Segmentation of Ultrasound Images Using Texture Discrimination

A Thesis Submitted to the College of
Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in the
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan

by

Russell E. Muzzolini

October, 1991

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 0W0

Segmentation of Ultrasound Images Using Texture Discrimination

Candidate: Russell E. Muzzolini

Supervisor: Herb Yang

Co-supervisor: Roger Pierson

Doctor of Philosophy Thesis

Presented to the College of Graduate Studies and Research

University of Saskatchewan

October, 1991

ABSTRACT

In medical imaging applications such as ultrasound, it is often required to segment the image to determine its meaningful parts. Segmentation may be simple differentiation between the background and the object, or more complexly, distinguishing different objects or parts of objects within each image.

There are many methods available for segmenting an image. The number of these methods applicable to segmenting 2D ultrasound images is significantly reduced because of the presence of noise. It was found experimentally that textures detectable in ultrasound images are more appropriate features for segmentation than simple edges. Detection of edges is profoundly affected by the presence of noise.

A segmentation algorithm based on texture discrimination is presented in this thesis. This algorithm uses a constrained optimization approach to produce a segmentation. Because this algorithm can not deal with varying texture region sizes effectively a multiresolution algorithm is developed. This algorithm can be used to efficiently produce a segmentation of the image. Both methods are statistically evaluated to determine their effectiveness in segmenting noisy ultrasound images. Based on a series of these images, it is shown how the segmentations obtained from the multiresolution algorithm can be used to produce a 3D representation of an ovarian follicle.

ACKNOWLEDGEMENTS

I thank Herb Yang for the time and effort he devoted to this thesis. Some people questioned my attitude but Herb always gave me the benefit of the doubt. His guidance pointed me in the right directions; his instruction helped fill in my intellectual gaps; his personal warmth, caring, genuine interest, and honest criticism motivated me to deliver my best work. The close interaction that I experienced working with Herb brought about significant changes is my life; both academically and personally. For this, I will always be greatful.

I thank Roger Pierson for his instruction and extremely helpful comments in the many pages of research that I dropped on his desk. Although we didn't interact on a regular basis, his personal experiences provided insight into realms of biology of which I was totally unaware.

I thank all the professors, staff and graduate students in the Department of Computational Science for their collegial interaction. I especially thank committee members, Eric Neufeld and John Cooke, and the external examiner, Scott Lozanoff, for their close review of my thesis.

I thank my friends and family for their support. My parents, Ennio and Janet Muzzolini have always been generous with their love and time. They have always supported my endeavors; they taught me how to work hard and how to make good decisions and they offered advice whenever it was asked for. I also thank my sister Tracey and my little brother Blair. Even though Blair is my younger brother he offered me things that a big brother could provide.

Most importantly, I thank my wife Leslie for her love, trust and support. The years we have spent together have had their share of rough moments, but together we managed to make it through them all. Leslie's encouragement and support has been needed at many times and has been deeply appreciated. To remind Leslie, and all those who read this thesis, how I feel about her, I leave the following:

"But most of all, for cryin' out loud, you know I love you." - MEATLOAF.

Dan, I am not the last one.

Russ

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In realizing a 3D representation of an object based on a series of 2D cross sections, it is necessary to ensure that the 3D representation provides an accurate description of the object. This is most important in medical imaging applications where potentially critical decisions are made based on the rendered view of the brain, heart or other human organs.

The accuracy of the 3D representation depends on how well each 2D cross section represents a physical slice of the object under examination, as well as how accurately the series of cross sections are aligned with respect to rotation and translation. These lower level considerations provide a basis upon which the 3D framework is built.

In diagnostic ultrasonography, the alignment of the cross sections poses a problem. Images are subject to rotation and translation because the ultrasound probe is maneuvered manually. As well, there is a large amount of noise present in the ultrasound images. An accurate 3D representation may be produced by the following two steps:

1. segmenting each ultrasound image to determine where the object(s) exists without being affected by noise and

2. producing a 3D model, using the segmented images, which accounts for translation and rotation of the images in the serial scan.

This thesis focuses on the first step, i.e., the segmentation of ultrasound images.

It explains why texture segmentation is more appropriate than conventional edge detection techniques for segmenting ultrasound images.

1.1 Thesis Organization

The organization of this thesis is described in the following paragraphs. In Chapter 2, background information on ultrasound imaging is provided. An overview of the ultrasound technique is described as well as some of the difficulties encountered when trying to construct an accurate 3D model from a series of diagnostic, cross sectional ultrasound images.

Chapter 3 provides a survey of image segmentation techniques. Characteristics of edge detection are identified in the examination of approaches using edge detection. Since the edge detection approaches are susceptible to noise in images, texture discrimination approaches are also examined. The advantages of a multiresolution approach in segmenting an image based on texture are discussed.

Theoretical background for two segmentation algorithms, the Constrained Optimization (CO) algorithm [12] and the proposed Multiresolution Texture Segmentation (MTS) algorithm, is provided in Chapter 4. Both simulated annealing and constrained optimization are presented. The CO algorithm is described and its weaknesses are identified. The MTS algorithm, which was developed to overcome the weakness in the CO algorithm, is also described.

The use of texture characterization in the MTS algorithm is discussed in Chapter 5. 32 features in the spatial and frequency domains are described. The experiments performed to evaluate the ability of the features to characterize textures sampled from a series of diagnostic ultrasound images are presented. It is also shown how only the best feature at each resolution is used to identify the type of texture present in a block of pixels during the segmentation process.

Experimental evaluation of the MTS algorithm is presented in Chapter 6. As well, comparisons between the MTS and CO algorithms are made based on experimentation with both synthetic and real images.

Chapter 7 describes how the segmentations produced by the MTS algorithm are used in creating a 3D model of an ovarian follicle. Chapter 8 discusses the contributions made by this thesis. Conclusions and future work resulting from this thesis are also discussed.

Chapter 2

Ultrasound

Ultrasound imaging provides a non destructive means of examining objects in the human body that can not be effectively or safely examined by other imaging techniques such as magnetic resonance imaging (MRI) [36, 49] or computed tomography (CT) scans [49, 31, 15, 53, 2] because the object is composed of soft tissue or when the object being examined is a fetus or an ovarian follicle (see Section 2.2). The following Sections discuss aspects of ultrasound imaging and its application in the area of Obstetrics and Gynaecology.

2.1 The Ultrasound Technique

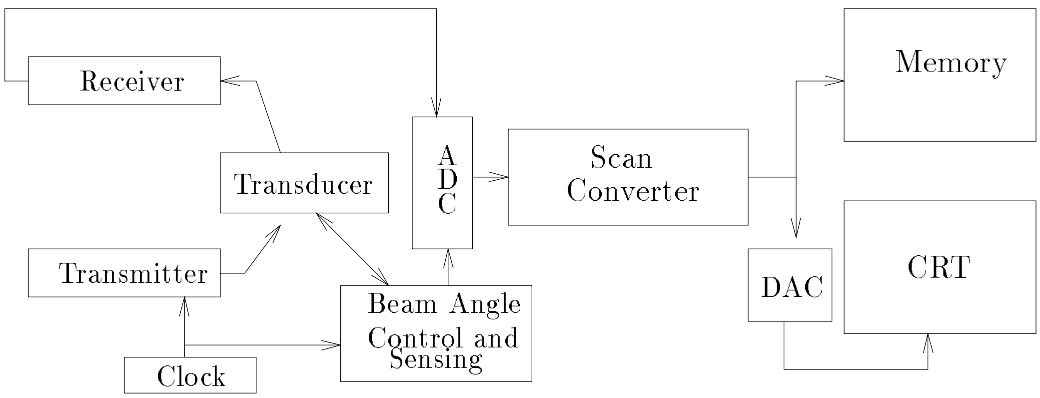


Figure 2.1: Block diagram of a typical ultrasound scanner.

The following discussion refers to the use of a digital scanner in the ultrasound technique [11, 24, 47] but can be extended to the use of older analog scanners with minor modifications.

