generates the desired boundary plane. This "pseudo-mean" is used as a mechanism to form a discriminant function for separating the mean from the mixture points. The "pseudo-mean" for separating class 3 from the mixture of classes 1 and 2 is shown in Fig. 7.3 as μ_P.

One pseudo-mean is determined for each hyperplane we wish to generate. Once the pseudo-means are determined for each class, the point can be classified exactly as before except it is being classified to either the true class or one of the associated pseudo-classes. If the classification result is one of the pseudo-classes this implies that the point is outside the restricted class region and it is placed

in the null-class. If the result is the true class, this implies that the point is inside the region and it is classified to that class.

The proper location of the pseudo-mean may be found using the results obtained in the previous section. Eqs. (7.13) and (7.4) gave us location of a point on a line segment nearest to another mean. As with the hyperspheres, the distance function used is the Bayes distance (Eq. (7.1)). Since the constant term in the Bayes distance is class dependent, it can be dropped in this situation since we are only considering distances from a single true class and the constant is in all terms. With the constant term removed, the Bayes distance becomes the Mahalanobis distance (Eq. (7.2)). Since the distance function is non-Euclidean, this implies that the "planes" are actually curved surfaces if examined in a Euclidean space. This is the same warping that caused the spherical neighborhoods described in the preceding section to actually be ellipsoids in a Euclidean space. As with the spheres, the warping is due to the presence of the non-identity covariance matrix in the function used to compute the distance.

Using Eqs. (7.13) and (7.4) we can find the location of the point $\mathbf{x}$ on the $\boldsymbol{\mu}_1$–$\boldsymbol{\mu}_2$ line segment that is closest to $\boldsymbol{\mu}_3$. We wish to find the location of the pseudo-mean, $\boldsymbol{\mu}_P$, that results in the hyperplane being located in the desired place. Since the same covariance matrix is used for both the true class and the pseudo-class, the boundary plane between them is located midway between the means and oriented normal to the $\boldsymbol{\mu}_3$–$\boldsymbol{\mu}_P$ line segment. If we let d_p represent the distance from $\boldsymbol{\mu}_3$ to the plane, then

$$d_p = \frac{1}{2} D_M(\boldsymbol{\mu}_3, \boldsymbol{\mu}_P). \tag{7.14}$$

Assuming the distance from $\boldsymbol{\mu}_3$ to $\mathbf{x}$ is denoted by d_x, then

$$d_p = (1-\epsilon)d_x = (1-\epsilon)D_M(\boldsymbol{\mu}_3, \mathbf{x}). \tag{7.15}$$

Combining the two expressions for d_p and expanding the terms for the Mahalanobis distance yields

$$(\frac{\boldsymbol{\mu}_3 - \boldsymbol{\mu}_P}{2})^T \Sigma_3^{-1} (\frac{\boldsymbol{\mu}_3 - \boldsymbol{\mu}_P}{2}) = ((1-\epsilon)(\boldsymbol{\mu}_3 - \mathbf{x}))^T \Sigma_3^{-1} ((1-\epsilon)(\boldsymbol{\mu}_3 - \mathbf{x})). \tag{7.16}$$

Solving for $\boldsymbol{\mu}_P$ results in

$$\boldsymbol{\mu}_P = 2(1-\epsilon)(\mathbf{x} - \boldsymbol{\mu}_3) + \boldsymbol{\mu}_3. \tag{7.17}$$

This expression can also be derived by just scaling the appropriate vectors. The vector from $\boldsymbol{\mu}_3$ to the point $\mathbf{x}$ is given by $\mathbf{x} - \boldsymbol{\mu}_3$. The vector from $\boldsymbol{\mu}_3$ to the plane is this vector scaled by $(1-\epsilon)$ or $(1-\epsilon)(\mathbf{x} - \boldsymbol{\mu}_3)$. If we now scale this vector by a factor of two the result is the vector from $\boldsymbol{\mu}_3$ to the desired location of $\boldsymbol{\mu}_P$. Finally, adding it to $\boldsymbol{\mu}_3$ gives the location of $\boldsymbol{\mu}_P$ or $2(1-\epsilon)(\mathbf{x} - \boldsymbol{\mu}_3) + \boldsymbol{\mu}_3$.

7.2.1 Experimental Results

Table 7.4 shows the results of classifying points along the various line segments between pairs of class means for the texture mosaic. For each class, the table lists the line segments that had points classified to that class and the percentage of points for which this happened. The percentage value indicates how much of the line segment passes through that class region.

These results confirm what we have been seeing in the classification results. Texture 6 is most affected by the mixture problem since it is nearest to the mean of all the feature points. Conversely, texture 8 is isolated from the others

Class	Line segment	Percent of Points
1	6–7	5
2	1–8	27
	3–8	37
	4–8	33
	5–8	39
	6–8	29
	7–8	51
3	1–4	11
	1–5	34
	5–6	30
	6–7	4
4	none	
5	3–4	27
	4–7	44
	6–7	17
6	1–2	24
	1–4	26
	1–8	24
	2–3	30
	2–4	10
	2–5	27
	3–8	27
	4–8	4
	5–8	22
7	none	
8	none	

Table 7.4: Results of classifying test points along line segments between pairs of class means

and is not affected by the mixture problem as we have modeled it. Texture 2 lies somewhere between texture 8 and the others. As a result of this, all the mixtures that are a potential problem for texture 2 are combinations which include texture 8. This is why the border areas of the texture 8 regions are consistently classified to texture 2.

Figure 7.4 is the result of classifying the texture mosaic using the Bayes classifier on the 31×31 features with the null-class implemented with the hyperplane technique described above. In this example, the value of ϵ was 0.4, thus the hyperplane was situated 40% of the distance from the mixture line to the mean. As with the hypersphere results (Fig. 7.2), the areas of the image that are black are the pixels that have been left unclassified. Visually comparing Figs. 7.2 and 7.4 we can see that the hyperplane performed better in the larger 128×128 regions. Substantially more of regions 3 and 5 have been correctly classified with this technique whereas much of these regions were left unclassified with the hypersphere technique.

Table 7.5 lists the results of this test. The number of incorrectly classified points for each region size is greater than with the hypersphere method since this is by design a less restrictive method. We are faced with a trade-off between a higher correct classification rate and a lower incorrect classification rate. If we wish to have a greater number of points classified during the first pass, then we run the risk of having them classified incorrectly. If we set the requirements for letting a point be classified too stringent, then too many of the points are not classified. Since the results of the large operator classification are used when performing the second classification with the small operator, it is important that at least some of the points be classified during the first pass.

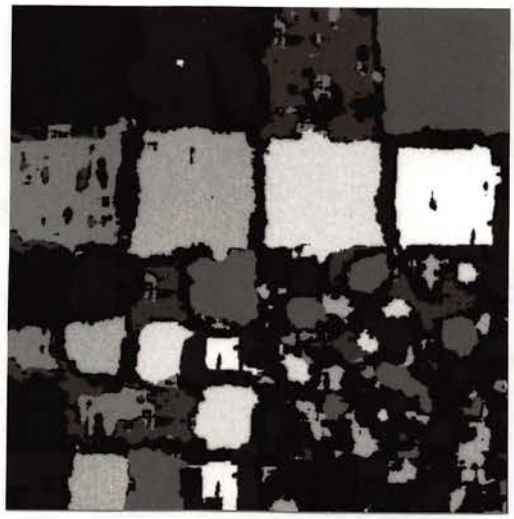


Figure 7.4: Classification of 31×31 features using hyperplane neighborhoods

Region	Correct	Incorrect	Not classified
Overall	60.8	13.1	26.2
128x128	76.4	7.7	15.9
64x64	63.8	14.2	22.0
32x32	36.3	19.5	44.3
16x16	16.6	25.9	57.5

Table 7.5: Hyperplane classification results (31×31 features), $\epsilon = 0.4$

7.3 Hypercylinder Method

Both of the previous methods suffer from the same problem in that an excessive amount of the feature space is allocated to the null-class. In the decision process using the hypersphere method the feature space was divided into a finite volume region around the mean and an infinite volume region for the null-class. With the hyperplane method, each plane divided the feature space into two half-spaces, one for classification and one for the null-class. Depending on the orientation of planes, the null class is either a finite or infinite volume in the feature space. The infinite volume of the null-class can often cause too many points to be left unclassified. In a sense these methods are too conservative in deciding whether or not to commit to a class assignment for a particular point.

The third method to be described is a natural extension of the first two methods. The hypercylinder method divides the feature space into a finite volume region for the null-class and an infinite volume region for classification. In this method, the null-class region is a cylindrical volume that has the line between two class means as its axis. Figure 7.5 shows a three class feature space in which the mixture of classes 1 and 2 must be separated from class 3. The null-class has been implemented using a cylinder aligned along the line between classes 1 and 2. This shape of this null-class region using the cylinder can be compared with that obtained using the hyperplane method shown in Fig. 7.3. From an intuitive sense, using the cylinder is preferable since it is more closely matched to the expected shape of the mixture density.

As with the previous methods, the computational cost can be reduced by doing as much work as possible before actually classifying the data. The class

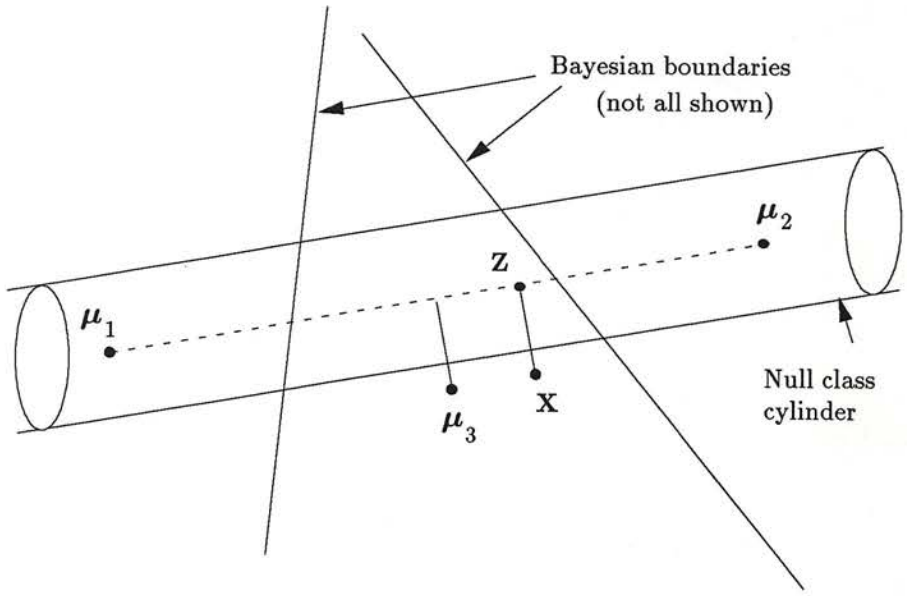


Figure 7.5: Sample feature space for hypercylinders

means and class covariances (known *a priori*) can be analyzed to determine which of the possible mixture pairs conflict with a particular class. This is determined in the same manner as for the hyperplane technique. Points along the line segment between pairs of class means are classified to determine which classes are affected by the mixing of the statistics. As with the hyperplanes, if a significant number of points along the line are classified to a third class, then this pair is checked later for any point initially assigned to that class. If none of the points along the mixture line are assigned to that class, then there is no need to check this mixture pair. This technique for determining whether or not to test a mixture pair is easy to implement but does have one drawback. Unless the mixture line actually passes through the classification region for a class, no test is made. It is conceivable that a mixture pair may not be tested if the line passes very close to the region but does not actually intersect the region.

The alternative is to not bother with a criterion for whether or not to check a mixture pair but to simply check every mixture pair for each point classified.

The only parameter that must be determined when using cylindrical regions is the radius of each cylinder. Since the distances are measured in the non-Euclidean feature space, the radius is calculated in a manner that reflects the the shape of the mixture density. This is achieved by measuring the distances with the Mahalanobis distance as in the previous methods. The cylinder radius is specified in units of Mahalanobis distance. This is similar to specifying the radius in units of standard deviation.

For the classified points, the test for membership in the null-class is somewhat simpler with this method than with the hyperplane method. For a point $\mathbf{x}$ that is classified to a particular class, a test is made to determine if the point is inside the null-class cylinder for any of the preselected mixture pairs for that class. This is done by finding the distance from the mixture line to the point and comparing the distance to the cylinder radius for that mixture pair. Eqs. (7.13) and (7.4) are used to find the location of the point $\mathbf{z}$ on the line segment between the two means that is closest to $\mathbf{x}$. The distance between the points $\mathbf{z}$ and $\mathbf{x}$ is found using the Bayes distance (Eq. (7.1)). This is repeated for all mixture pairs associated with the class. If the point is outside of all the cylinders then it is assigned to the original class. If the point is in the interior of any of the cylinders, then it is assigned to the null-class.

7.3.1 Experimental Results

The criterion used for selecting the mixtures which must be checked for this method is the same as that for the hyperplane method. The results that were

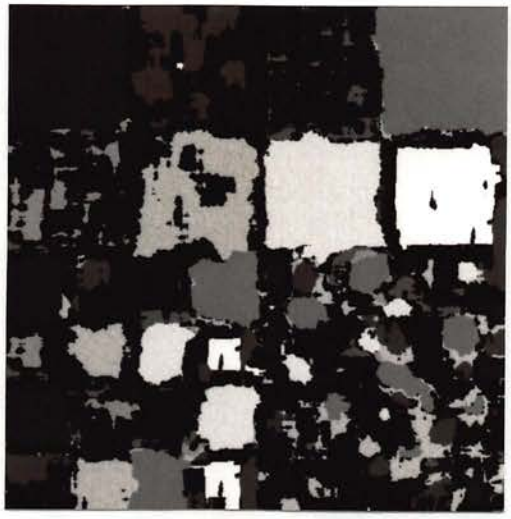


Figure 7.6: Classification of 31×31 features using hypercylinder neighborhoods

shown previously in Table 7.4 for the hyperplane method for the classification of points along the line segments between class means are also used to select mixture pairs to check for the hypercylinder method.

Figure 7.6 is the result of classifying the texture mosaic using a Bayes classifier on the 31×31 features with the cylindrical null-class. The cylinder radius was set to a Mahalanobis distance of 3. The black areas of the image are the pixels that were left unclassified.

Table 7.6 lists the results of the this test. The percent of incorrectly classified points has decreased in the larger regions as compared to the results with the hyperplanes method. The number of correctly classified points has also decreased but this is acceptable since we can classify these points at a later stage.

Region	Correct	Incorrect	Not classified
Overall	45.3	8.7	46.0
128x128	56.2	4.0	39.8
64x64	49.4	7.0	43.6
32x32	27.5	18.0	54.5
16x16	11.2	21.9	66.9

Table 7.6: Hypercylinder classification results (31×31 features), $r = 3\sigma$

7.4 Computational Complexity

The computational complexity of the three null-class methods described above varies significantly. The complexity can be examined for both the analysis that is done before the data is classified and for the data classification. The major problem with the above methods for implementing the null-class is that each class in the feature space can potentially be affected by a line segment between any pair of other class means. For an image with N texture classes, the number of line segments that must be tested for each class is $(N-1)(N-2)/2$. The total number of potential line segments is thus given by

$$N_L = N\frac{(N-1)(N-2)}{2} \tag{7.18}$$

which implies that in a worst case situation, the number of tests grows as the cube of the number of classes.

A reasonable metric to use in comparing the three techniques is the number of times a distance in the feature space must be computed. Finding the Mahalanobis distance between two points involves calculating a quadratic vector equation. The time required to do this type of calculation overshadows any scalar operations or linear vector operations such as adding two vectors.

In some cases, multiple quadratic vector quantities must be calculated to determine a single distance value. In the discussion below, let T_D represent the computational cost of doing one quadratic vector calculation.