A cross sectional view of an object can be produced by transmitting an ultrasound wave through the object. Since the speed at which the sound wave travels through soft tissue is relatively constant, the time from which the sound wave was transmitted to the time it returns is proportional to the depth of the tissue which reflected it. Thus, the longer it takes for a sound wave to be returned, the greater the depth of the tissue reflecting the sound wave.

Figure 2.1 illustrates a basic ultrasound scanner. The directed arrows indicate the flow of information among the components in the scanner. Ultrasound pulses are transmitted and received by the *transducer*. The transducer emits a constant rate of ultrasound pulses under the control of a clock and the *beam angle control and sensing circuitry*. The beam angle control circuitry tells the transducer where to aim the beam of ultrasound pulses. The sensing portion of this circuitry determines the angle at which the beam is directed and passes this information on to the *scan converter* [32]. Depth signals received by the transducer are also passed on to the scan converter. Since the depth signals are analog they must pass through an analog to digital converter before reaching the scan converter. The scan converter uses the information it gets from the beam angle control and sensing circuitry to calculate the appropriate pixel address for the intensity value based on the transducer depth signal. Once the address calculations are performed, the intensity values can be displayed on a video display (CRT) and stored in memory for further processing.

Ultrasound scanners in medical imaging typically produce a pulse with a frequency in the range of 3.5 to 10 MHz. Better resolution can be achieved with a higher frequency, but the pulse can not penetrate as deeply into the tissue. Smaller structures which are closer to the surface are usually scanned with a high frequency while larger, deeper structures are scanned at a lower frequency. The ultrasound pulse rate is limited by the speed at which sound travels through tissue. Pulse rates of 100Hz to 1000's Hz are physically possible.

The scan converter controls placement of each ultrasound echo to its appropriate pixel location in memory (or video display). Most scan converters locate pixel intensities on a 512x512 grid. The resolution of the pixel intensity ranges from 5 to 8 bits. In real-time sector scanning, the transducer oscillates to move the beam rapidly back and forth. Each sweep produces a 512x512 image (A single point penetrates the tissue and produces one row of the cross sectional image). As the transducer is moved along the object, a series of serial cross sections or images is produced.

2.2 Biological Aspects

Ultrasound imaging in obstetrics and gynaecology has led to some interesting applications. The application area of interest in this thesis is the study of ovarian follicles [18]. It is hoped that research in this area will reveal new information about the reproductive biology of women. For example, ultrasound imaging of ovarian follicles could capture changes in the shape of the follicles over time. By recording these changes up to the point of ovulation, it may be possible to characterize different shape patterns of the follicle. These patterns could then be used to predict when ovulation of a follicle occurs.

Another use for ultrasound imaging of ovarian follicles is reconstructing a 3D representation of a follicle, based on a series of serial cross sections. By accurately segmenting each follicular image in the series of cross sections, a 3D model can be created by "stacking" the segmented 2D images on top of each other. The stacking procedure would have to account for such things as rotation and translation of the segmented images, as well as perform some type of interpolation between successive images. Once a 3D model has been obtained, rendering software can be used to look at different perspectives of the follicle. Surface abnormalities, impossible to see in single cross sections, can be detected by projecting different views of the follicle. Statistical information such as surface area and volume can also be estimated, based on the 3D model.

An ovarian follicle typically has an ovular shape in the ultrasound image. It has

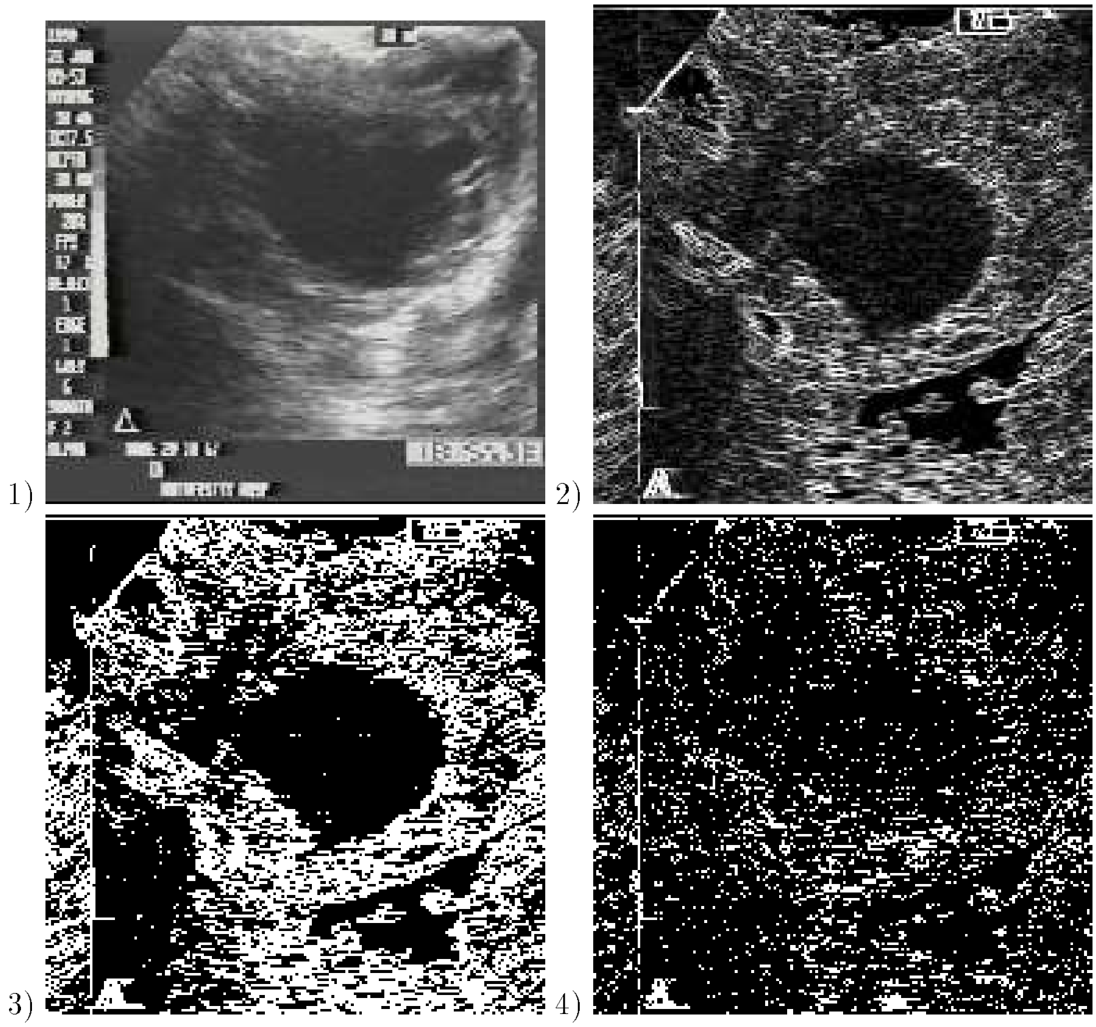
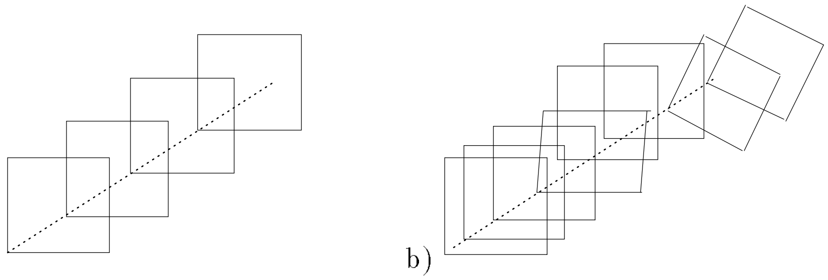


Figure 2.2: (1) Original image and edge map using (2) Prewitt, (3) Sobel, (4) Laplacian operators.

both a large inner area surrounded by a thin outer layer. The borders of the follicle appear fuzzy at best. Interpretation of the shape of a follicle, until now, has been left to the physician's subjective judgment. Accounting for this and the fuzziness of the follicle's border, segmentation of a follicular ultrasound image will be subject to a certain amount of error. The goal of this thesis is to reduce the error to a tolerable level. This will result in a segmentation which accurately reflects the true shape of the follicle, based on an ultrasound image.

2.3 Difficulties With Ultrasound



a) b)

Figure 2.3: Two serial cross section scans a) ideal b) alignment error

A number of difficulties arise when trying to construct an accurate 3D model from a series of cross sectional ultrasound images. First, the noise present in an ultrasound image makes it difficult to accurately distinguish the borders between objects and the background. As a result, edge detection algorithms can not be used to accurately locate objects in an ultrasound image. More generally, any segmentation algorithm that is susceptible to noise is not useful for segmenting ultrasound images. Figure 2.2 shows a follicular ultrasound image and the resulting edge maps produced with the Prewitt, Sobel, and Laplacian edge detectors. Each edge map was created by convolving the follicular ultrasound image with the appropriate operator and then thresholding the result to extract the edge information. It is obvious from the figure that the edge maps produce too many extraneous edges. Although it is possible for the average person to see a blob (follicle) in the middle of the edge maps, it is impossible for a computer to accurately determine the location of the object from these maps.

Another difficulty with ultrasound arises from the fact that an ultrasound scan, producing a series of cross sections, is performed manually. Therefore, a certain amount of alignment error is introduced. This alignment error is also due to the fact that the object being scanned moves as well. The resulting series will have cross sections which are rotated and/or translated along the scanning axis. Figure 2.3 illustrates these types of alignment errors as well as an ideally aligned serial scan.

The rotational alignment errors of slices (cross sections) 7 and 8 in b) are obvious. There are different types of translational alignment errors. The ones illustrated in b) are:

1. Some slices are off the scanning axis (slices 5 and 6)

2. The inter-slice distances along the scanning axis vary (slices 1-4)

3. Slice 4 is not perpendicular to the scanning plane

A third difficulty with ultrasound arises from the fact that until this point, most, if not all interpretation of ultrasound images has been left to the discretion of the viewer. Visual inspection of ultrasound images is certainly prone to error. Because of the noise present in the images, it is difficult to accurately locate the borders of an object. As a result, the physician can recognize the general shape of the object but can not precisely locate its borders. It is this precision that is necessary when producing a 3D model based on ultrasound images. Any inaccuracies incurred in the initial segmentation process will lead to larger error when extrapolating information from the 3D model. The work in this thesis addresses this very important problem, namely, the segmentation of ultrasound images.

Chapter 3

Image Segmentation Techniques

The purpose of image segmentation is twofold. First, the large amount of information present in images (especially natural images) is reduced in the segmentation process. In order for automated processes to understand images, it is necessary to reduce the amount of data by retaining only that information which is necessary in interpreting the images. This could mean retaining only the outline of parts in an assembly line or finding the size of a region with a certain texture to determine defects. Secondly, some images do not contain enough information for automated processes to identify objects or regions. This is especially true when noise in an image can corrupt the boundary between objects or the outline of objects. It is the purpose of image segmentation to determine, as accurately as possible, where boundaries lie or which regions in the image are the same or different.

There is a wide variety of techniques available for segmenting images. One of the most common and studied approaches employs some method of edge detection in which the outline of regions is used to segment an image. Another common approach for segmentation involves examining the textures present in an image and grouping all regions together which have similar textures. This chapter discusses these basic approaches as well as some variants made to increase the speed and accuracy of the segmentation process.

3.1 Edge Detection

Edge detection approaches involve determining the outline of objects or the boundary between different regions in an image. Typically a measure such as the gradient, which is the slope of the intensity change, is taken between a pixel and its neighbors and then compared to a given value called a threshold. If the gradient is above the threshold, then an edge point can be placed at that pixel. Depending on the actual gradient of the edges in the image, the threshold value will determine whether or not an edge detection algorithm will identify the edges.

When every pixel in the image has been examined, the original image will have been transformed into a form containing information about the edges. A common way of representing this reduced form is in an edge map. An edge map applies an intensity to only those pixels which exceed the threshold. Usually, the brighter the intensity, the sharper is the edge.

There are a number of general issues to consider besides the setting of a threshold when performing image segmentation with edge detection. First, the accuracy of the placement of edges is highly dependent on the type of images being segmented. If an image is noisy then the edges will be placed with some amount of error. Extraneous edge points will cause edges to be extended beyond the outline of objects. Also, when there are smooth intensity changes in an image, the exact location of an edge is difficult to determine.

When it is difficult to determine the exact location of an edge in an image the detected edge may occupy thin regions both inside and outside the image edge. This will result in a thicker detected edge. Fine changes in an object's outline can also cause a detected edge to thicken. Since accurate placement of edges is required, it is desirable for edges to be as thin as possible and in so doing, retaining as much detail as possible.

Continuity of edges is necessary in the segmentation output. When a broken outline of the objects in an image is produced, the accuracy of the segmentation is reduced because the next level of processing has to connect the broken edges. These connections are made on the edge information; much less than in the original image.

The amount of noise in an image has a profound effect on the results of an edge detection approach. Broken and spurious edges are a direct effect of noise. As the amount of noise in an image increases, the effectiveness of edge detection techniques decrease to the point where they are incapable of producing an accurate segmentation. This is precisely the problem when these techniques are applied to ultrasound images.

3.1.1 Mask Convolutions

A very common technique used in edge detection involves the use of convolution of an image with a function (mask) [33, 23, 10, 9, 13, 41]. The purpose of this is to determine local measurements of the edge strength based on a pixel and its surrounding neighbors. A measure for each pixel in the image is obtained by centering the mask over top of the image at each pixel location. Since the mask is a small array of values, typically 3x3, 5x5, or 7x7 (any odd size mask is possible), multiplication of the pixel values and the mask values is performed and their sum provides a measure of the edge strength at the center pixel.

A number of issues have to be considered when performing mask convolutions to detect edges. The values that comprise the mask are of obvious importance. They will affect the edge strength attributed to each pixel. The values are chosen in such a way that the orientation of a line in the image may or may not affect the mask's ability to measure the edge strength. The Laplacian operator, for example, is a mask which is not affected by the orientation of a line. This is especially true when curved edges are being detected. The Sample Variance [9] and Laplacian-of-Gaussian [23] are operators which also detect edges without regard to orientation. Operators such as Roberts, Sobel, and Prewitt [33] respond better to edges or corners in a certain orientation but can not always detect all edges (or curves) with accuracy. Operators which compute direction derivatives of the intensity [10] also fall into this class

The mask size is another issue to consider. The larger the mask size the more statistically accurate the measure for each pixel becomes. But, less detail is detected as small changes in the direction of an edge will be averaged out by a large mask. As well, the time to compute the convolution of the image with the mask will increase significantly when the mask size is large.

Since all of the above masks are convolved with the absolute intensity values of an image they may not respond similarly to the same image under different illumination; a trait which is undesirable in applications such as robotics. One way to limit this problem is to modify convolution masks so that they are sensitive to contrast rather than absolute intensity differences. Johnson [26] has shown how conventional operators such as the Sobel operator can be modified to detect contrast. Admittedly the contrast based operators suffer a performance penalty as extra computation per pixel is required.

3.1.2 Template Matching

Another edge detection technique related to mask convolutions is template matching [48, 41, 13]. A small mask or template is applied to each pixel in the image. Instead of convolving the template, however, various templates are matched against each pixel center. The template with the best match indicates the type of edge, if any, that is at the pixel center. Templates may be designed to match corners, lines of any orientation, or curved lines. Because of this, the resulting edges are characteristically thinner and more continuous.

The effectiveness and practicality of template matching depends on the size of the template used. A small template does not require very many permutations but is susceptible to noise in the image. A large template can match the edges in the image accurately but requires a huge number of permutations in order to detect possible edge orientations.

A template matching approach was developed by Tan et al. [48]. This approach uses a cost function to evaluate how well each template matches an edge at a pixel center. The cost function assigns a weight to each template match based on such factors as edge thickness, edge continuity, edge length, and number of edge pixels.

The least cost template is chosen as the best current match for the pixel center. Templates are matched to randomly chosen pixel centers to find the best configuration of templates for the entire image. This process is iterated until no better configuration can be found, or some stopping criteria is met. Tan et al. found that it was computationally more efficient to step through the pixels in a raster scan fashion. Iteration proceeded until **k** scans were completed or no better configurations were produced after a certain minimum number of scans. Since no global information is maintained, a parallel implementation is quite possible.