For the pre-classification data analysis, the hypersphere method must calculate the distances from the class means to the Bayes class boundaries and from class means to the line segments. The distance to the Bayes class boundaries involves three quadratic vector quantities: two in the calculation the value of α (Eq. (7.6)) and one in calculating the distance from the mean to the point determined by the value of α. This must be done for each of the $N-1$ boundaries, for all N classes for a total cost of $N(N-1)3T_D$. The distance from the class means to the line segments is similar and it also requires three quadratic vector calculations. However this distance must be calculated for all line segments so the cost is $N(N-1)(N-2)3T_D/2$. The sum of these two values determines the result for the hypersphere pre-classification analysis. The hyperplane and hypercylinder methods are basically the same in the pre-classification phase in that both must determine which line segments pass through a particular region in the feature space. For each of N classes, all $(N-1)(N-2)/2$ line segments are examined by sampling points along the line. Each point must be classified, with a cost of NT_D, the cost of measuring the distance from the pixel to each class mean. If N_T point are tested on each line segment, the total pre-classification cost for both methods is $N(N-1)(N-2)N_TNT_D/2$.

The computational cost of doing the classification is highly dependent on the data. It can range from a worst case to a minimum as set by the classification algorithm with no null-class tests. The minimum per pixel cost for the nearest-mean classifier is simply NT_D. For the hypersphere method there is essentially

Method	Pre-classification	Classification		
		Best	Worst	Typical
Hypersphere	$\frac{N^2(N-1)3T_D}{2}$	0	0	0
Hyperplane	$\frac{N^2(N-1)(N-2)N_TT_D}{2}$	0	$\frac{(N-1)(N-2)T_D}{2}$	$2.86T_D$
Hypercylinder	$\frac{N^2(N-1)(N-2)N_TT_D}{2}$	0	$\frac{(N-1)(N-2)3T_D}{2}$	$8.63T_D$

Table 7.7: Summary of the computation complexity of three null-class methods

no further cost since only a scalar comparison of the minimum distance versus the sphere radius must be done. For the hyperplane, the worst case is testing the distance to the maximum possible number of pseudo-means, one for each line segment. This additional cost is given by $(N-1)(N-2)T_D/2$. For the hypercylinder, the worst case is having to test a cylinder for each line segment. Each test requires three quadratic vector measurements: two in finding the nearest point on the line segment and one in finding the distance from that point. The worst case addition cost is thus given by $(N-1)(N-2)3T_D/2$.

Table 7.4 summarizes the results of best and worst cases, and indicates a more likely cost based on the number of cases that had to be checked in classifying the texture mosaic. The table values for the classification phase only show the addition computational cost of the null-class test beyond the normal cost for the nearest-mean classification algorithm.

Chapter 8

Spatial Cohesion

In the discussion of classification vs. segmentation algorithms in Chapter 2, it was pointed out that the goal of image segmentation is to group the pixels into spatial regions. This is based on the assumption that the objects of interest in the image occur in relatively compact, connected regions of some minimum size as opposed to scattered and disconnected regions. The classification techniques we have used so far are all based on the features for an individual pixel. The classification decision for one pixel is not dependent on the decision made for any neighboring pixel, nor is any consideration made of the spatial arrangement of the data. To proceed from a classification of the texture data to a segmentation, information about the spatial arrangement of the pixels must be used. For example, if a point is believed to belong to class 1 but all the neighboring pixels in a surrounding region have been classified to class 2, then we should strongly consider classifying the pixel to class 2. A drawback to making a decision of this type is that no matter how we go about taking the surrounding neighborhood into consideration, situations can always be constructed that result in

segmentation errors and it would have been better off to ignore the neighboring pixels.

8.1 Simple Cohesion

To implement the use of spatial information, we need to measure the extent that a pixel matches its neighbors. This can be described as a type of cohesion measurement. This use of the term "cohesion" should not be confused with the term "coherence" as used in signal processing and analysis applications [54]. In the case of image texture, a pixel whose classification matches that of all of its neighbors in a surrounding area can be said to be maximally cohesive. A pixel that does not match any neighbors is non-cohesive. One of the simplest possible cohesion measurements is to count the number of pixels in some surrounding neighborhood that are classified to the same class as the pixel in question. If the pixel class assignments at point (x, y) are given by $H(x, y)$, then the cohesion, $C(x, y)$ is given by

$$C(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} \delta(H(x+i, y+j), H(x,y)) \tag{8.1}$$

where the Kronecker delta function is given by

$$\delta(a,b) = \begin{cases} 1 & \textit{if } a = b \\ 0 & \textit{otherwise.} \end{cases} \tag{8.2}$$

Figure 8.1 shows the result of measuring the cohesion of the classification results shown in Fig. 5.1b, which used the 31×31 features with no null-class. The cohesion of each pixel has been measured within a 15×15 neighborhood of the pixel. Maximum cohesion occurs when all 224 neighbors match the center pixel;

this is indicated by the lightest parts of the image. Minimum cohesion occurs when none of the neighbors match the center pixel and is indicated by the blackest parts of the image.

8.2 Multi-Class Cohesion

In a multi-class problem, a simple measurement of the degree of neighborhood cohesion is not sufficient to help the classification results. In a two class problem, low cohesion indicates that the pixel should perhaps belong to the other class. However, with more than two classes the cohesion measurement describe above does not indicate which of the alternative classes to select. A more useful piece of information can be obtained by measuring a set of cohesion values between the center pixel and its neighbors for each possible class assignment for the center pixel. The resulting vector of cohesion values (one for each class) is essentially a histogram of the pixels in the local neighborhood. It can be referred to as a *cohesion histogram* in which the value in each histogram bin is a measure of the cohesion of the center pixel with the pixel around it under a different assumption for the center pixel's class assignment. The kth element of the histogram vector at point (x, y) represents the cohesion of that point with neighboring pixels in class k and is given by

$$\mathbf{C}_k(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} \delta(H(x+i, y+j), k). \tag{8.3}$$

This is basically a generalization of Eq. 8.1 to include all possible class assignments for the pixel at (x, y). If that pixel is assigned to class k, then the kth element of the cohesion histogram vector is equal to the result of the simple

cohesion calculation

$$\mathbf{C}_k(x,y) = C(x,y). \tag{8.4}$$

Figures 8.2 and 8.3 show the results of measuring this cohesion histogram in a 15×15 window for classes 1 and 6 respectively of the classification results using the 31×31 features with no null-class. These two classes have been selected as examples since they represent a best-case and worst-case situation. The histogram results for class 1 are about what one expects with the bright areas showing where there is strong evidence of class 1 being present. The results for class 6 demonstrate one of the problems that can occur when measuring the cohesion histogram . Class 6 is heavily affected by the mixture density problem previously described. The result is that the cohesion for class 6 is excessively high in the areas with small regions. The cohesion histogram is useful for determining which class assignment is most likely for a pixel based on the neighboring pixel class assignments. A histogram with one bin that is dominant indicates a strong likelihood for that class. Conversely, a histogram that has an approximately uniform distribution of values indicates that no information is to be gained by observing the class assignments of the neighboring pixels.

The cohesion values described above are measured for each pixel using the class assignments of all the pixels within a window around the pixel. Since the cohesion must be recomputed for each position of the window and the window is different for each pixel in the image, measuring the complete cohesion histogram is a computationally demanding process. However, for a reasonably large window (say on the order of 15×15 or larger), a shift of one pixel vertically or horizontally does not appreciably change the data within the window. This fact can be utilized by calculating the cohesion values on a block basis rather

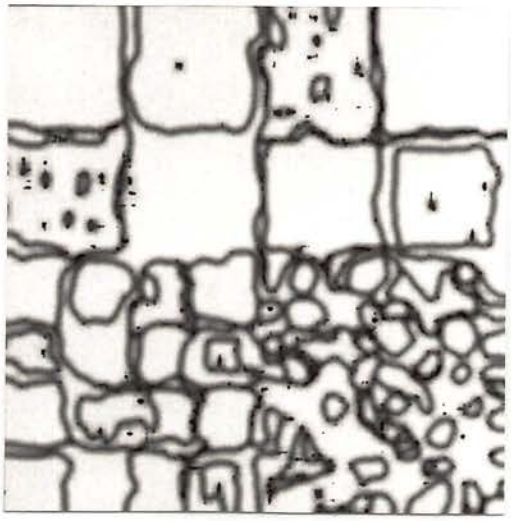


Figure 8.1: Pixel cohesion in a 15×15 neighborhood

Figure 8.2: Cohesion histogram - 15×15 neighborhood, class 1

Figure 8.3: Cohesion histogram - 15×15 neighborhood, class 6

than an individual pixel basis. The image is considered to be subdivided into rectangular blocks whose size is some fraction of the desired window size. The cohesion values are first calculated within each block and then the values from the proper blocks are combined for the final result. Figures 8.4 and 8.5 are the results for measuring the cohesion for classes 1 and 6 of the same image as before with the cohesion measured in a 5×5 neighborhood of blocks, where the dimensions of each block are 4×4 pixels. This gives an effective window area of 20×20 pixels. Some sensitivity to rapid variations in class assignments is lost when using a block method as opposed to a pixel method. However for the manner in which we use the cohesion histograms, the block method is sufficient.

8.3 Using Spatial Cohesion Information

When using the results of a spatial cohesion measurement it is important to understand the type of errors that can occur. For example, suppose we measure the cohesion in two regions of an image containing only two classes of pixels (Fig. 8.6). In one region the class assignments happen to alternate along rows and columns, much like a checkerboard. The classes in the other region are split down the middle into two separate, homogeneous blocks. It should be pointed out that if a texture classification produced results like the checkerboard pattern of region 1, this would indicate that the LAO used to generate the classification features was too small since the checkerboard pattern is in itself a texture. An LAO two times larger would be better suited to generating the features.

Both regions of Fig. 8.6 have the same number of class 1 and class 2 pixels. Therefore making a count of neighborhood pixels around the pixel in question

Figure 8.4: Cohesion histogram - 5×5 neighborhood of 4×4 blocks, class 1

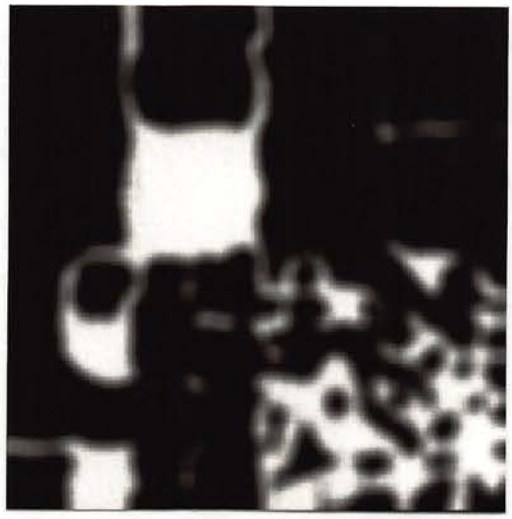


Figure 8.5: Cohesion histogram - 5×5 neighborhood of 4×4 blocks, class 6

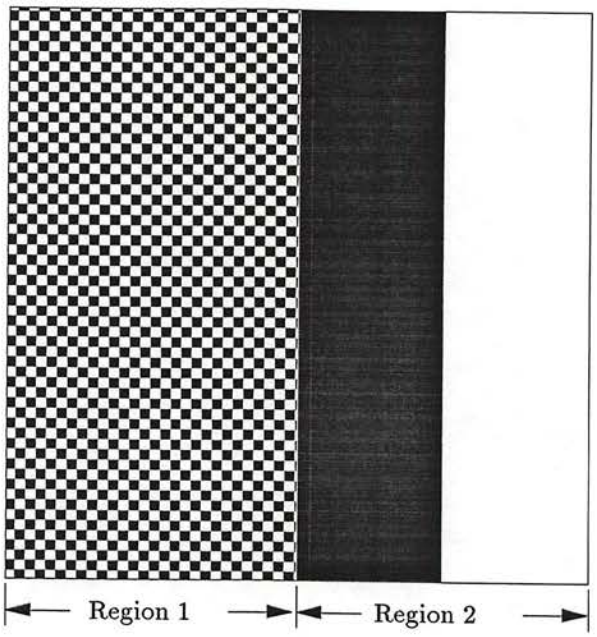


Figure 8.6: Two regions of image containing two classes

gives identical cohesion values for both regions. This result is opposite the intuitive notion of assigning a lower cohesion value to the checkerboard-like region since the pixels are more scattered. This implies that a cohesion measurement should be sensitive in some way to the arrangement of the neighboring pixels. To state this in another way, the cohesion measurement for a pixel should take into account the cohesion value of neighboring pixels, not just their class assignment.

One example of trying to make use of the spatial arrangement of the pixels in the surrounding region is used by Keller [55] in calculating the cohesion values for use in what he calls a *Lysing* process. The cohesion method he describes is for use on a two class image in that it calculates the cohesion for both the class to which the center pixel belongs and the other class. The cohesion values are based on the neighboring pixels in a 7×7 region centered on the pixel in

question. The cohesion calculation is done in two steps. First, each pixel in the central 5×5 region that belongs to the class under consideration is examined to see how many of its neighboring pixels are also assigned to that class. A second count is made of the possible neighbors of the pixels in the 7×7 perimeter of the block (outside the 5×5 but within the 7×7 region). This includes an estimate of the number of neighboring pixels that might lie outside the region under examination. The probability of their existence is based on the number of pixels of that class that are within the region. The final cohesion value is the first sum (neighbors of the pixels in the 5×5 block) minus the second sum (neighbors outside the 5×5 block). The calculation is done the same way for both classes. An advantage of this method of calculating cohesion is that the arrangement of the pixels in each class that lie in the surrounding region has a significant effect on the cohesion value. A checkerboard arrangement yields a smaller cohesion value than a block shape.

When calculating the cohesion histogram, the presence of points that are assigned to the null-class must also be accounted for. Points in the image that were assigned to the null-class during the classification using the first stage LAO are not counted when measuring the cohesion histogram. Since the histogram is used to guide the final classification of these null-class points, it should only be based on classification results with a high probability of being correct. Pixels that have a low probability of a correct classification should have been assigned to the null-class. These points are excluded from the cohesion histogram calculation and do not have an effect on the spatial information. Thus we redefine the cohesion for class k as the kth element of the cohesion histogram vector given

by

$$\mathbf{C}_k(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} g(x+i, y+j)\delta(H(x+i, y+j), k) \qquad (8.5)$$

where the function $g(x,y)$ is 0 for points in the null-class and 1 for points not in the null-class.

Figures 8.7 and 8.8 show the same cohesion histogram planes as Figs. 8.4 and 8.5 except that the histogram is based on the classification results using the hypercylinder method of generating the null-class. The most obvious differences can be seen in the the lower right portion of Figs. 8.5 and 8.8 where the region sizes are quite small. Figure 8.5 shows a significant amount of high cohesion pixels in this area. However this is due to many of the the pixels being classified to the "default" class as a result of the mixture density problem that was discussed in Chapter 6. This has resulted in high cohesion values that are based on very questionable pixel classifications. The cohesion of these same points is much lower in Fig. 8.8 since the use of the hypercylinder null-class resulted in many of these same points being classified into the null-class and are therefore not part of the cohesion measurement.

By using the results of the null-class test, we are also provided with some protection against assigning the same cohesion to regions that have the checkerboard and block distributions as described above. It is unlikely that pixels would be classified into the checkerboard pattern if the null-class test has worked properly since this type of pattern indicates a large amount of mixing of the class statistics. Most of the pixels are likely to be placed in the null-class. The resulting histogram would show a low count for all classes and be different from the histogram for the block distribution.