Because this approach is based on a minimum cost function it is possible that it could get stuck in a local minimum and not produce the most appropriate segmentation. Tan et al. conjecture that their approach could be globally optimized through stochastic relaxation (see Section 4.2) to reduce the problem but would induce a higher computational cost.

3.1.3 Three Dimensional Extension

Edge detection does not have to be limited to 2D. It can be performed in 3D [19, 28] and even multidimensional space [50]. In dealing with edge detection in more than two dimensions, the voxel, rather than the pixel, becomes the basic unit of processing. In 2D, a pixel can be envisioned as a square. In three and multidimensions the voxel occupies a cubic volume. Planes, rather than edges, are to be detected. The goal of the segmentation process is to determine the boundary planes of different volumes in the image.

Potential applications for 3D edge detection can be found in the medical imaging domain. Useful information can be gained by examining various volumes in 3D¡ medical data. The usefulness of multidimensional edge detection in image processing has to be questioned, however. Interpreting the segmentation results would be difficult. Moreover, the usefulness of this information in some application is hard to justify.

3.2 Texture Discrimination

An alternative image segmentation technique to edge detection is texture discrimination. With texture discrimination, rather than trying find the edges in an image, it is attempted to characterize regions of an image based on measures of texture. It is still possible to produce an edge map of an image, as with edge detection, by creating a border between regions determined to have different textures. It is important to note, however, that the edge map is created by examining regions rather than edges.

3.2.1 Texture Features

A texture feature is used to characterize texture in an image. The result of the characterization is a measure of how well the texture can be identified by that feature. This measure is then used by a region partitioning algorithm or other texture discrimination algorithm to segment an image. Rao [38] developed a taxonomy for the description of different types of textures which could effectively classify a set of synthetic images (the Brodatz texture images). This taxonomy provides a standardized set of descriptors which can be used for qualitative descriptions of textures. Thus, a means is provided for which texture features can be evaluated according to a standard set of texture descriptors. Although it is not certain that these descriptors can be extended to real images, it is possible to achieve some basis of common comparisons among different texture features based on these synthetic texture descriptors.

A texture feature can be applied to an image in both the spatial or spatial/frequency domains. A spatial texture feature is one which samples the raw intensities in the image, evaluates the intensities with some sort of measure, and then produces a value of texture strength for the sampled intensities in the image. All computations are performed in the spatial domain. There are numerous texture features that can be applied to images in the spatial domain [8, 46, 16, 51]. Xiaohan and Baozong [54] proposed the Direction Measure (DM) based on the observation that in homogeneous textures the pixel intensities along any direction must change with some regularity. Their texture feature basically computes first, second, and third order directional derivatives in 8 directions. They also claim that it can be extended to be rotationally invariant. The effectiveness of the DM to identify texture was claimed to be around 95% although this number has to be taken with some caution as the method in which the results were obtained is sketchy at best. A number of questionable assumptions and statements also undermine the effectiveness of the DM as a texture feature.

Gotlieb and Kreyszig [16] use co-occurrence matrices as the basis for a statistical description of texture. Of most importance in their work was the development of an analytical model for the experimentation performed with their features. This was one of the few attempts at producing a solid foundation for experimental evaluation of texture features.

Unser and Eden [51] used non-linear operators to improve texture discrimination. They found that the combination of the non-linear operators $f(x) = |x|^\alpha$ and $g(x) = \log x$ applied to their set of original texture features improved the ability of the features to identify texture as well as the performance of the clustering algorithm used to group regions of the same texture.

A spatial/frequency texture feature [39, 29, 25] is obtained by transforming the raw intensities from the spatial domain to the frequency domain before the texture measure is computed. The transformation of the data from the spatial to the frequency domain is performed with the Fourier transform [4, 14]. Liu and Jernigan [29] believe it is advantageous to use spatial/frequency texture features when the image being segmented is noisy because the frequency domain is less sensitive to noise. They also believe that the power spectrum can provide more effective texture features. They defined 28 texture features based on the power spectrum $P(u, v)$ and the phase spectrum $\phi(u, v)$. Liu and Jernigan experimentally discovered that a subset of 4 features was optimal for identifying individual textures. They also discovered that the ability of the features to identify a texture dropped when the signal to noise ratio was decreased. Two important points to note from the Liu and Jernigan work are:

1. The method for determining the optimal texture features was performed in an experimentally controlled environment;

2. The test images were synthetic and selected from Brodatz.

Point 1 is a necessity for any evaluation of texture features (or for any segmentation algorithm in general). Without a precisely defined method for experimentation, a description of the experimental environment, and a list of assumptions involved in performing the experiment, the validity and accuracy of the texture features has to be questioned. It is necessary for other researchers to be able to reproduce the results that were originally claimed so that the results can be validated or disproven.

Point 2 indicates a trend in much of the research in image processing. The use of synthesized images is widespread in many experimental procedures. There is nothing wrong with this except that many claims, based on the results obtained from synthesized images, have been used to predict how well (never how poorly) a texture feature (or segmentation algorithm) will perform on any image. Often times the synthesized images are chosen so that the results look favorable. Other times images are chosen without any justification for why they were selected; as is the case with Liu and Jernigan. The "well known" Brodatz images have been used by many researchers in evaluating texture features, if not for any reason other than other researchers were using them. This is one of the main problems when synthesized images are used in experimentation.

3.2.2 Region Partitioning

The goal behind region partitioning is to segment an image into a number of homogeneous regions based on texture features. This segmentation does not produce an outline of each region, rather it identifies or labels the homogeneous regions. This is advantageous over edge detection because there is more information contained in the segmentation, that being the added knowledge of which (possibly distant) regions have the same texture and thus belong to the same homogeneous region.

Both supervised and unsupervised techniques have been proposed for region partitioning (image segmentation). In unsupervised image segmentation [37, 22], a general set of texture features are used to characterize any image. Raafat [37] claims that his set of "essentially non-parametric" texture features can identify homogeneous regions in any image. Once an initial partitioning of the image is obtained, a post processing step is performed which fine tunes the boundaries of the regions to produce a more accurate segmentation. The absence of parameters is characteristic of the unsupervised technique. The problem with this is that since images are so diverse it is very difficult to derive a set of texture features which will accurately detect texture differences in any image.

Supervised image segmentation is more appropriate for producing an accurate image segmentation. An algorithm using this technique typically has one or more parameters which have to be adjusted according to the type of image being segmented. These parameters can be adjusted manually on a trial and error basis or they can be adjusted automatically by "training" the algorithm with a set of sample images. Various measures can be obtained from the training set to either sensitize or desensitize the texture features to the type of textures in the images. The Constrained Optimization (CO) algorithm (see Section 4.3) and the Multiresolution Texture Segmentation (MTS) algorithm (see Section 4.4) are examples which perform supervised image segmentation. There is a set of parameters which must be adjusted according to the type of image being segmented.

3.2.3 Multiresolution Approaches

Region growing [22, 37] is one way of partitioning an image into homogeneous regions. Initial points are selected and neighboring blocks are added to the points if they are determined to have the same texture. In this way, a region "grows" from a point to a larger homogeneous region. An alternative to region growing is to consider an image as consisting of a number of blocks. Each of these blocks can be characterized by the texture within it and can be identified with some sort of label. A multiresolution approach takes the view that an image can be broken into blocks of different sizes [3]. By using different block sizes it is possible for each block to represent only 1 texture in the image, thus producing the strongest texture feature measures. When a block has a weak measure, it can be split up into more blocks at a higher resolution (smaller size). In this way, the optimal size for each texture region in the image can be obtained.

Once the appropriate block sizes are obtained, it is necessary to "link" together all the similarly textured blocks. One way of achieving this is through the use of pyramid linking [17, 21, 34, 1, 30, 6]. In pyramid linking, a tree like structure is used to represent the image. The lowest level of resolution is the entire image and it is the root of the structure. Successively higher levels of resolution exist at higher levels of the pyramid. Each lower level in the pyramid is a subdivision of the previous level. That is, a node at level i in the pyramid will typically have 4 children at level $i+1$ which are 1/4 the size of the node. The leaf nodes in the pyramid represent the highest level of resolution which is usually the individual pixels in the image.

In pyramid linking, there are a number of possible ways to link similarly textured blocks. A bottom-up approach can be performed to link smaller blocks into larger ones. Structural neighbors (children of the same parent node) with similar texture can be merged into their parent. Alternatively, similar blocks of texture can all be linked to one common parent. This can continue all the way up the pyramid until all the nodes are linked to one of a small number of upper level nodes. A top-down approach can be used to split up nodes which do not have a strong texture feature measure. This splitting is carried out until all the nodes can no longer be split. At this point similar texture blocks can exist anywhere in the pyramid even though all the appropriate block sizes have been obtained. It is necessary to link these similar textured blocks (nodes in the pyramid) to obtain the desired homogeneous regions in the image. Samet [43, 44, 45] has developed many methods for finding the physically adjacent image blocks in the pyramid structure. Using these methods it is possible to link the remaining similarly textured blocks in a manner similar to that used in the bottom-up strategy.

Chapter 4

The Basic Approaches

This chapter presents the basic approaches used by the image segmentation algorithms being evaluated in this thesis. The Multiresolution Texture Segmentation (MTS) algorithm uses the *simulated annealing* approach in producing an image segmentation. The Constrained Optimization (CO) algorithm uses a similar approach in segmenting an image. In Section 4.1 the simulated annealing approach is described. In Section 4.2 the constrained optimization approach, used by the CO algorithm, is described. The CO algorithm is presented in Section 4.3 and the proposed MTS algorithm is presented in Section 4.4.

4.1 Simulated Annealing

Simulated Annealing [35, 52] is a technique used for optimization problems of a large scale. Typically, these problems involve combinatorial minimization. There is an objective function to minimize and the space in which the solution is obtained is a large, discrete configuration space such as all the possible allocations of labels to pixel blocks in an image. The space is so large that it is combinatorially impossible to exhaust every possible configuration. Simulated Annealing continually modifies an initial configuration until the objective function is minimized and in so doing utilizes a much smaller set of configurations from the space.

Simulated Annealing draws its analogy from thermodynamics. At high temperatures molecules of a liquid are allowed to move around freely. As the liquid cools the molecules begin to lose their mobility. If the liquid is cooled sufficiently, the molecules form a crystalline structure. This is the minimum energy state for the system. When a liquid cools rapidly, there is not enough time for the molecules to organize themselves in an orderly fashion before they lose their mobility. Thus, the system ends up in a polycrystalline state with higher energy. So the essence of Simulated Annealing is slow cooling allowing a system to settle into a state of low energy.

In 1953, Metropolis et al. [35] simulated a thermodynamic system. They assumed that the system would change its state with probability

$$p = e^{\frac{E_2 - E_1}{-kT}}. \tag{4.1}$$

Each time the system moves into a new state with energy E_2, the energy E_1, from the old configuration is compared to E_2 using (4.1) to provide a measure of how probable it is for the new configuration to occur given the old configuration. If $E_2 < E_1$ the new configuration will always occur (p is greater than unity). When the state change increases the energy in the system (i.e. $E_2 > E_1$) the new configuration will occur with probability p. As kT decreases (cools) the probability of going into a higher energy configuration decreases. When kT reaches its lowest temperature (i.e. $T = 0$) the system will have its lowest energy and the current configuration will provide the minimum solution.

In order to apply simulated annealing to a problem, the problem must be described in terms of the following components:

1. an objective function to minimize,
2. a description of possible configurations,
3. a generator of random changes in configuration and
4. the control parameter T (temperature) and a cooling schedule telling how to lower T from high to low.

To illustrate how a problem can be defined in terms of these components, consider the traveling salesperson problem. A description of the problem could be as follows:

1. Minimize cost to visit all cities x, $1 \leq x \leq N$.

2. Visit city x_i from city x_j, $i \neq j$, accumulating cost c_i.

3. Guess initial schedule of cities to visit. Randomly pick two cities from the schedule and determine the cost of going directly to each other and then the remaining cities in the schedule.

4. $T = T/2$, where T is the initial temperature reduced by 50% at each iteration.

The solution is obtained by continually performing 3-4 until the cost can not be reduced any further. Re-scheduling the order of the cities is always performed when the *cost for the new schedule* < the *cost for the old schedule*. When the *new cost* > the *old cost*, the probability that the schedule will still be changed can be determined using (4.1) with the new cost being E_2 and the old cost being E_1. A random number between 0 and 1 is selected and if the resulting probability is greater than the random number, the schedule is modified (increasing the total cost).

4.2 Constrained Optimization

Constrained Optimization [12] is similar to simulated annealing. It seeks to optimize an objective function by refining an initial configuration. However, instead of cooling a system to arrive at a low energy state, Constrained Optimization fixes the temperature and applies constraints to the objective function. The goal is to minimize the objective function while not violating any of the constraints.

The probability of a state change from a configuration with energy E_1 to a configuration with energy E_2 is

$$p = e^{\frac{E_2 - E_1 + \lambda V}{-T}}. \tag{4.2}$$

This equation is the same as (4.1) with the addition of the constraint conditions λV and the removal of the constant k. V represents the constraints imposed on the system and λ determines the rate at which the constraints are increased. A low energy configuration occurs when $V = 0$ (no constraints violated) because as $\lambda \rightarrow \infty$, $p \rightarrow 0$ which rules out the chance of any state changes producing a higher energy configuration.

State changes in Constrained Optimization can be controlled in a number of ways. The Constrained Optimization algorithm (see Section 4.2) presented in this thesis uses *stochastic relaxation* to achieve its state changes. Stochastic relaxation is a probabilistic method which enumerates the possible state changes from one configuration to the next and then uses the probabilities to choose the appropriate state change. After each state change, the probabilities are updated for the next state change. The objective function is minimized by performing the relaxation process until the probabilities controlling the state changes reach unity (the ideal condition).

The effect of V in (4.2) is dependent upon the penalty schedule λ. As stochastic relaxation proceeds, λ is increased, thus increasing the effects of the constraints on the objective function. Consequently, illegal configurations that may have been allowed early in the relaxation process are gradually being eliminated from the solution. Depending on the design, the effect of the constraints can be scheduled to occur early or in a gradual fashion. It is important to note that the constraints should not be applied too early in the relaxation process as the configuration can get stuck in a local minimum. On the other hand, too gradual a schedule should be avoided because it slows down the convergence of the algorithm.

4.3 The Constrained Optimization Algorithm

The characteristics of ultrasound images provide motivation to use a texture segmentation algorithm in order to produce an accurate segmentation. Since much of the research discussed in Chapter 3 dealt with texture classification or texture feature analysis, it was the goal to find an appropriate algorithm which could both identify textures and segment an image into regions based on the texture features in the image. Also, a long term goal of the thesis research is to provide a basis for which a real time solution for 3D reconstruction of the ultrasound images can be achieved. An algorithm which incorporates these elements (with perhaps an exception for real time computation) is the Constrained Optimization algorithm developed by Geman et al. [12]. This algorithm uses a stochastic relaxation process to label similar texture blocks in an image as belonging to the same region. A global measure of the segmentation process is continually updated and when the measure reaches its minimum, the segmentation process is complete.

The components, presented in the previous section of Constrained Optimization are the foundation of Geman et al.'s algorithm. Recall that in Constrained Optimization, an objective function is to be minimized while taking into consideration the constraints that are imposed on the minimization process. In this algorithm, the function is a measure of labels and pixels, U(*labels,pixels*), which encourages the placement of boundaries between different texture regions. Minimizing U results in an optimal labeling of the image into its appropriate regions. The constraints imposed on U inhibit the formation of illegal label configurations. For example, very thin regions or very small regions are not allowed to exist.