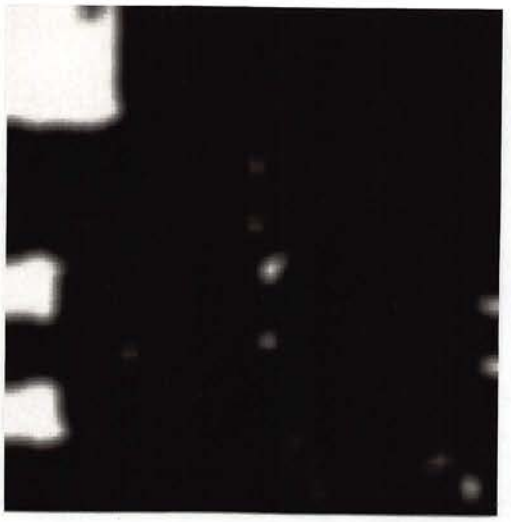


Figure 8.7: Cohesion histogram - 5×5 neighborhood of 4×4 blocks, with null-class exclusion, class 1

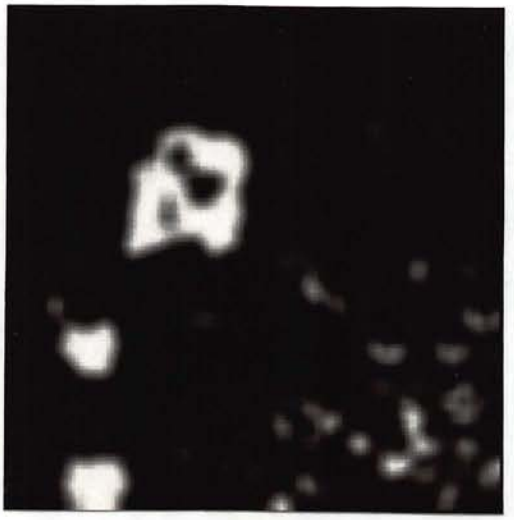


Figure 8.8: Cohesion histogram - 5×5 neighborhood of 4×4 blocks, with null-class exclusion, class 6

An underlying basis for using cohesion measurements such as these is that small, isolated regions are not likely to occur in images and should be removed if encountered. A small region should return a low enough cohesion value to cause subsequent processing to reclassify its members to the class of the surrounding region. The cohesion measurement is one component of the hierarchy of information that must be combined to determine the final class assignment.

By examining the values in the cohesion histogram, we can determine the presence of nearby regions that are of both sufficient size to warrant consideration when classifying points between them. The cohesion of the pixel with each potential class in the surrounding neighborhood is used to determine which class assignments are most likely to be correct on the basis of having a high cohesion for that class assignment. Similarly, the cohesion measurements are used to eliminate from consideration certain classes with low cohesion.

Obviously, the size of the area used for calculating the cohesion histogram has a significant impact on this process. At a minimum, the cohesion measurement should cover enough spatial area to detect the presence of any nearby regions with a significant proportion of pixels classified. If the area is too large, it may encompass other regions that do not actually border on this particular unclassified area. If the area is too small, the classified regions nearby are never found. This use of spatial information is described in the next chapter. By using the spatial information in this way, we can compensate for the typically high error rate that occurs when classifying the data using only the features generated with the smaller LAO, and this results in a better overall performance.

Chapter 9

Segmentation Process

9.1 Hierarchical Segmentation

In a segmentation problem with a variety of information sources, a major consideration is how to combine the information in such a way achieve the optimum results. In a hierarchical decision process, certain pieces of information are considered before others and may determine the manner in which information lower in the hierarchy is used. In a non-hierarchical decision process, all the sources of information are used, although they may have different degrees of influence on the final decision.

Keller [55] describes a hierarchical decision method for implementing his Lysing process in a two-class problem that is used to decide whether to use the original class, switch to the other class, or reclassify based on some new criterion. The available information consists of the cohesion values of the two classes and the values of the pixels in a local neighborhood. To summarize his decision process, if the class of the center pixel is cohesive and the other class is

not cohesive, replace the center pixel with the median of the neighboring pixels that are in the center class. If the center class is not cohesive and the non-center class is, replace the center pixel with the median of the neighboring pixels that are in the non-center class. If both classes are non-cohesive, replace with the median of all neighboring pixels. If both classes are cohesive, replace with the median of one of the two classes depending on the number of neighboring pixels in each class. This represents a hierarchical decision process since the values of the neighboring pixels that belong to one of the classes are not considered if that class is non-cohesive and the other class is cohesive.

The multi-resolution classifying scheme discussed in the previous chapters provides us with the following sources of information:

- The classification choice using features generated using the large LAO (including the null-class).
- The cohesion histogram giving spatial information about the surrounding area.
- The classification choice using features generated using the small LAO.

As we have seen in previous sections, all of these are prone to error under certain conditions. However we also have available our knowledge of how these parameters are likely to perform in different areas of the image. By using this knowledge when combining the above information, the decision process can be made less sensitive to lack of performance in any single parameter.

The basic hierarchical segmentation process is shown in Fig. 9.1 and can be stated as follows: If a point was not classified using the large LAO features (it was assigned to the null-class), classify the point using the small LAO features,

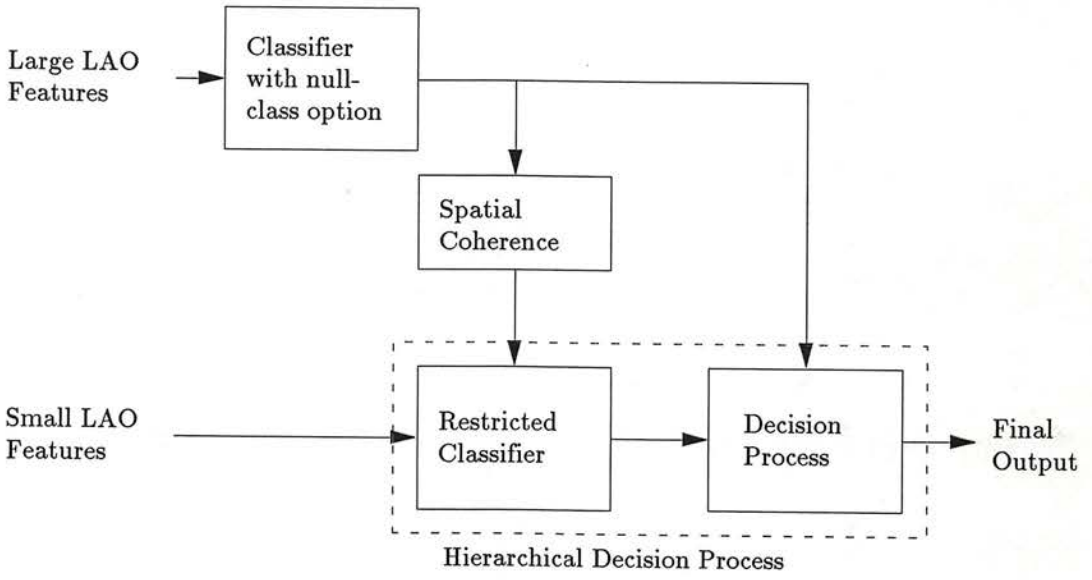


Figure 9.1: Block diagram of hierarchical segmentation process

with the cohesion histogram used to influence the classification by eliminating certain classes from consideration for the final choice. Humans appear to do this type of operation when classifying regions of texture in a scene. A human instinctive restricts the choice of class assignments based on the surrounding regions. If we see a large, homogeneous region to the left that we have decided is sand and a similar region to the right which seems to be wool, we tend to classify the points between them as either sand or wool but not tree bark, even if it looks a lot like bark.

One of the goals of the segmentation process is to eliminate small, isolated regions from the final output by merging them into a neighboring region. While the altering of a small region is often the correct action to take, it is also possible that in many cases the region should not be changed. This problem is not unique to this method but occurs in any segmentation technique that must decide

whether a small region of one class belongs to a large region of another class that surrounds it. Any segmentation scheme has an implicit minimum region size. Regions that fall below the threshold are merged into the surrounding region. This can occur for small, isolated regions and for protuberances of a larger region. Use of the cohesion results can leave corners or other narrow extensions of regions rounded off or otherwise reshaped since these pixels may lack sufficient cohesion to warrant being left intact. This is in part due to the size and shape of this type of region, leading to low cohesion, and also because it is likely that many of the pixels were classified into the null-class due to the mixture density effects of the boundary. The combination of these effects results in a significantly lower value in the cohesion histogram. A hypothetical example of this can be seen in Fig. 9.2 where region 2 becomes too narrow for the large LAO features to accurately classify it. The cohesion measurements taken in the region 2 area would not indicate a high cohesion for class 2 in the surrounding area. In this case all of the points in this part of region 2 would probably be classified to either textures 1 or 3 and the resulting classification might look like Fig. 9.3.

9.2 Non-Hierarchical Segmentation

To properly evaluate the results of the hierarchical segmentation using multiple size operators, a comparison must be made with the results of a non-hierarchical segmentation based on single size operators. This is done using the results of the full classifications (no null-class) that were described in Chapter 5 and shown in Figs. 5.1b, 5.1c, and 5.1d. These represent classification of all the data

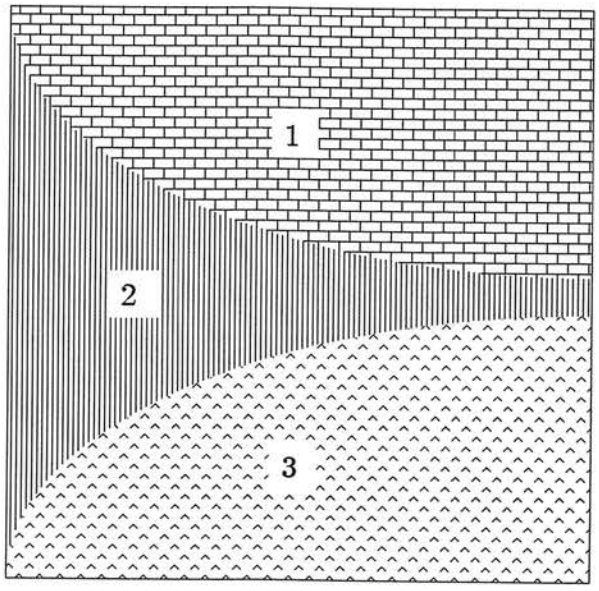


Figure 9.2: Hypothetical texture image

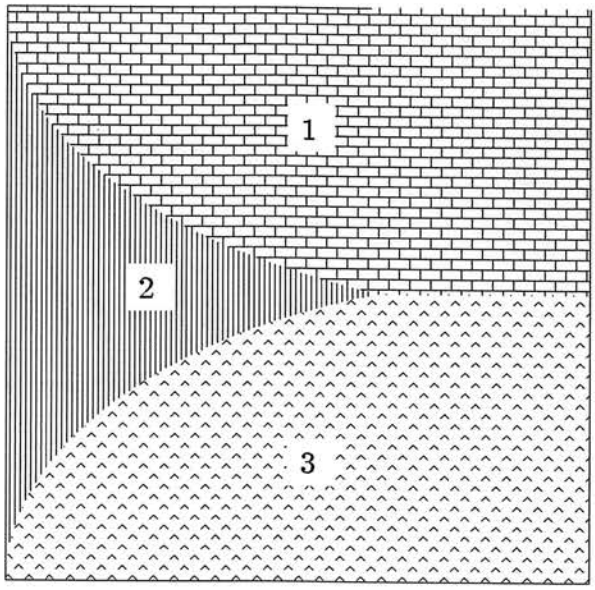


Figure 9.3: Resulting classification

using 31×31 features, 15×15 features, and both sizes simultaneously. Three cohesion histograms are calculated from these results in the same manner as they were calculated from the results of the three null-class classifications. The only significant difference is that the sum of the pixel counts in the cohesion histogram bins for each block of pixels is the same for each block and is equal to the number of pixels being analyzed in the local area. This is because no pixels were excluded from the histogram count due to being in the null-class.

The decision process for the multiple resolution operators, hierarchical segmentation was described on page 108. The equivalent decision process for assigning pixels to a texture class with a non-hierarchical segmentation technique is shown in Fig. 9.4 and can be stated as follows: For all points in the image, classify the point using the features, with the cohesion histogram used to influence the classification by eliminating certain classes from consideration for the final choice. There are some significant differences between the two segmentation methods. Both classification phases of the non-hierarchical segmentation use features based on the same size LAO. This implies that we are actually classifying the data twice with the same features, once to provide input for calculating the cohesion histograms, and later to select the final assignment for each pixel. The second major difference is that the coherence histogram is used to determine the final class assignment for all pixels, rather than just those assigned to the null class by the first classification.

The segmentation method just described obviously does not represent an effort to develop the best possible segmentation method that uses information from single resolution operators in a non-hierarchical manner. It is designed to operate in a similar manner on some of the same information that is used in the

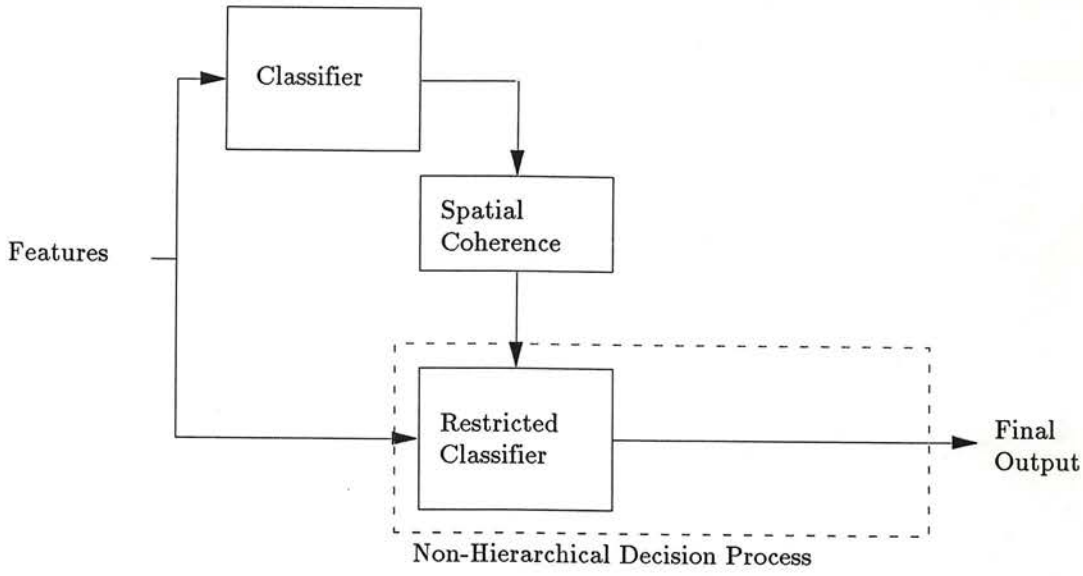


Figure 9.4: Block diagram of non-hierarchical segmentation process

more sophisticated hierarchical segmentation method so as to make comparisons between the two approaches possible.

9.3 Decision Methods

Implementing the hierarchical segmentation process basically involves the selection of the criteria for classifying points using the large LAO features and the design of the class elimination rule for making the final decision on the class assignment. For implementing the non-hierarchical segmentation process, only the class elimination rule is needed since the classifier does not use a null-class. Various methods for determining whether or not to classify a point were discussed in Chapter 7. This chapter discusses a variety of methods that have been tried for the class elimination rule. There does not appear to be any solid

quantitative way to develop a decision rule. The differences in the techniques described here are mostly small, evolutionary changes that were made to avoid various problems that appear with each new version. The same rule is used for both the hierarchical and non-hierarchical segmentation processes.

The results shown in this section are all based on the texture energy feature generated with 31×31 and 15×15 standard deviation operators. For the hierarchical segmentation results, the classification using the large LAO features was done using the three null-class methods previously described (hypersphere, hyperplane, and hypercylinder). For the non-hierarchical segmentation results, full classifications were done with the two sizes of features and both sizes together. The cohesion histogram was calculated using 8 by 8 blocks with an LAO size of 5 by 5 blocks (40 by 40 pixels). These parameters were chosen since they appeared to result in the most promising intermediate data.

The following sections describe a variety of decision methods that have been used to segment the image using both the hierarchical and non-hierarchical process. Most of the discussion of the decision methods concerns the effect of the on the hierarchical process. Each section describes the method used and presents results in three forms.

- The final segmented images are shown for each of the three hierarchical segmentation methods. The results of the non-hierarchical segmentation are not shown in image form. However, a sampling of some of the non-hierarchical segmented images are shown in Section 9.4 with a summary of the results.

- The percentage of correctly assigned pixels are listed in a table. Values are listed for the overall image and for the four sizes of regions in the image. The tables contain the results for both the non-hierarchical and the hierarchical segmentation processes.
- The results listed in the table are graphed showing the performance in each of the four sizes of regions in the image. The plots show the results of the hierarchical segmentation using all three null-class methods, the non-hierarchical segmentation at two resolution sizes, and the classification at two resolution sizes. The results of the non-hierarchical segmentation and the classification using the 31×31 and 15×15 features together are not graphed since the results are very close to those obtained with the 31×31 features alone.

9.3.1 Method 0: Ignore histogram

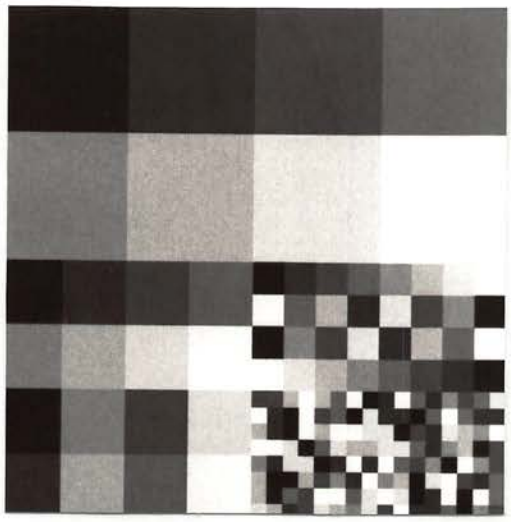
Probably the simplest way to implement a decision rule is to ignore the cohesion histogram altogether. With this method, the final segmentation choice is solely a function of the large and small LAO classification choices. If a pixel is classified into one of the texture classes by using the large LAO feature, then that class assignment is the final assignment. If the pixel is placed in the null-class, then the final assignment of the pixel is the classification using the small LAO features. This method ignores all spatial information such as that available from the cohesion histogram. For this reason, this method does not really qualify as a segmentation of the image since it is not making use of any spatial information in deciding on the final assignment of the pixels. For the non-hierarchical

method, the results are the same as with a simple classification of the data. The results of using this rule are shown in Table 9.1 and Figs. 9.5 and 9.6. Comparison with the the single size operator classifications show that this method does provide an improvement in many areas of the image. However, since the spatial information was ignored, there is little improvement in shrinking or removing the false regions that appear between correct regions.

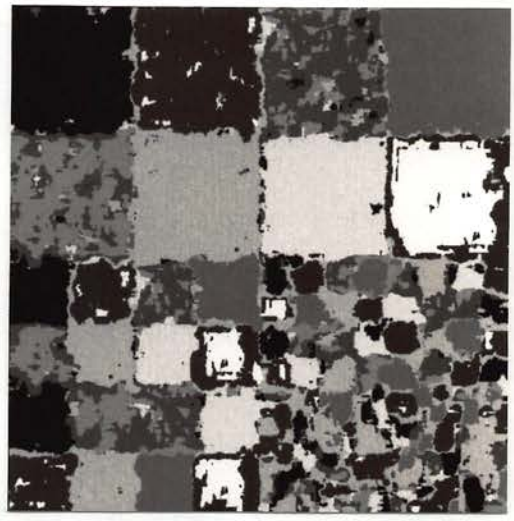
9.3.2 Method 1: Fixed number of classes

For the pixels that are originally classified into the null-class, the cohesion histogram data can be used to implement a variety of more sophisticated decision rules. Generally the goal here is to use the spatial information contained in the histogram to shrink or eliminate some of the smaller regions. A typical rule to follow might be to only allow assignment to a limited number of the more dominant classes as indicated by the histogram values. Intuitively this makes good sense since we want to assign the pixel to a class that already exists nearby in the image.

Table 9.2 and Figs. 9.7 and 9.8 show the results of implementing a relatively simple decision rule that makes use of the information in the cohesion histogram. In this case, the final segmentation choice for pixels not classified when using the large features is limited to assignment to either of the two most dominant bins as determined from the cohesion histogram. We can see that this rule leads to problems in the areas of the image where there are small regions. The mixture problem discussed in Chapter 6 is very prevalent in these areas since the regions are virtually the same size as the LAOs being used to generate the features. There is very little chance of ever improving the segmentation accuracy for the

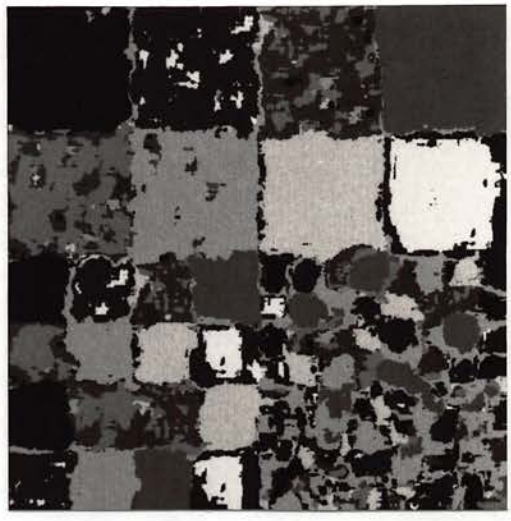


(a) Gray-level texture map



(b) Hyperspheres

(c) Hyperplanes



(d) Hypercylinders

Figure 9.5: Method 0: Ignore histogram

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	68.1	69.9	68.5	72.7	73.6	71.4
128×128	83.2	78.2	83.4	82.7	85.6	81.6
64×64	70.2	71.6	70.8	72.2	75.3	74.9
32×32	48.4	58.1	48.5	58.8	57.1	56.0
16×16	23.5	44.9	24.2	43.7	39.1	39.0

Table 9.1: Method 0: Ignore histogram

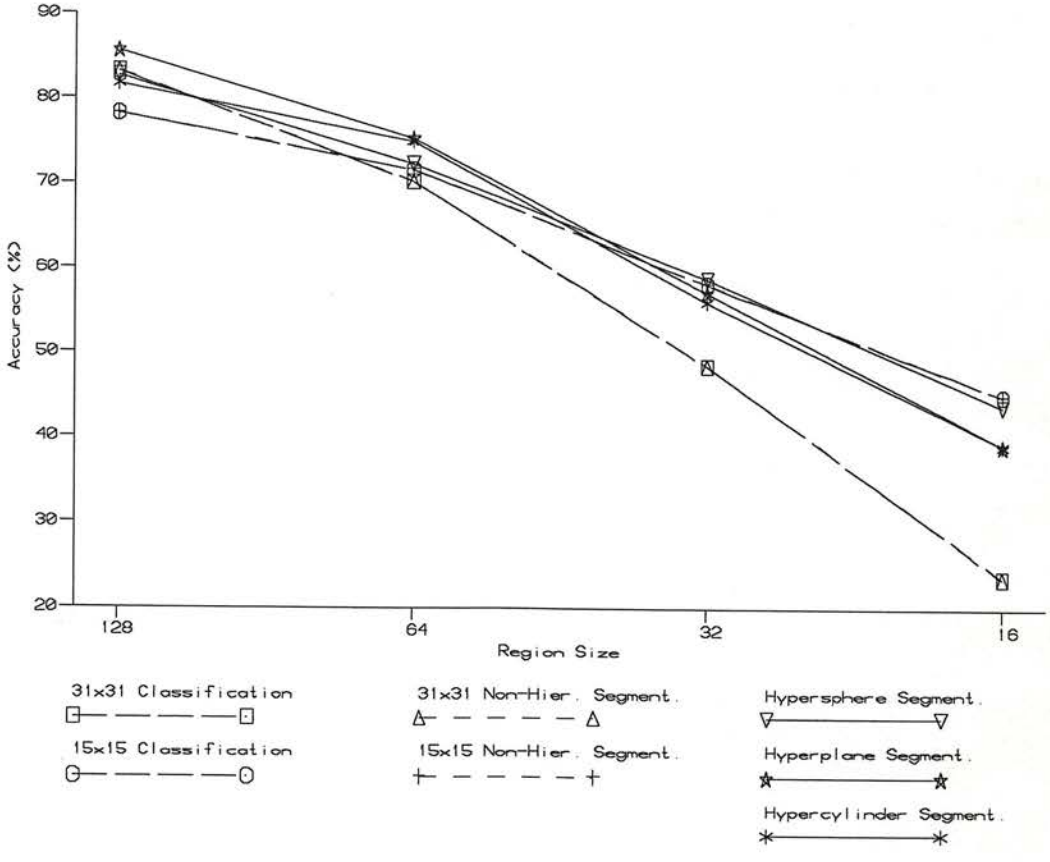


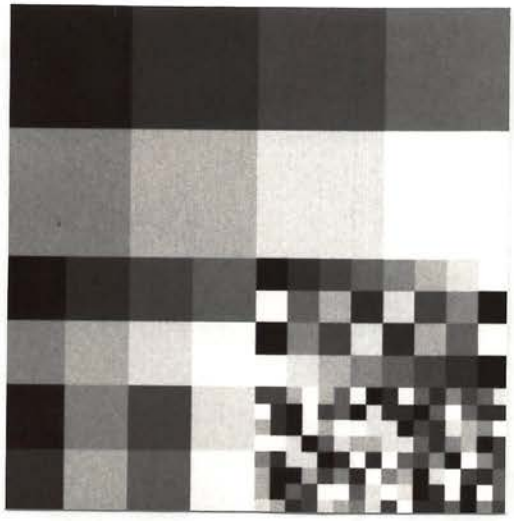
Figure 9.6: Graph of results for method 0

data in regions of this size without going to a smaller LAO for feature generation. The null-class methods discussed in Chapter 7 created null classes for guarding against mixtures of pairs of classes. In the areas of the image where the texture regions are very small, many of the feature points are based on the mixture of more than two textures, which the null-classes are not designed to detect. Many of the pixels in these areas of the image are assigned to the class that has its mean closest in the feature space to the expected location of these higher-order mixtures. The overall effect is that of having a default class for pixels in areas where there are small regions.

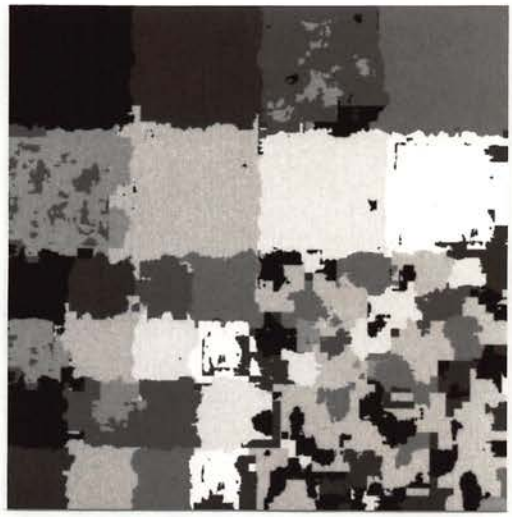
Even with an effective method for detecting mixture points, a significant number of points are incorrectly assigned to this default class by the classification using the large LAO features. The cohesion histogram bin for that class can contain a significant value. This same bin often turns out to be dominant for most pixels in this general area of the image. Since this dominant class is usually incorrect, limiting the classification to this class and perhaps one or two others generally results in poor performance in the areas of the image containing small regions. While this decision method has helped somewhat, it is obvious that the choice should not always be limited in such a simple-minded manner.

9.3.3 Method 2: Adaptive Threshold

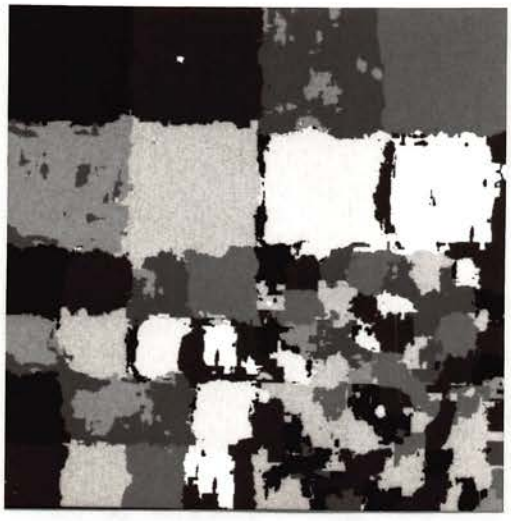
A variation on the above method is to allow classification to any class that has its histogram value above a threshold. However, setting an absolute level for the histogram threshold leads to problems. In many places in an image, none of the classes exceed this value leaving no classes eligible for classification. More common are places where only one class exceeds the threshold, making



(a) Gray-level texture map



(b) Hyperspheres



(c) Hyperplanes

(d) Hypercylinders

Figure 9.7: Method 1: Fixed number of classes

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	70.3	73.8	70.3	75.1	75.5	74.6
128×128	85.3	83.3	85.3	89.1	88.4	89.0
64×64	73.5	76.8	74.6	82.1	78.1	78.4
32×32	50.5	65.3	48.3	50.5	60.4	54.9
16×16	24.0	38.0	23.8	29.1	33.6	29.3

Table 9.2: Method 1: Fixed number of classes

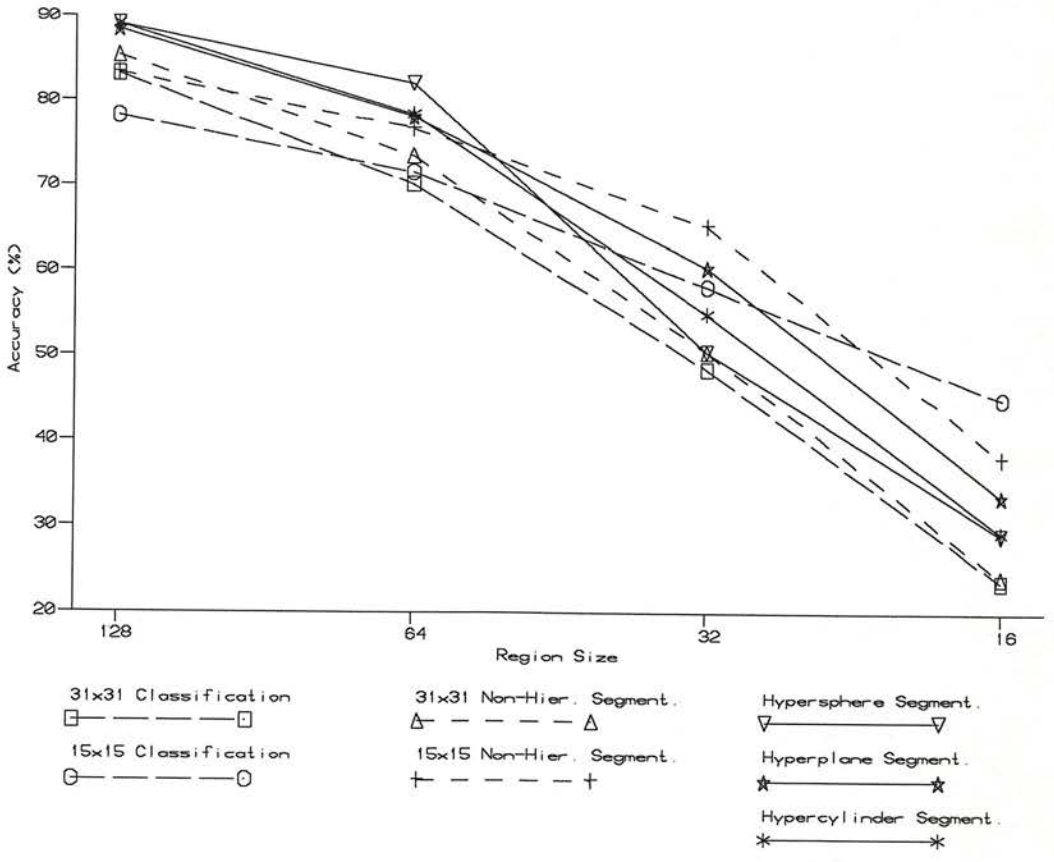
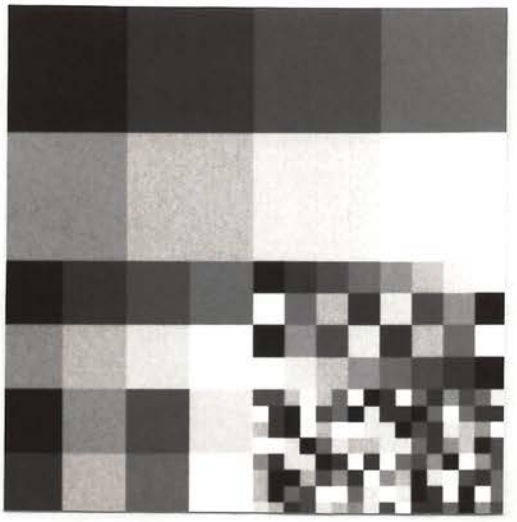