Two models are proposed in Geman et al.'s algorithm, namely, the region partitioning and boundary models. The partition model assigns labels to blocks of pixels in the image. Each label determines to which region a block of pixels belongs. The boundary model uses labels to classify "boundary label sites" as being either "on" or "off" the boundary between different blocks of pixels. The resulting segmentation is an edge map separating differently textured regions in the image. The boundary detection model is more effective for complex, multi-textured images while the partition model is appropriate when a small number of homogeneous regions are present. Since ultrasound images fall into the latter category, the partition model is being investigated in this thesis. The goal of the partition model is to partition an image so that all regions in the image having the same texture are determined to belong to the same region. A lattice of *label sites* is placed on the image. Each label site S, in the lattice, is located at pixel $(\sigma i+1, \sigma j+1)$ where $0 \leq i, j < \frac{N-1}{\sigma}$ and N is the size of the image (assuming the image is NxN pixels). Each block of pixels is centered around site S. The region to which each block belongs is determined by the label at S. Site S is assigned the label which minimizes the energy between itself and its neighboring sites. The optimization function U(*labels,pixels*) measures this *interaction energy* and is expressed as

$$U(labels, pixels) = \sum_{\substack{all\ label \\ pairs\ s,t}} \delta_{\{label_s = label_t\}} \Phi(pixels\ at\ s,\ pixels\ at\ t) \tag{4.3}$$

where

$$\delta_{\{\ \}} = \begin{cases} 1 & \text{if } label_s = label_t \\ 0 & \text{otherwise} \end{cases}$$

$$\Phi(\ ) = \begin{cases} -1 & \text{if texture of pixels at s,t are } \textbf{similar} \\ 1 & \text{if texture of pixels at s,t are } \textbf{distinct}. \end{cases}$$

Similarity of pixel blocks is determined through the use of texture measures (see Section 3.2). In (4.3) Φ measures the amount of disparity between pixel blocks.

The disparity measures between two blocks of pixels are obtained by evaluating different texture features with the Kolmogorov-Smirnov (KS) statistic [12]. The KS statistic measures the difference between two probability distributions. In this case, the probability distributions are normalized cumulative distributions of the histograms for each pixel block. The KS statistic is defined to be the maximum distance between the two cumulative distributions of the data sets V_1 and V_2

$$d(V_1, V_2) = \max_{-\infty < t < \infty} |F_1(t) - F_2(t)| \tag{4.4}$$

where

$V = (v_1, v_2, ..., v_n)$ and

$F(t) = \frac{1}{n}\#\{k : v_k \leq t\}$, the cumulative distribution function. The limits placed on t are dependent on the transformation of the intensity values in V. In practical applications, the limits of t can be determined by

lower limit: min(min(V_1),min(V_2))
upper limit: max(max(V_1),max(V_2)).

When the raw intensities are used usually $0 \leq t \leq 255$.

The penalty function V(*labels*) is based upon forbidden label configurations. Penalties are assigned for configurations producing:

- very small regions and
- very narrow regions.

A 5x5 block of label sites centered at S is examined in determining if the application of a label at S will produce a small region. Site S will incur a small region penalty if less than 9 of the 24 sites in the block have the same label as S. Site S will incur a narrow region penalty if both horizontally or vertically adjacent sites have different labels than S. The total penalties for S can thus be expressed as

$$V(labels) = \delta_{\{S\ small\}} + \delta_{\{S\ narrow\ vert.\}} + \delta_{\{S\ narrow\ horz.\}}. \tag{4.5}$$

The optimum solution (best segmentation) is achieved when (4.3) is minimized with the penalty function V(*labels*) = 0.

The steps in Geman et al.'s segmentation algorithm can be summarized as follows:

1) assign initial labels to each site in the label lattice
For each iteration
2) update λ, the penalty function coefficient

For each site in the label lattice

3) select label site S to visit according to a site visitation schedule

4) calculate the texture measures of pixel block at S and its neighbors

For each possible label

5) calculate the KS measures between S and each of its neighbors

6) calculate U based on the current label assignments and KS values

7) using (4.2) and U update the label probability

8) choose the label for S which has the highest probability

4.4 The Multiresolution Texture Segmentation Algorithm

This section describes the Multiresolution Texture Segmentation (MTS) algorithm, developed in this thesis to produce a more accurate and faster segmentation than the CO algorithm. A problem with the CO algorithm is that it fails to take varying texture region sizes within an image. The resulting segmentation is dependent on the block size used to sample a local area of texture in the image. If the block size is small then a large amount of unnecessary computation will be performed when identifying large regions. When a large block is used, detail is lost as differences in neighboring texture regions are averaged out. It is more appropriate to use varying block sizes to sample texture in an image. This varying block size approach is used in the the proposed Multiresolution Texture Segmentation (MTS) algorithm.

In the MTS algorithm, the image is stored in a quadtree structure (see Section 3.2.3) with the root node representing the entire image and the leaf nodes representing the individual pixels in the image. Lower levels in the quadtree achieve finer resolution, non-overlapping blocks in the image. This is conceptually the same as applying a lattice on top of the image as in Geman et al.'s approach except that there is a corresponding lattice for each level in the quadtree. The problem now is to devise a method for measuring texture between blocks of varying sizes at different levels of the quadtree. Also at issue is the procedure for determining which blocks within the quadtree should be used to measure the texture within the image. For example, a block at level x is used to measure texture for a block of pixels in the image but, if the region with that texture is large, it may be more appropriate to use the block's parent at level *x-1*. Similarly, if the region is smaller than the block size, the children at level *x+1* will provide a more accurate measure of the texture in the region. This leads to the use of a *split and merge* approach in selecting which blocks are to be used in measuring texture in an image. A block is split if its 4 children produce a stronger measure than its own. A block is merged with its 3 neighbors if its parent produces a stronger measure than its own and its 3 neighbors.

The MTS algorithm uses simulated annealing to control splitting and merging of blocks. There are no constraints imposed on the cooling (minimization) process and unlike the CO algorithm the temperature is reduced at a scheduled rate rather than remaining constant. The energy for a configuration in the system is a function of the quadtree structure x. Note that the structure of x is different for each new configuration due to the splitting and merging of nodes.

The energy for a configuration in the system is represented as

$$U(x) = \sum_{C \in S_x} \Phi_x(C) \tag{4.6}$$

where

x is the quadtree structure

C is a leaf node in the quadtree

S_x is the set of leaf nodes in x.

$\Phi_x(C)$ measures how well the current block fits the texture region in the image. It is defined as

$$\Phi_x(C) = d_s + d_m. \tag{4.7}$$

The measures d_s and d_m indicate whether or not splitting or merging will take place.

Splitting occurs when $d_s \geq 0$ and is expressed as

$$d_s = (2\delta(s_r > r) - s_d - 1)\,\delta(R < .5) \tag{4.8}$$

where

$s_d = \delta(d_C(P) > t)$ and

$s_r = e^{\frac{E-t}{kT}}$.

Merging occurs when $d_m \geq 0$ and is expressed as

$$d_m = (2\delta(m_r > r) - m_d - 1)\,\delta(R \geq .5)\delta_C(P) \tag{4.9}$$

where

$m_d = \delta(d_C(P) \leq t)$

$m_r = e^{\frac{t-E}{kT}}$

$$\delta_C(P) = \begin{cases} 0 & \text{if any child, } C_i\text{, of P is a parent node} \\ 1 & \text{otherwise} \end{cases}$$

and P is the parent of C.

In (4.8) and (4.9) R and r are random numbers in the range [0,1] and t is a pre-determined threshold. In the MTS algorithm, $t = 0$. In (4.9), $\delta_C(P)$ discourages merging when at least one of the child nodes, C_i, has children of its own. A node with its own children indicates that there are at least two distinct types of textures within its block. Merging these types of nodes results in averaging distinct textures into one: the exact problem occurring in Geman et al.'s algorithm when the block size is too large.

Both s_d and m_d in (4.8) and (4.9) are based on the following equation

$$d_C(P) = {}^{MIN}_{1 \leq k < i} \left(\sum_{j=k+1}^{i} \delta\left(T(C_k) \neq T(C_j)\right) \right) \tag{4.10}$$

where

i is the number of child nodes of P, since quadtree structure is used, i=4

$T(C_m)$ indicates the texture class of child block m.