Figure 9.8: Graph of results for method 1

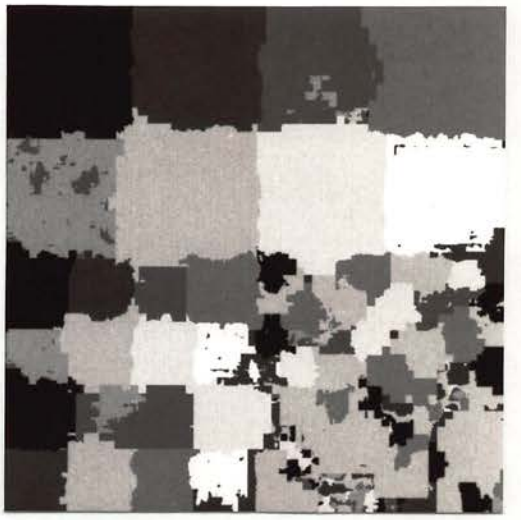
the resulting classification a function of only the large LAO features. A possible way to avoid this situation is to set the level as a fraction of the largest value in the histogram. In theory, this will result in more classes being considered while simultaneously eliminating the least likely classes. The result of not allowing classification to any class with less than 25% of the maximum value in the histogram is shown in Table 9.3 and Figs. 9.9 and 9.10. The results show a slight improvement in the larger regions but a decrease in performance in the small regions. The 16 by 16 regions are classified very badly, with most pixels classified to the same class. This is probably due to the spatial cohesion calculation not finding many points that were assigned to something other than the null-class. The resulting histogram may not have any other classes above the threshold determined by the dominant class.

9.3.4 Method 3: Conditional fixed threshold

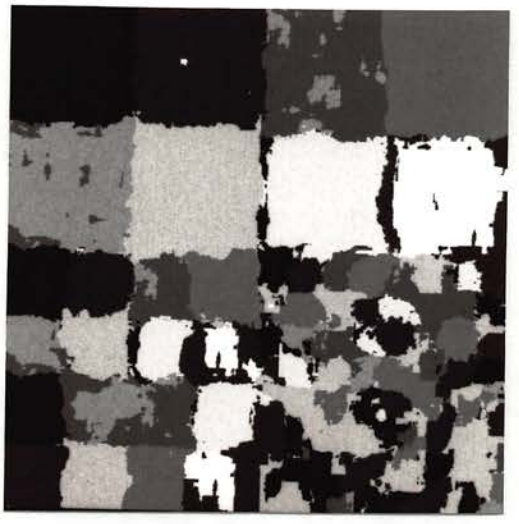
It becomes obvious when testing decision methods is that it is wrong to always try to eliminate certain classes from consideration based on the data in the cohesion histogram. There should be circumstances under which all classes are considered. This should occur in areas where there are very few previously classified points. If the concentration of classified points is sparse, it is likely that the classifications are incorrect and the values in the histogram can be treated as worthless. In this situation, comparing the histogram values against the dominant values as used in the previous method results in the classification being limited to a small number of incorrect choices. To avoid this situation, a test can be made on the histogram data to determine if any limits should be placed on the classification. If the data in the histogram does not pass this



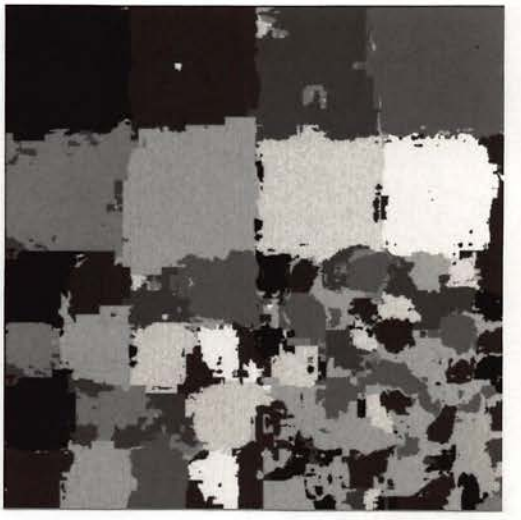
(a) Gray-level texture map



(b) Hyperspheres



(c) Hyperplanes



(d) Hypercylinders

Figure 9.9: Method 2: Adaptive threshold

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	68.9	73.4	69.1	76.2	75.5	74.6
128×128	84.7	83.4	84.6	90.3	88.6	89.7
64×64	70.7	74.7	71.4	84.8	78.6	76.9
32×32	49.0	59.5	48.7	52.1	59.8	53.9
16×16	21.8	44.1	22.3	26.7	32.3	30.3

Table 9.3: Method 2: Adaptive threshold

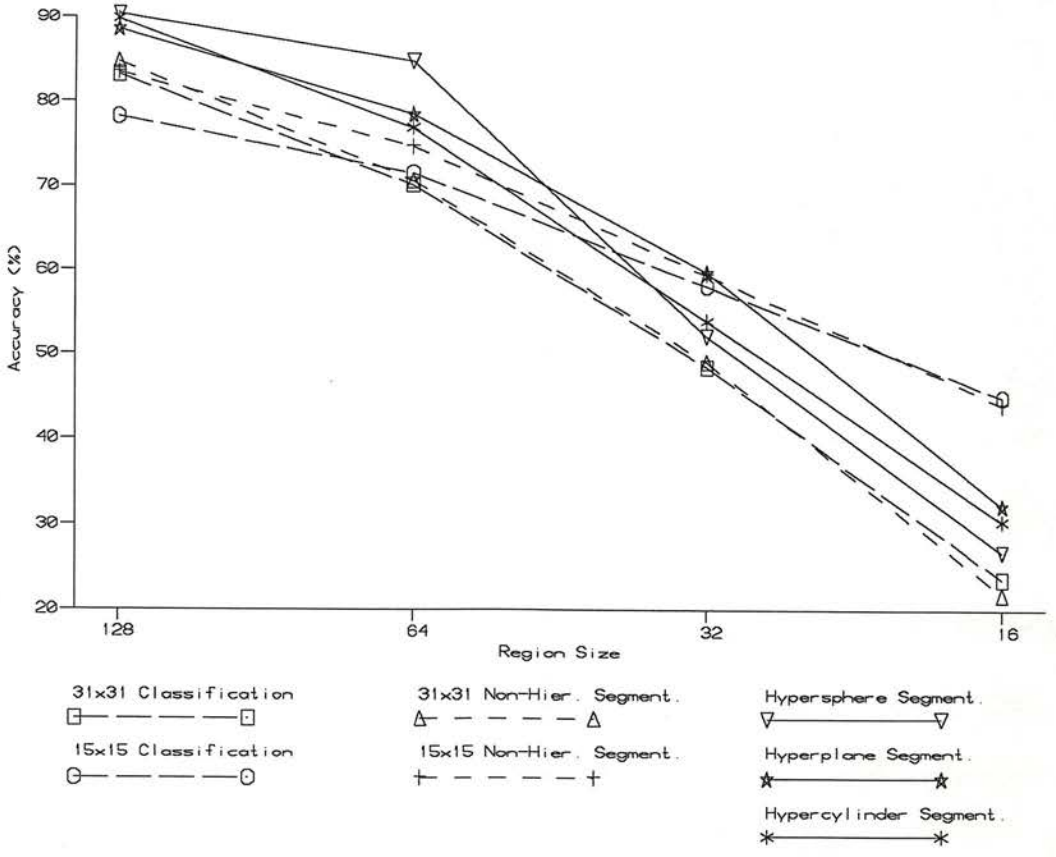


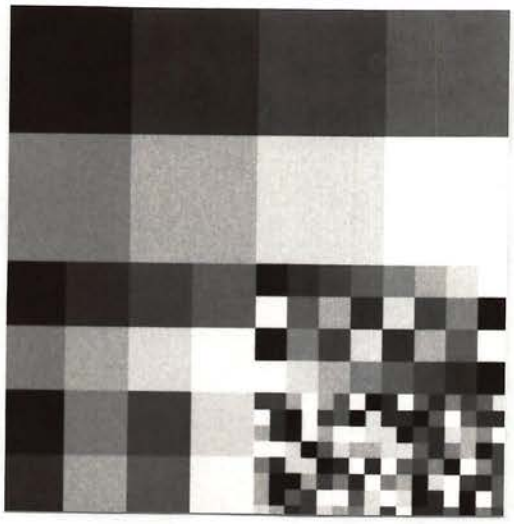
Figure 9.10: Graph of results for method 2

test, then the histogram is ignored and all classes are available for the final assignment.

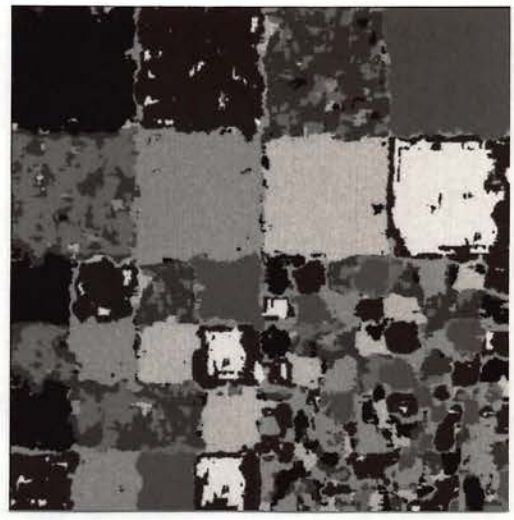
Table 9.4 and Figs. 9.11 and 9.12 show the result of applying a test of this type. The test for this method is based on a fixed threshold rather than a threshold that is a function of the data. Final classification is allowed to any class that has a histogram value greater than 25% of the maximum possible value. However, if the number of allowed classes is less than 2, then all classes are allowed for final classification.

9.3.5 Method 4: Conditional threshold with fixed number of classes

Another way to avoid an overly restrictive final classification is by testing the value in the dominant bin against a somewhat high threshold. If the value in the dominant bin does not exceed the threshold, then all classes must be considered. Table 9.5 and Figs. 9.13 and 9.14 show the result of using this rule. Here, assignment was allowed to all classes unless the dominant histogram value exceeded 50% of the maximum attainable histogram value. This implies that all classes were allowed in areas where less than 50% of pixels in the local neighborhood were assigned to the same class. In areas where the dominant class was over the 50% threshold, assignment was limited to one of the two most dominant classes based on the values in the histogram bins.

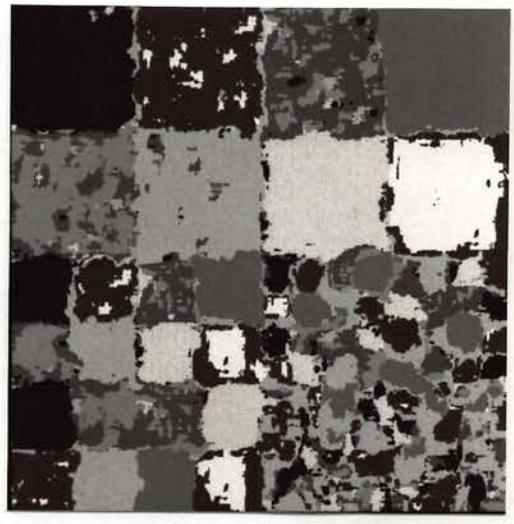


(a) Gray-level texture map



(b) Hyperspheres

(c) Hyperplanes



(d) Hypercylinders

Figure 9.11: Method 3: Conditional fixed threshold

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	69.6	72.0	69.9	72.7	74.2	71.6
128×128	84.8	80.7	84.7	82.7	86.1	81.8
64×64	72.1	74.4	73.6	74.2	76.4	75.1
32×32	49.6	62.2	49.2	58.8	57.7	56.2
16×16	23.9	42.1	24.1	43.7	38.6	39.0

Table 9.4: Method 3: Conditional fixed threshold

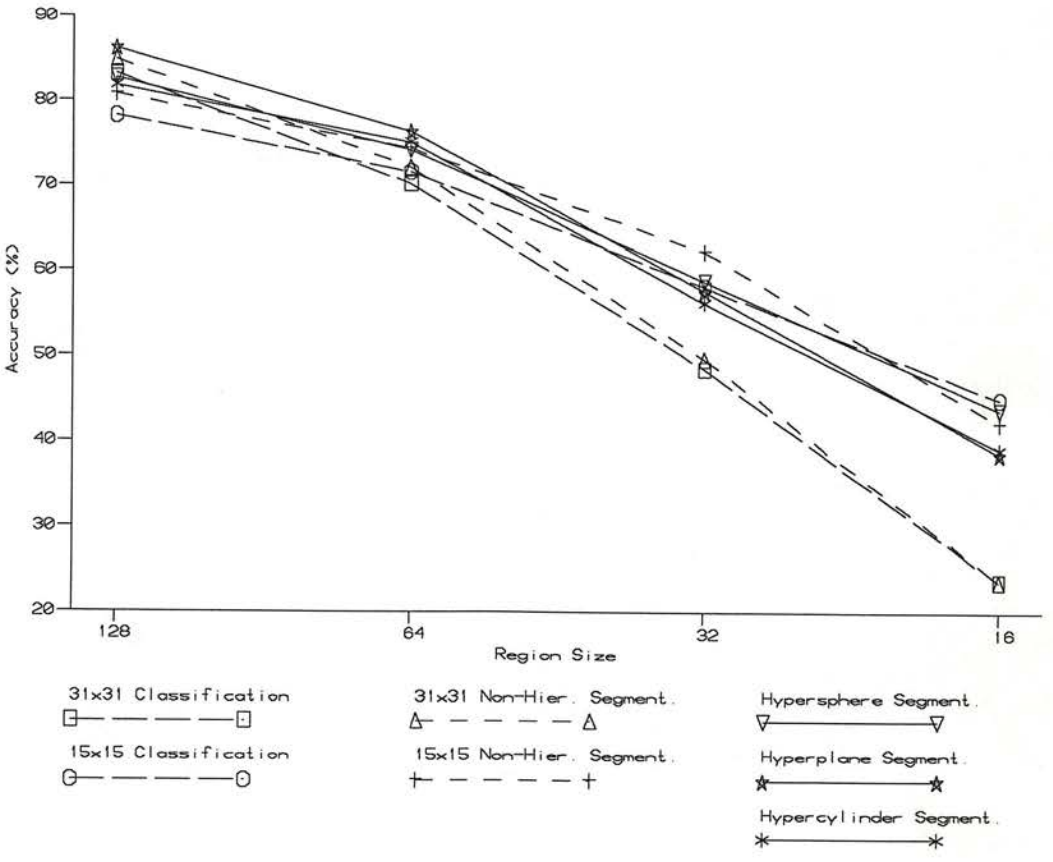
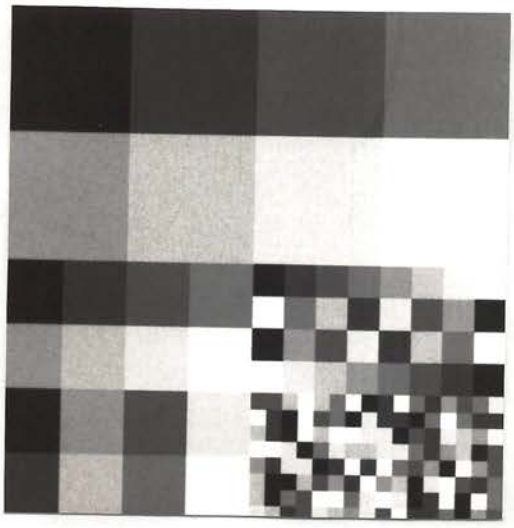
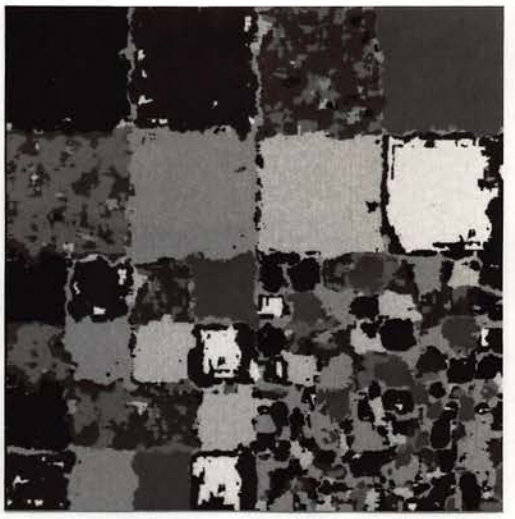


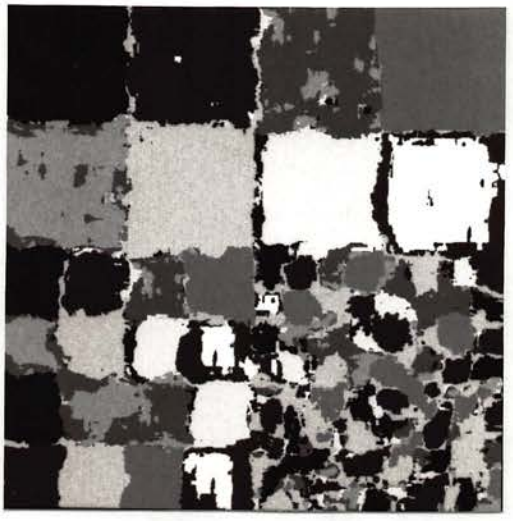
Figure 9.12: Graph of results for method 3



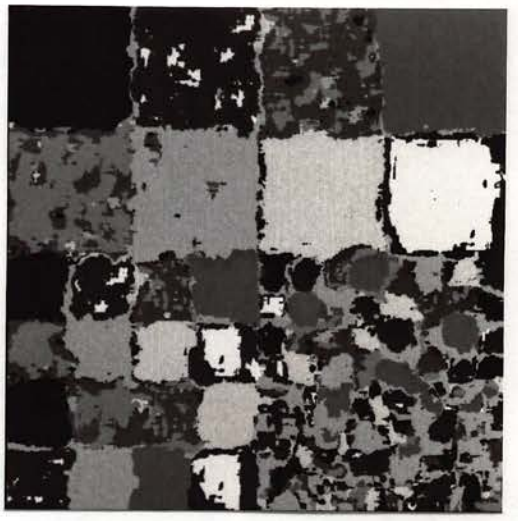
(a) Gray-level texture map



(b) Hyperspheres



(c) Hyperplanes



(d) Hypercylinders

Figure 9.13: Method 4: Conditional threshold with fixed number of classes

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	68.6	71.5	68.9	72.9	74.0	71.6
128×128	83.7	80.4	83.8	83.1	86.1	82.0
64×64	70.5	73.7	71.4	74.3	75.6	75.0
32×32	48.7	58.5	48.7	58.8	57.2	56.0
16×16	24.1	44.9	24.3	43.7	39.2	39.0

Table 9.5: Method 4: Conditional threshold with fixed number of classes

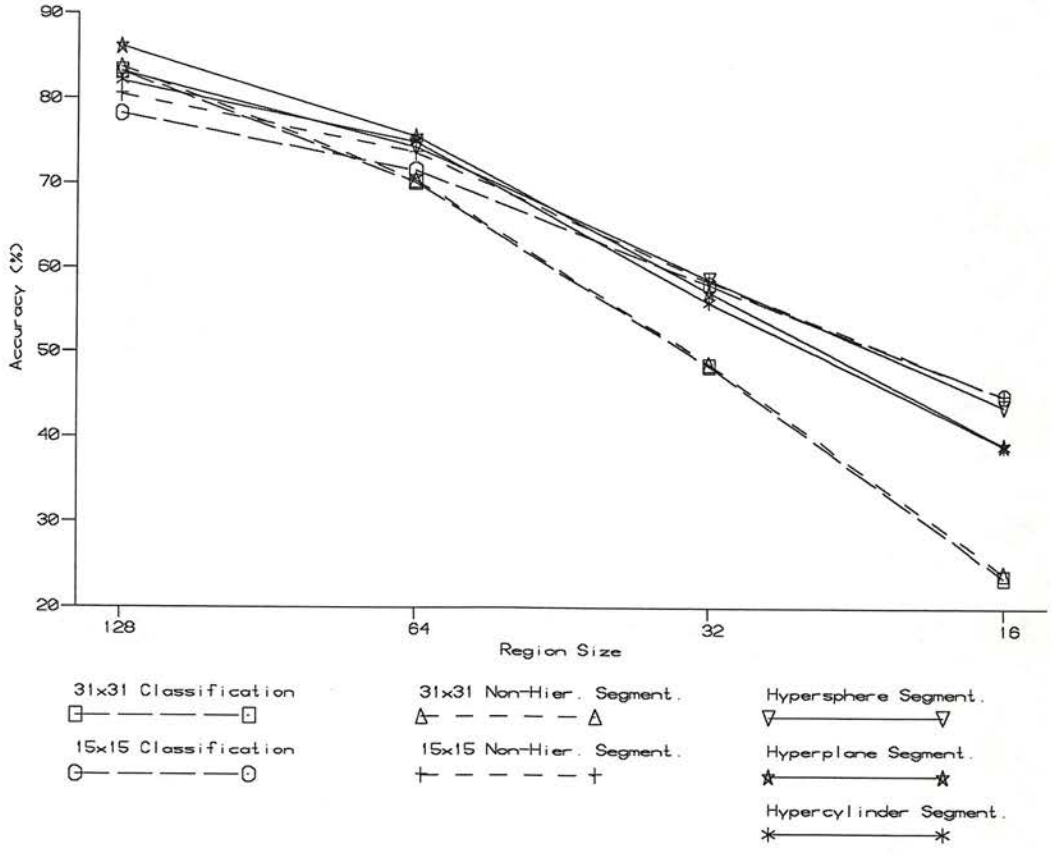


Figure 9.14: Graph of results for method 4

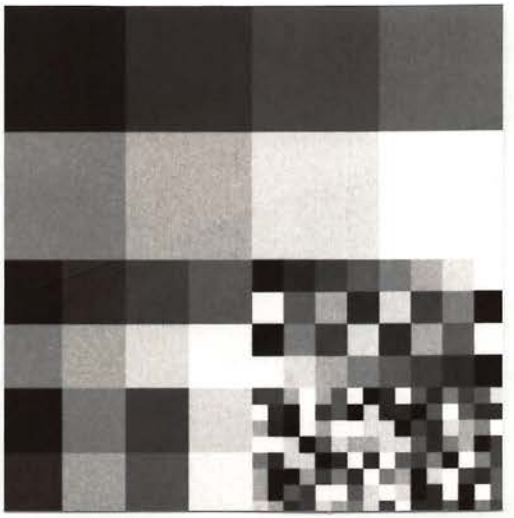
9.3.6 Method 5: Conditional threshold with variable number of classes

A variation on the above rule is shown in Table 9.6 and Figs. 9.15 and 9.16. In this method, if the dominant histogram value exceeds the threshold of 50% of the maximum attainable value, classification is allowed for any class that has a histogram value above 15% of the same maximum. If the dominant histogram value is not above the 50% threshold, then classification is not restricted and any class may be picked for the final assignment. This differs from the previous rule in which only the two most dominant classes were allowed. By adjusting the two thresholds, the number of alternative classes allowed can be varied. This type of rule appears to be better than most. It includes capabilities to avoid most of the problem situations that have caused trouble for the earlier methods. Visual inspection of the image shows that it does a reasonably good job of reducing the size of the false regions between the true regions, and does not increase the number of error in the middle of the large regions.

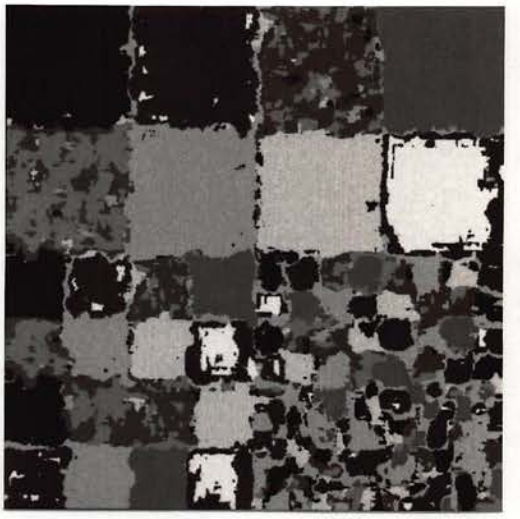
9.4 Summary of Test Results

Figures 9.17 shows some of the output images from the non-hierarchical segmentation process. These are shown for purposes of comparison with the corresponding output images of the hierarchical segmentation process.

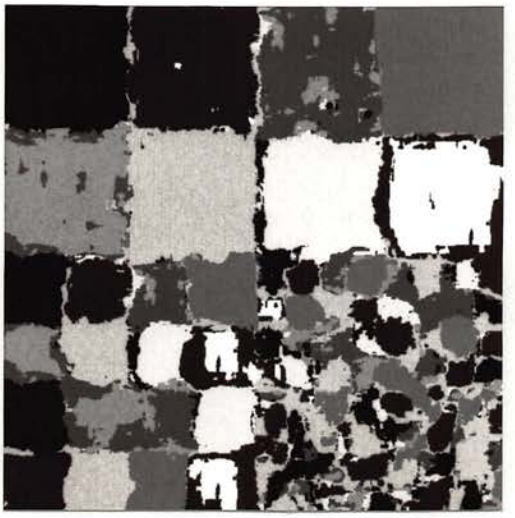
Table 9.4 summarizes the performances of the six methods described above for segmenting the entire image. The above decision methods are by no means a complete set. By changing the decision rules slightly, or modifying the thresholds, one can generate any number of different decision processes. In above rules,



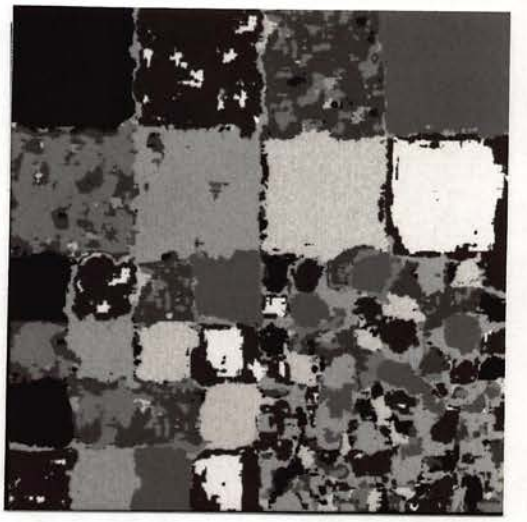
(a) Gray-level texture map



(b) Hyperspheres



(c) Hyperplanes



(d) Hypercylinders

Figure 9.15: Method 5: Conditional threshold with variable number of classes

	Non-hierarchical			Hierarchical		
	31×31	15×15	31×31 + 15×15	Sphere	Plane	Cylinder
Overall	68.6	72.4	68.9	73.0	74.2	71.8
128×128	84.2	82.0	84.3	83.2	86.4	82.2
64×64	70.4	74.1	71.2	74.4	75.8	75.3
32×32	48.6	58.5	48.7	58.8	57.2	56.0
16×16	22.8	44.9	23.3	43.7	39.2	39.0

Table 9.6: Method 5: Conditional threshold with variable number of classes

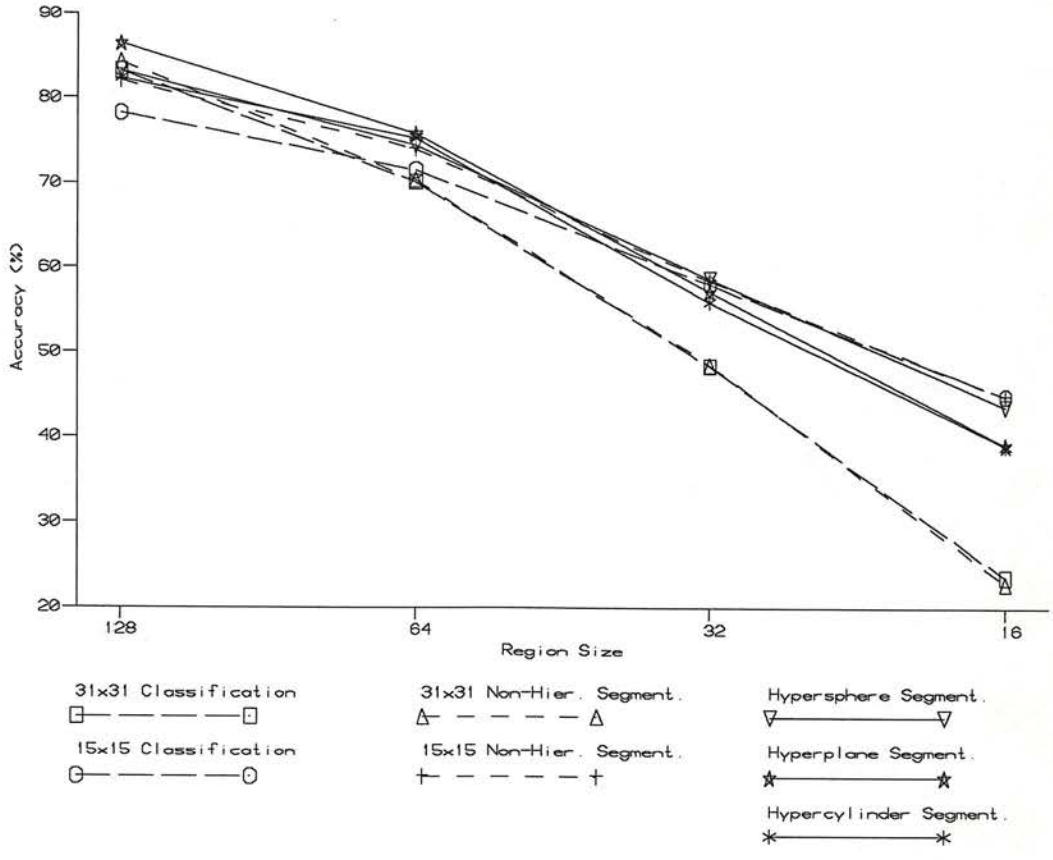
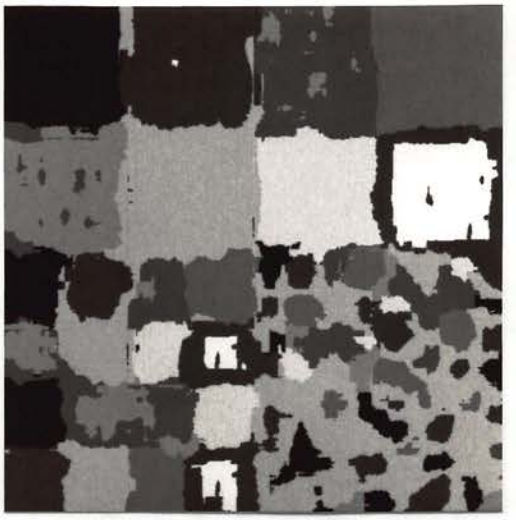
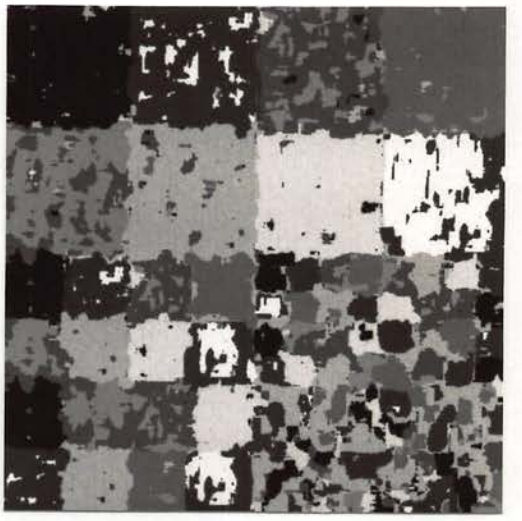