Equation (4.10) defines the number of children with a texture class which is different from the texture class of the majority of the children. For example, if the texture classes for the four children of P are represented as {1,1,2,3} then the texture class of the majority is 1 and the number of children with a different texture class is two, $d_C(P) = 2$. If a child can not be identified as belonging to any of the texture classes it is called unknown and $T(C_m)$ is defined to be 0 (characteristics for the texture classes are determined *a priori*, see Chapter 5). When all the children are identified as belonging to the same texture class, $d_C(P) = 0$ and $m_d = 1$, indicating that merging is possible.

When both m_d and s_d are 0, a node may be randomly split or merged (random state change). A random state change is performed to minimize the chances of getting stuck in a local minimum solution and is characteristic of the simulated annealing technique. The values of s_r and m_r determine if a random state change will occur. When $R < .5$ random splitting will occur if

$$s_r = e^{\frac{E-t}{kT}} > r \tag{4.11}$$

otherwise $R \geq .5$ and random merging will occur if

$$m_r = e^{\frac{t-E}{kT}} > r \tag{4.12}$$

where $E = d_C(P)$, the threshold $t = 0$ and r is a uniformly distributed random number in the range [0,1]. As the temperature kT decreases the probability of a random state change becomes much less likely.

When both $d_s = 0$ and $d_m = 0$ in Equation (4.7), the node C is either split or merged and the energy in U remains the same since $\Phi_C(P) = 0$. When either $d_s = 1$ or $d_m = 1$ the energy in U is increased since $\Phi_C(P) = 1$. This occurs due to a random state change. When either $d_s = -1$ or $d_m = -1$ node C is not split or merged and the energy in U is decreased since $\Phi_C(P) = -1$. The minimum energy in U occurs when no splitting or merging takes place for every leaf node in S_x.

Using Equation (4.7) and applying it to the simulated annealing framework the MTS algorithm performs a segmentation of an image by iterating until no more splitting and merging takes place or some maximum number of iterations is reached. In the first case the minimum energy state is reached. In the second case an approximation to the solution results. It is possible that this approximation does not provide a reliable estimate but it can decrease the time it takes to produce an initial segmentation.

Since blocks are labeled as they are examined, when iteration completes, all blocks should be labeled. In some cases (terminating the segmentation process at a maximum number of solutions, for example) it is possible that some unknown blocks (blocks having no texture class label) may exist. The identity of these blocks is determined by assigning each block to the texture class with the closest characteristics to the block.

4.4.1 Pseudocode

This section presents the pseudocode outlining the steps involved in the MTS algorithm.

1) choose an initial resolution, (the level in the quadtree) at which pixel
   blocks will be examined

For each temperature

  2) decrease the temperature at the cooling rate of α

  For Ω iterations

    3) randomly select a node (block), C, in S_x

    4) use R to determine if splitting or merging of node C is to occur

    if split

      a) examine the children of block C using (4.10)

      if $s_d = 0$

b) if $\delta(s_r > r) = 1$, split and assign unknown labels

c) else split and update labels

else merge

a) examine block C and its neighbors using (4.10)

if $m_d = 0$

b) if $\delta(m_r > r) = 1$, merge and assign unknown label

c) else merge and update label

6) update U with (4.6)

7) if no splits or merges occurred (U is minimum), then all blocks labeled so exit

8) reached maximum iterations so assign closest texture class to the label of any unknown blocks

Chapter 5

Texture Characterization

The purpose of texture characterization is to produce a set of measurements which identify different types of textures. This does not necessarily require that textures be classified according to the measures obtained; rather, the goal is to determine the characteristics of texture classes based on the measures. In texture characterization, it is desirable to use measurements which produce unique characteristics for each type of textures. This allows a texture class to be described by a set of parameters which can be used by a segmentation algorithm to partition an image into homogeneous texture regions.

Figure 5.1 illustrates how texture characterization is used in the MTS algorithm. The Feature, Texture Characterization, and Feature Table components are described in this chapter. Section 5.1 describes the 32 features used to characterize different classes of textures. Section 5.2 describes how the features are used in characterizing different classes of textures. Section 5.3 discusses how robust statistics are used in identifying outliers. Outliers are defined as data which are inconsistent with the rest of the population, in a set of texture measures. Section 5.4 describes how the outlier descriptions of the texture measures are used to rank the features and how these rankings are used by the segmentation component of the MTS algorithm. Section 5.5 shows the results of the features in characterizing the different texture classes present in a series of diagnostic ultrasound images of an ovarian follicle. Highlights of this chapter are summarized in Section 5.6.

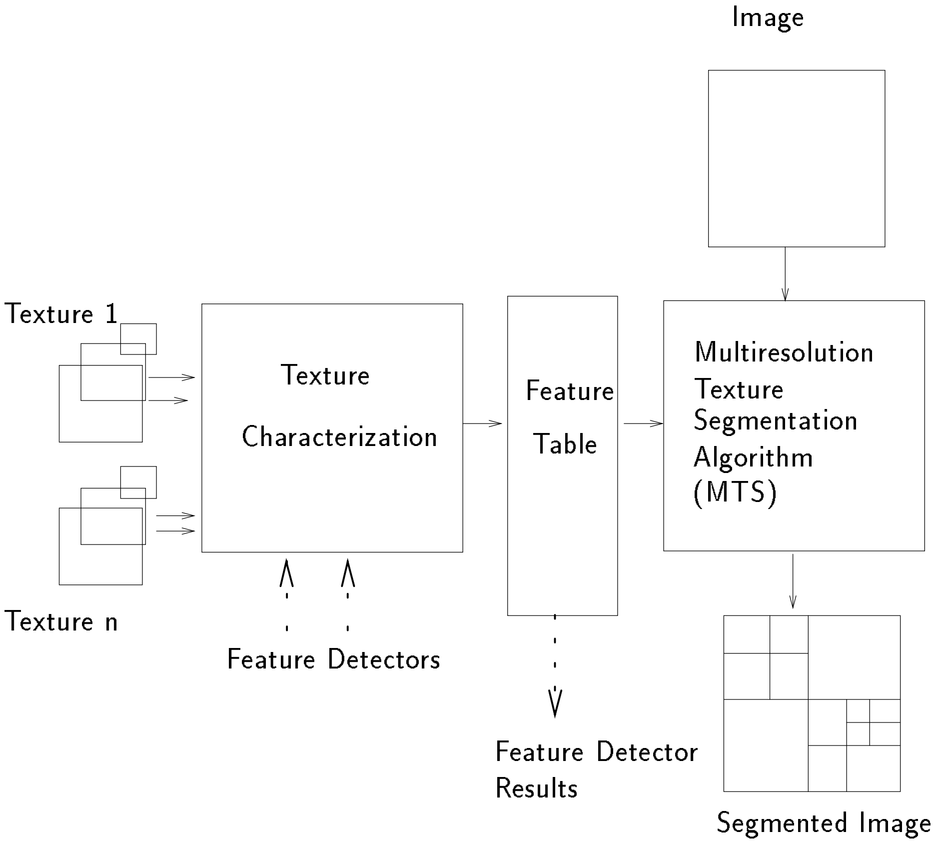


Figure 5.1: Block diagram of the MTS algorithm. The feature table, used in the segmentation process, contains statistics on the effectiveness of the features in characterizing the texture classes.

5.1 Features

Features can be computed by combining pixel intensities, in many different ways, into a set of numbers which hopefully represent the texture of interest. In this thesis, 21 spatial and 11 frequency domain features are used to characterize textures in ultrasound images. Some of these 32 features are based on the raw pixel intensities, $p_{i,j}$, while others are based on transformations of the raw intensities, $p^m_{i,j}$. By transforming the raw pixel intensities it is possible to enhance some characteristics that are specific to a particular type of texture.