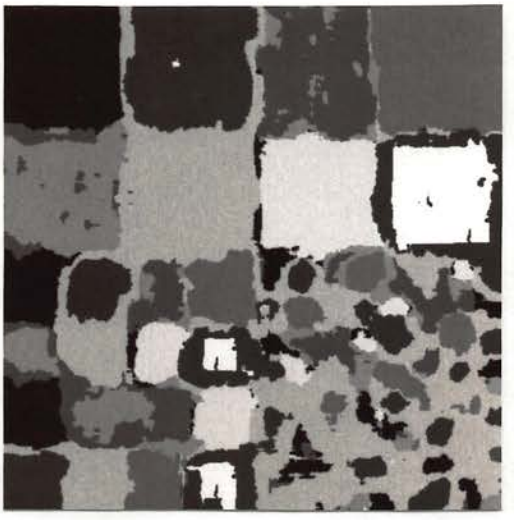
Figure 9.16: Graph of results for method 5



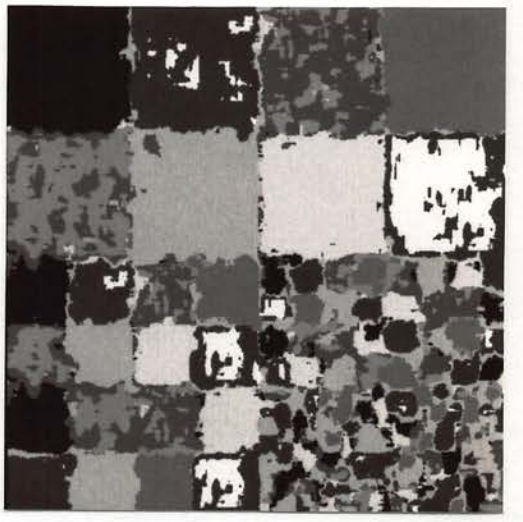
(a) 31×31 – Method 1



(b) 15×15 – Method 3



(c) 31×31 and 15×15 – Method 4



(d) 15×15 – Method 5

Figure 9.17: Non-hierarchical segmentation results

only one or two courses of action were allowed, such as "restrict classification to these classes" or "allow classification to any class". More complicated rules could be devised that use more than two alternatives.

The above tests illustrate that flexibility is the most important requirement of the decision rule. The rule must not place too much emphasis on information that is not accurately known. This is best achieved by avoiding any relative comparisons between the information in the spatial cohesion histogram bins. Implementing rules such as "if bin n is three times larger than bin m, then do not allow classification to bin m" are bound to fail since bins n and m could have a counts of 3 pixels and 1 pixel, out of a possible maximum of several thousand. All information in the histogram bins must be considered on a absolute basis against the maximum values that could occur under best conditions.

9.5 Tests on Non-Rectangular Mosaic

The same mosaic has been used in all the classification and segmentation experiments described above. This design of this mosaic, described in Section 4.2, is based on rectangular regions with vertical and horizontal class boundaries. The classification and segmentation processes are also based on rectangular regions. The features are generated with rectangular LAOs that are aligned with the rectangular regions in the mosaic. The cohesion measurement described in Chapter 8 is also based the classification results in a rectangular region. This does not necessarily mean that the rectangular regions in the mosaic are the optimum shape for the feature generation and cohesion technique. However, it does mean that the results using this mosaic do not give any indication as to how

Classification	31×31	15×15	31×31 + 15×15
	68.1	69.9	68.5
Non-hierarchical Segmentation	31×31	15×15	31×31 + 15×15
Method 0	68.1	69.9	68.5
Method 1	70.3	73.8	70.3
Method 2	68.9	73.4	69.1
Method 3	69.6	72.0	69.9
Method 4	68.6	71.5	68.9
Method 5	68.6	72.4	68.9
Hierarchical Segmentation	Sphere	Plane	Cylinder
Method 0	72.7	73.6	71.4
Method 1	75.1	75.5	74.6
Method 2	76.2	75.5	74.6
Method 3	72.7	74.2	71.6
Method 4	72.9	74.0	71.6
Method 5	73.0	74.2	71.8

Table 9.7: Results of classification and segmentation of texture mosaic (percentage of correctly assigned pixels)

the same methods work on non-rectangular regions or regions with non-vertical and non-horizontal boundaries as are found in more natural images.

To test the effect of non-horizontal and non-vertical texture boundaries, a second mosaic is used that has many of the properties not found in the first mosaic. The new mosaic is shown in Fig. 9.18a with the maps of the texture regions shown in Figs. 9.18b and 9.18c. This mosaic contains the same eight Brodatz textures that were present in the first mosaic (Table 4.1). The textures are present in the image in approximately equal proportions. The mosaic consists of three basic regions. The upper-left portion contains three textures in an arrangement similar to that shown in Fig. 9.2 where two texture are converging along curved paths. The upper-right portion of the new mosaic contains regions with non-vertical and non-horizontal boundaries, both straight and slightly curved. The bottom half of the image is made up of the eight textures in irregularly shaped regions of approximately equal size.

As described in Section 4.2, the textures used in the original mosaic were obtained by extracting 128×128 pixels regions from 512×512 pixels scanned images, and the *a priori* statistics were based only on the data in these regions. For generating the new mosaic, the 512×512 scanned images were used and data was extracted from various parts of the images as dictated by the region map (Fig. 9.18b). For classifying and segmenting the new mosaic, the same set of estimated statistics are used as was used for the original mosaic. This implies that the statistics are not entirely accurate for the new mosaic since much of the new mosaic consists of pixels that were not included in the previous statistics calculation. To a certain degree, this is similar to what would happen with a

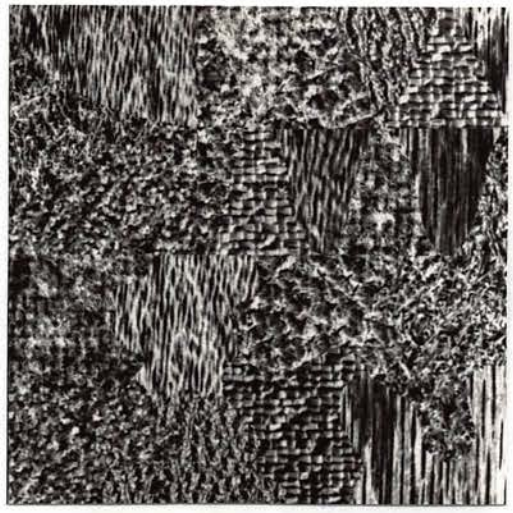
real image, where the classifier is based on available test data but must classify data that is slightly different.

The results for a full classification (no null-class) with 31×31 features, 15×15 features, and both 31×31 and 15×15 feature together are shown in Fig. 9.19. These results correspond to that shown in Fig. 5.1 for the original mosaic. The numerical results for the classification are listed in Table 9.8. Figures 9.21, 9.22, and 9.23 show the results of doing the classification with the null-class tests described in Chapter 7. These results correspond to those shown in Figs 7.2, 7.4, and 7.6 for the original mosaic.

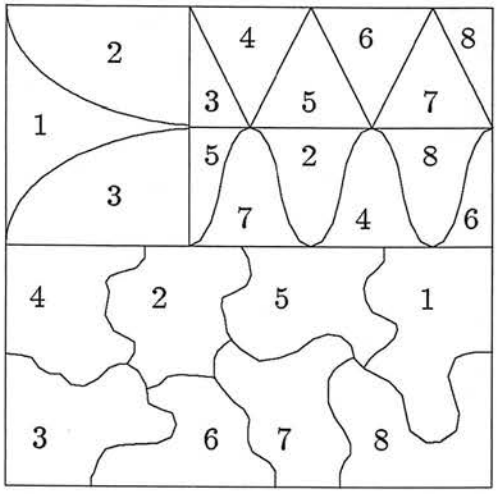
The cohesion histograms for the new mosaic are calculated the same way as with the first mosaic. The cohesion results are used to calculate the same set of final images that were shown above in this chapter for the first mosaic. These represent all combinations of the three null-class methods and the six decision methods for both the hierarchical and non-hierarchical segmentation. The numerical results are listed in Table 9.8. A sampling of some of the final images processed by the hierarchical segmentation methods are shown in Figures 9.24

The results with the non-rectangular mosaic lead to the conclusion that the quality of the segmentation is not greatly affected by the lack of vertical and horizontal texture boundaries. An unexpected result is that there appears to be little correlation between the how well the segmentation process performs in certain areas and how well a human observer performs when trying to find the same texture boundaries. Some of the boundaries that the process has the most trouble with are easy for a human to find, and vice-versa

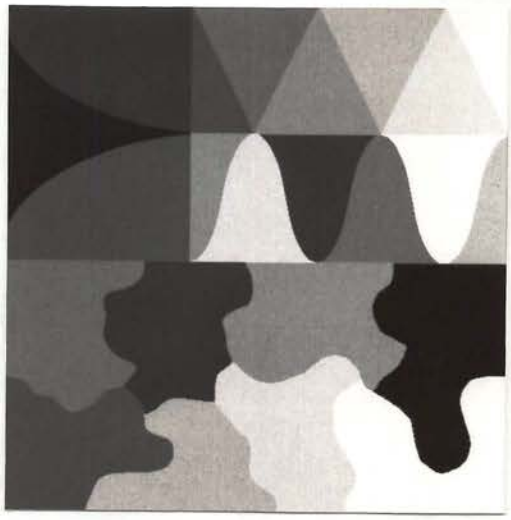
All of the methods resulted in very poor performance in selecting the correct class for the middle region of the three in the upper left corner. Note that



(a) Texture mosaic image



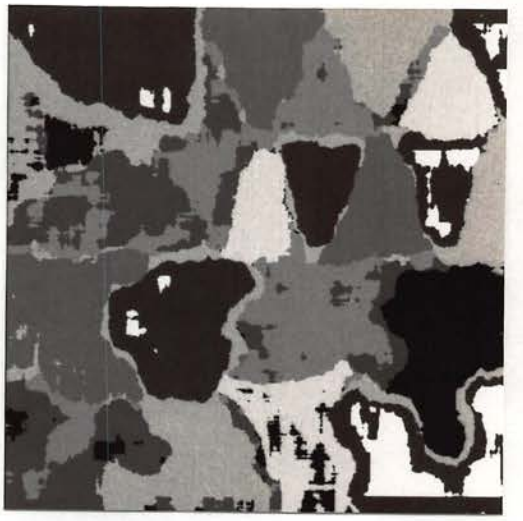
(b) Numerical texture map



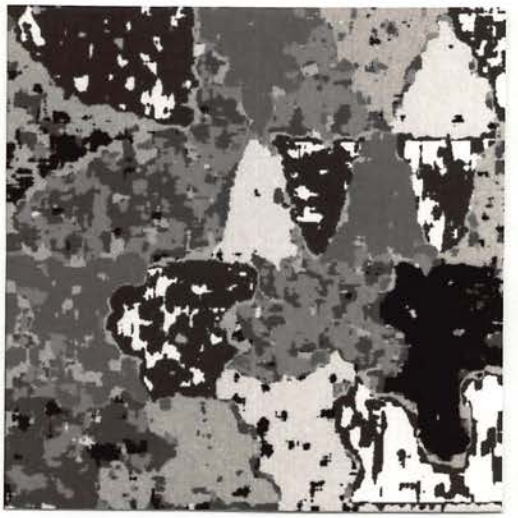
(c) Gray-level texture map

Figure 9.18: Nonrectangular texture mosaic image and reference maps

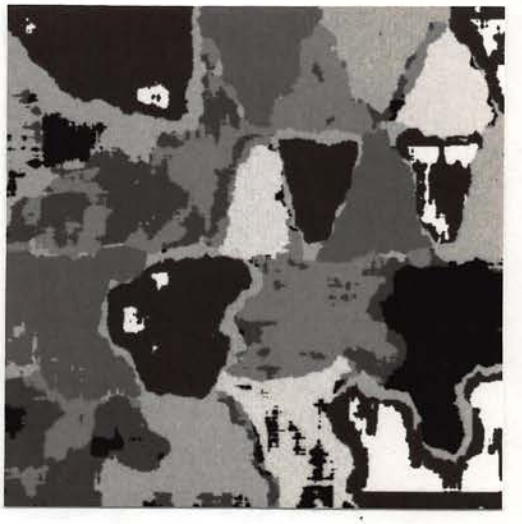
(a) Gray-level texture map



(b) Classification results – 31×31 features



(c) Classification results – 15×15 features



(d) Classification results – 31×31 and 15×15 features

Figure 9.19: Classification results for mosaic image with non-rectangular regions

Figure 9.20: Gray-level texture map

Figure 9.21: Classification with hypersphere null-class

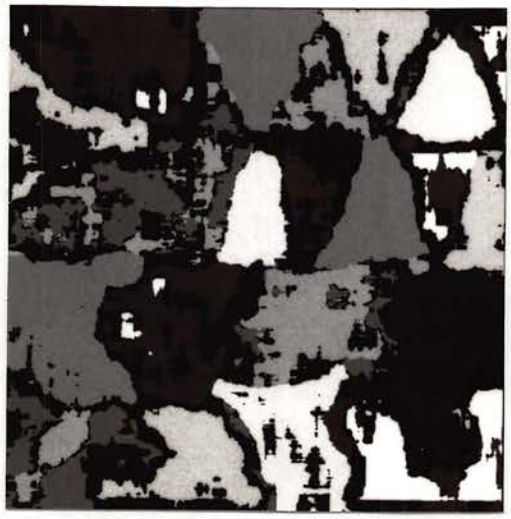
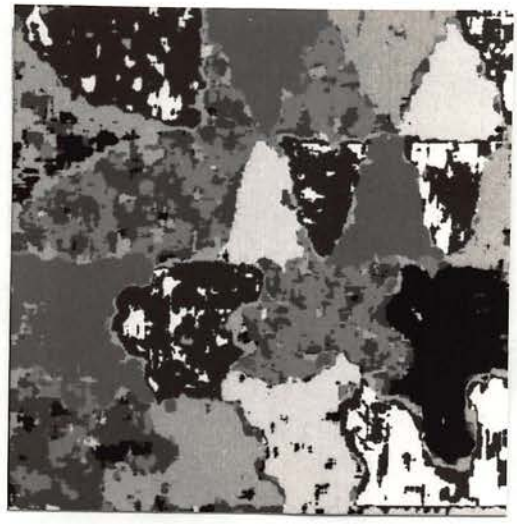
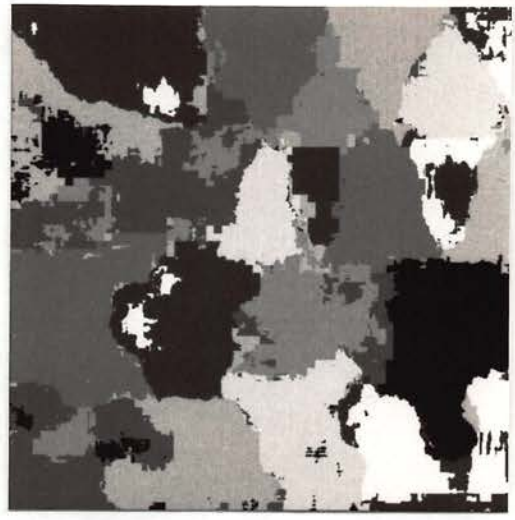


Figure 9.22: Classification with hyperplane null-class

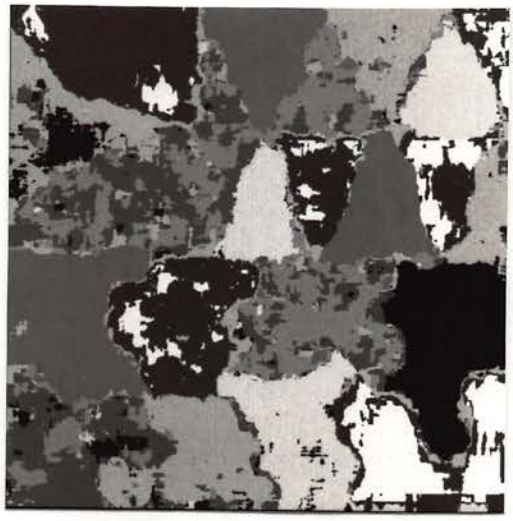
Figure 9.23: Classification with hypercylinder null-class



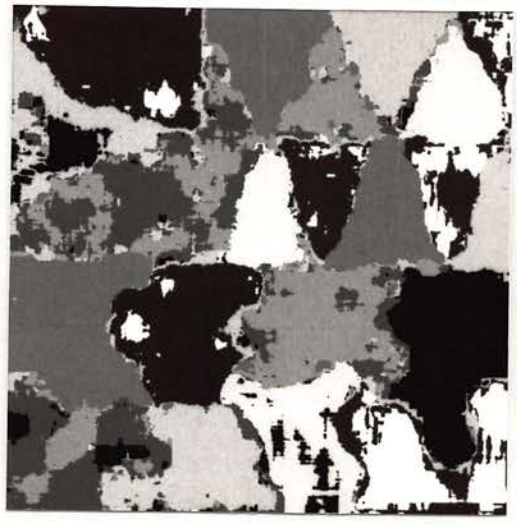
(a) Hypersphere – Method 0



(b) Hypercylinder – Method 2



(c) Hypercylinder – Method 4



(d) Hyperplane – Method 5

Figure 9.24: Non-rectangular mosaic segmentation results

Classification	31×31	15×15	31×31 + 15×15
	70.7	65.7	70.1
Non-hierarchical Segmentation	31×31	15×15	31×31 + 15×15
Method 0	70.7	65.7	70.1
Method 1	72.5	71.4	72.0
Method 2	71.6	70.7	71.2
Method 3	71.9	68.4	71.1
Method 4	71.2	68.3	70.8
Method 5	71.4	69.4	71.0
Hierarchical Segmentation	Sphere	Plane	Cylinder
Method 0	68.8	72.0	70.3
Method 1	75.3	76.0	75.2
Method 2	75.0	76.3	75.7
Method 3	68.8	72.4	70.4
Method 4	69.2	72.7	70.7
Method 5	69.3	72.9	70.9

Table 9.8: Results of classification and segmentation of non-rectangular texture mosaic (percentage of correctly assigned pixels)

the boundary between the middle and top region was determined relatively accurately, but that the middle class was selected incorrectly on the lower side of the boundary. Problems also exist at the edge of the image, most noticeably along the right side of the lower edge. This is also present in the first mosaic but not to this extent. The computation of the features at the edge of the image is done somewhat differently due the the lack of image data and this mosaic appears to be very sensitive to the situation.

When comparing the overall percentage of correctly assigned points, the results for the second mosaic are not significantly different from those obtained with the first mosaic. However, this is somewhat misleading since the regions in the second mosaic are generally larger than those in the first mosaic and this should improve the overall accuracy. Since the segmentation accuracy in this mosaic is not significantly improved over that achieved with the first mosaic, this implies that the combination of the non-rectangular boundaries and the non-optimum statistics are slightly detrimental to the performance of the segmentation.

Chapter 10

Summary and Conclusions

10.1 Summary

In this dissertation a method for segmenting images on the basis of texture was developed that combines the results of texture classifications done at more than one resolution. The multiple resolution classification data is combined in a hierarchical manner that achieves a better performance than is possible with either single resolution features or with nonhierarchical, multiple resolution methods. A major component of the segmentation is the detection of feature points that are affected by the nonstationarity of features generated from a mixture of texture classes.

Chapter 2 presented a overview of the texture segmentation problem. Image texture segmentation is broken down into individual processes of feature generation, classification, and segmentation. A review of past work in these three areas described several types of texture features that have been used in the past. Classification algorithms were briefly categorized and compared. A significant

amount of work has been done by others on segmenting images based on gray level information. The types of algorithms used are discussed including the region-merging and region-splitting methods. Multiple resolution approaches such as the those based on image pyramids are discussed in more detail.

The motivation behind the use of multiple resolution operator is described in Chapter 3. The size of the local area operator (LAO) used to generate the features has a significant effect on the way the resulting features perform in different parts of the image. Large operator work better away from texture boundaries while small operators are preferred when near a boundary. The manner in which the multiple resolution features can be combined in a hierarchical manner is described.

The features to be used in the classification and segmentation are described in Chapter 4. The features are based on the "texture energy measures" developed by Laws [22]. These features essentially measure the energy in the output of a matched filtering operation in a local area around the pixel. The filters are designed to be sensitive to the types of intensity variations commonly found in image texture.

Chapter 5 discusses the pattern classification algorithm used to assign the image pixels to a texture class. The distribution of the features is assumed to be Gaussian and the image pixels are classified using a Bayes classifier based on first and second order moments estimated from the image data. Results are shown and compared for classifications using two sizes of operators. In addition, results are shown for a classification of the test image using features at both resolutions simultaneously in a non-hierarchical manner. The use of both sizes of features

at the same time does not significantly improve the classification performance over that achieved with a single size operator.

The cause of the problem in using large operator near a texture boundary can be traced to the nonstationarity of the class statistics in the boundary area. Chapter 6 examines the nature of this problem. The larger the LAO, the more likely that it encompasses data from more than one class, resulting in features based on a mixture of the statistics from the classes present. An expression for the the probability density of the mixture is derived. An example is shown indicating a good agreement between the model of the mixture statistics and those measured in the actual data.

Chapter 7 presents proposed methods for dealing with the problem of the mixture densities. The methods are based on a common approach of using the *a priori* statistics of the class data to develop a test for detecting the presence of feature points that are likely to be the result of a mixing of the classes. The test is incorporated into a standard Bayes nearest-mean classification algorithm by assigning probable mixture points to a null-class for later classification. Three tests are examined: hypersphere, hyperplane, and hypercylinder. Each test implements a different shape of null-class in the feature space. Pictorial and numerical results are shown of classifying the test data with each method. The computational complexity of each methods, both for the initial analysis of the statistics and for the classification of each pixel, is discussed and compared.

Spatial information about the distribution of the texture classes in the area around each pixel is necessary to do a segmentation of the data. The manner in which the spatial information is obtained is described in Chapter 8. A measure of the similarity of a pixel with the neighboring pixels called "cohesion" is defined.

The spatial information about the cohesion of a pixel with the neighbors in multiple classes is calculated and stored in a vector of cohesion values called a cohesion histogram.

Chapter 9 describes the technique used to determine the final segmentation of the textures. The results from the classifications at two resolution sizes and the spatial information in the cohesion histogram are combined to determine the final choice for a texture class assignment. Six hierarchical methods for determining the final segmentation are described. The methods differ in the manner in which the spatial information is interpreted and used to control the class selection.

10.2 Conclusions

The results shown in the previous chapters have demonstrated that using multiple resolution operators in hierarchical manner can improve the performance of the segmentation. The numerical improvement in the segmentation accuracy is slight in many cases but it is consistent. The improvement in the visual appearance of the final images are generally noticeable to a human observer.

An accurate method for detecting mixture points is the key component of the hierarchical process. The three null-class methods described in Chapter 7 all provide an acceptable level of performance. All three have demonstrated the capability to increase the segmentation accuracy when used in the hierarchical process. The hypersphere, hyperplane, and hypercylinder methods are mostly ad-hoc with little theoretical basis. It is possible that better results can be achieved with a test that is more closely customized to the shape of the optimum

null-class. However it appears that any attempt to implement such a test would significantly increase the computational load for classifying each pixel even more. It seems unlikely that a more complicated test would improve the segmentation enough to warrant the cost. The current set of test are suggested as a trade-off that has acceptable performance, both in performance and speed.

The increases in computational cost in implementing the various null-class methods and the decision rules described in Chapter 9 are significant. For many applications of image segmentation, it is questionable as to whether the performance improvement is worth the extra computational cost. In circumstances where an optimum feature set can be carefully selected, it is likely that a greater improvement at a lower computational cost can be achieved by using a better feature set. However, there are many real world applications of image segmentation where the selection of the optimum feature set is either difficult or impossible. In these situations, the extra computational cost of implementing the techniques described above may be justified.

One area that could be investigated further is the development of a more sophisticated manner of measuring the cohesion in the local areas. The method described in Chapter 8 is somewhat simple-minded and could almost assuredly be improved upon. The same is probably true for the decision rules presented in Chapter 9. Most of these rules are not based on any firm theory but evolutionary changes over previous rules in an attempt to improve the segmentation performance.

References

[1] D. H. Ballard and C. M. Brown, *Computer Vision.* Prentice Hall, 1982.

[2] R. C. Gonzalez and P. Wintz, *Digital Image Processing.* Addison-Wesley, 1977.

[3] W. K. Pratt, *Digital Image Processing.* Wiley-Interscience, 1978.

[4] A. Rosenfeld and A. C. Kak, *Digital Picture Processing, Volume 2.* Academic Press, 1982.

[5] H. C. Andrews, *Introduction to Mathematical Techniques in Pattern Recognition.* Wiley-Interscience, 1972.

[6] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, pp. 786–804, May 1979.

[7] J. Bescos and T. C. Strand, "Optical pseudocolor encoding of spatial frequency information," *Applied Optics*, vol. 17, pp. 2524–2531, Aug. 1978.

[8] G. Lendaris and G. Stanley, "Diffraction pattern sampling for automatic pattern recognition," *Proceedings of the IEEE*, vol. 58, pp. 198–216, Feb. 1970.

[9] H. Harmuth, *Sequency Theory.* Academic Press, 1977.

[10] L. Kirvida, "Textures measurements for automatic classification of imagery," *IEEE Transactions on Electomagnetic Compatability*, vol. 18, pp. 38–42, Feb. 1976.

[11] W. K. Pratt, W. Chen, and L. R. Welch, "Slant transform image coding," *IEEE Transactions on Communications*, vol. 22, pp. 1075–1093, Aug. 1974.

[12] J. O. Eklundh, "On the use of Fourier phase features for texture discrimination," *Computer Graphics and Image Processing*, pp. 199–201, 1979.

[13] R. M. Haralick, K. Shanmugan, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 610–621, June 1973.

[14] B. Julesz, "Visual pattern discrimination," *IRE Transactions on Information Theory*, vol. 8, pp. 84–92, Feb. 1962.

[15] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 269–285, Apr. 1976.

[16] S. W. Zucker and D. Terzopolous, "Finding structure in co-occurrence matrices for texture analysis," *Computer Graphics and Image Processing*, vol. 12, pp. 286–308, 1980.

[17] L. Davis, S. Johns, and J. K. Aggarwal, "Texture analysis using generalized cooccurrence matrices," in *Proceedings, IEEE Conference on Pattern Recognition and Image Processing*, pp. 313–318, 1978.

[18] D. Terzopolous and S. W. Zucker, "Texture discrimination using intensity and edge co-occurrences," in *Proceedings, 5th International Conference on Pattern Recognition*, pp. 565–567, 1980.

[19] F. Vilnrotter, *Structural Analysis of Natural Textures.* PhD thesis, University of Southern California, 1981.

[20] F. M. Vilnrotter, R. Nevatia, and K. E. Price, "Structural analysis of natural textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 76–89, Jan. 1986.

[21] K. I. Laws, "Rapid texture identification," in *Image Processing for Missile Guidance (Vol. 238)*, pp. 376–380, Society of Photo-Optical Instrumentation Engineers, 1980.

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

[23] K. I. Laws, "Texture energy measures," in *Proceedings, Image Understanding Workshop*, pp. 47–51, Nov. 1979.

[24] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* Wiley-Interscience, 1973.

[25] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Addison-Wesley, 1974.

[26] G. H. Ball and D. J. Hall, "A clustering technique for summarizing multivariate data," *Behavioral Science*, vol. 12, pp. 153–155, March 1967.

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

[28] K. S. Fu and J. K. Mui, "A survey on image segmentation," *Pattern Recognition*, vol. 13, pp. 3–16, 1981.

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

[30] M. H. Hueckel, "An operator which locates edges in digitized pictures," *Journal of the Association for Computing Machinery*, vol. 18, pp. 113–125, Jan. 1971.

[31] L. G. Roberts, "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, (J. T. Tippet, ed.), pp. 159–197, MIT Press, 1965.

[32] C. R. Brice and C. L. Fennema, "Scene analysis using regions," *Artifical Intelligence*, vol. 1, pp. 205–226, 1970.

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

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

[35] S. L. Horowitz and T. Pavlidis, "Picture segmentation by a directed split-and-merge procedure," in *Proceedings, 2nd International Joint Conference on Pattern Recognition*, pp. 424–433, 1974.

[36] S. L. Horowitz and T. Pavlidis, "Picture segmentation by a tree traversal algorithm," *Journal of the Association for Computing Machinery*, vol. 23, pp. 368–388, Apr. 1976.

[37] S. G. Carlton and O. R. Mitchell, "Image segmentation using a local extrema texture measure," *Pattern Recognition*, vol. 10, pp. 205–210, 1978.

[38] S. G. Carlton and O. R. Mitchell, "Image segmentation using texture and gray level," in *Proceedings, IEEE Conference on Pattern Recognition and Image Processing*, pp. 387–391, 1977.

[39] S. Tanimoto and T. Pavlidis, "A heirarchical data structure for picture processing," *Computer Graphics and Image Processing*, vol. 4, pp. 104–119, 1975.

[40] T. Pavlidis and S. Tanimoto, "Texture identification by a directed split-and-merge procedure," in *Proceedings, Conference on Computer Graphics, Pattern Recognition, and Data Structures*, May 1975.

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

[42] P. Burt, T. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 802–809, Dec. 1981.

[43] M. Pietikäinen and A. Rosenfeld, "Image segmentation by texture using pyramid node linking," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 822–824, Dec. 1981.

[44] W. B. Thompson, "Textural resolution," in *Proceedings, 3rd International Joint Conference on Pattern Recognition*, pp. 283–286, 1976.

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

[46] D. T. Kuan, A. A. Sawchuk, T. C. Strand, and P. Chavel, "Adaptive noise smoothing filter for images with signal-dependent noise," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, no. 2, pp. 165–177, 1985.

[47] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* Dover Publications, 1966.

[48] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 1965.

[49] R. Bellman, R. Kalaba, and L. Zadeh, "Abstraction and pattern classification," *Journal of Mathematical Analysis and Applications*, vol. 13, pp. 1–7, 1966.

[50] S. Tamura, S. Higuchi, and K. Tanaka, "Pattern classification based on fuzzy relations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-1, pp. 61–66, Jan. 1971.

[51] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[52] T. S. Ferguson, *Mathematical Statistics.* Academic Press, 1967.

[53] R. M. Gagliardi, *Introduction to Communications Engineering.* Wiley-Interscience, 1978.

[54] J. W. Goodman, *Statistical Optics.* Wiley-Interscience, 1984.

[55] R. Keller, A. T. Zavodny, and T. A. King, "Pre-similarity-detection (psd) image processing," in *Proceedings, 5th International Conference on Pattern Recognition*, pp. 937–942, 1980.