Four pixel intensity transformations, in addition to the raw intensities, are used. Each transformation is obtained from an N×N block of pixels and defined as follows:

1. Gradient magnitude

$$p^{(1)}_{i,j} = |p_{i,j} - p_{i+1,j}| + |p_{i,j} - p_{i,j+1}|,$$

2. Residual

$$p^{(2)}_{i,j} = |p_{i,j} - \bar{p}| \quad \text{where} \quad \bar{p} = \frac{1}{N^2} \sum_i \sum_j p_{i,j},$$

3. Horizontal directional residual

$$p^{(3)}_{i,j} = \left| p_{i,j} - \frac{p_{i-1,j} + p_{i+1,j}}{2} \right|,$$

4. Vertical directional residual

$$p^{(4)}_{i,j} = \left| p_{i,j} - \frac{p_{i,j-1} + p_{i,j+1}}{2} \right|.$$

From these transformations, five sets are defined for each block of pixels:

$$
\begin{aligned}
P_1 &= \{p_{i,j}^{(1)}\}, \\
P_2 &= \{p_{i,j}^{(2)}\}, \\
P_3 &= \{p_{i,j}^{(3)}\}, \\
P_4 &= \{p_{i,j}^{(4)}\}, \\
P_5 &= \{p_{i,j}\},\ 0 \le i, j < N = \beta.
\end{aligned}
$$

The spatial domain feature $(f_1, \cdots, f_{21})$ are computed using the above sets. The KS distance (see Section 4.3) is used to derive features $f_1, \cdots, f_5$, each of which is defined as follows:

$$
\begin{aligned}
f_1 &= KS(CD_1, NCD_1), \\
f_2 &= KS(CD_2, NCD_2), \\
f_3 &= KS(CD_3, NCD_3), \\
f_4 &= KS(CD_4, NCD_4), \\
f_5 &= KS(CD_5, NCD_5).
\end{aligned}
$$

where CD_m is the cumulative distribution function for set P_m for a given texture class and NCD_m is the normalized cumulative distribution function for pixel transformation m of a representative sample of pixel blocks (size β) from the same texture class (Section 5.5.1 describes how a representative sample is obtained for a given texture class). NCD_m is based on the transformed pixel values $p_{i,j}^{(m)}$, of each block in the sample, which are obtained by transforming the original pixel intensities, $p_{i,j}$, in each block, with either the gradient magnitude (m=1), the residual (m=2), the horizontal directional residual (m=3), the vertical directional residual (m=4) or using the original pixel intensities (m=5). The normalized cumulative distribution function for transform m, $1 \le m \le 5$, of the pixel intensities in each block of a representative sample is computed as follows:

1) For each sample block i, 0< i ≤S (sample size S), of a given texture class
   a) create histogram $Hist_m^{(i)}(P_m),\ 1 \le m \le 5$, from the set P_m,
   b) compute the cumulative distribution function, $CD_m^{(i)}$ from $Hist_m^{(i)}(P_m)$.

2) Compute the normalized cumulative distribution function, NCD_m such that

$$NCD_m(j) = LMS\left(\left\{\bigcup_{1\le i\le S} CD_m^{(i)}(j)\right\}\right)$$

where

NCD_m(j) is the jth bin of NCD_m,

LMS(B) is the value minimizing the Least Median of Squares for set B.

The LMS is a robust statistical method based on the Least Median of Squares approach, the discussion of which can be found in Section 5.3. Using the LMS in step 2) makes normalization less susceptible to outliers present in the representative sample.

Features f_6 and f_7 use the Euclidean distance from $(f_1, \cdots, f_5)$ to the mean, $\bar{f}_i$, and median, $\hat{f}_i$, respectively. In particular, feature f_6 is defined as:

$$f_6 = \sqrt{(\bar{f}_1 - f_1)^2 + (\bar{f}_2 - f_2)^2 + (\bar{f}_3 - f_3)^2 + (\bar{f}_4 - f_4)^2 + (\bar{f}_5 - f_5)^2} \tag{5.1}$$

and feature f_7 is defined as:

$$f_7 = \sqrt{(\hat{f}_1 - f_1)^2 + (\hat{f}_2 - f_2)^2 + (\hat{f}_3 - f_3)^2 + (\hat{f}_4 - f_4)^2 + (\hat{f}_5 - f_5)^2} \tag{5.2}$$

where $\bar{f}_i, (1 \le i \le 5)$ is the average of feature f_i, $(1 \le i \le 5)$, (for the representative sample) and $\hat{f}_i, (1 \le i \le 5)$ is the median of feature f_i, $(1 \le i \le 5)$.

Features $f_8, \cdots, f_{12}$ are based on the five transformed sets of data $P_m, 1 \le m \le 5$, just as features $(f_1, \cdots, f_5)$ except that the average deviation of the pixel intensities, $p_{i,j}^{(m)}$, is used as the measure. In particular,

$$\begin{aligned} f_8 &= ADev(P_1), \\ f_9 &= ADev(P_2), \\ f_{10} &= ADev(P_3), \\ f_{11} &= ADev(P_4), \\ f_{12} &= ADev(P_5), \end{aligned}$$

where

$$ADev(P_m) = \frac{1}{N^2}\sum_i \sum_j |p_{i,j}^{(m)} - \bar{p}^{(m)}|,\ 1 \leq m \leq 5.$$

Features f_{13} and f_{14} are based on the Euclidean distance from $(f_8, \cdots, f_{12})$ to the mean, $\bar{f}_i$, and median, $\hat{f}_i$, respectively. They are defined in exactly the same way as in Equations (5.1) and (5.2) with the exception that the subscripts (1,$\cdots$,5) are replaced with the subscripts (8,$\cdots$,12). Feature f_{13} is defined as:

$$f_{13} = \sqrt{(\bar{f_8} - f_8)^2 + (\bar{f_9} - f_9)^2 + (\bar{f_{10}} - f_{10})^2 + (\bar{f_{11}} - f_{11})^2 + (\bar{f_{12}} - f_{12})^2}$$

and feature f_{14} is defined as:

$$f_{14} = \sqrt{(\hat{f_8} - f_8)^2 + (\hat{f_9} - f_9)^2 + (\hat{f_{10}} - f_{10})^2 + (\hat{f_{11}} - f_{11})^2 + (\hat{f_{12}} - f_{12})^2}.$$

Features $f_{15}, \cdots, f_{19}$ use the standard deviation, rather than the average deviation, as the measure and are defined as follows:

$$\begin{aligned} f_{15} &= SDev(P_1), \\ f_{16} &= SDev(P_2), \\ f_{17} &= SDev(P_3), \\ f_{18} &= SDev(P_4), \\ f_{19} &= SDev(P_5), \end{aligned}$$

where

$$SDev(P_m) = \sqrt{Var(P_m)},$$

$$Var(P_m) = \frac{1}{N^2}\sum_i \sum_j \left(p_{i,j}^{(m)} - \bar{p}^{(m)}\right)^2,\ 1 \leq m \leq 5.$$

Features f_{20} and f_{21} are based on the Euclidean distance from $(f_{15}, \cdots, f_{19})$ to the mean and median of $(f_{15}, \cdots, f_{19})$ and are defined as

$$f_{20} = \sqrt{(\bar{f_{15}} - f_{15})^2 + (\bar{f_{16}} - f_{16})^2 + (\bar{f_{17}} - f_{17})^2 + (\bar{f_{18}} - f_{18})^2 + (\bar{f_{19}} - f_{19})^2}$$

and

$$f_{21} = \sqrt{(\hat{f_{15}} - f_{15})^2 + (\hat{f_{16}} - f_{16})^2 + (\hat{f_{17}} - f_{17})^2 + (\hat{f_{18}} - f_{18})^2 + (\hat{f_{19}} - f_{19})^2},$$

respectively.

Features $f_{22}, \cdots, f_{32}$ are defined in the frequency domain. A subset of the features defined in [29, 25] compose the majority of the features used (features f_{22}, $f_{25}, \cdots, f_{28}$ and f_{32}). These features are based on the power spectrum, $P(u,v)$, for a pixel block from the representative sample for a given texture class. The power spectrum is normalized resulting in the following definition:

$$p(u,v) = \frac{P(u,v)}{\sum_{u,v \neq 0} P(u,v)}.$$

The spatial frequency domain features are as follows:

f_{22}, energy in major peak,

$$f_{22} = p(u_1, v_1) \cdot 100$$

where u_1, v_1 are the frequency coordinates of the maximum peak of the normalized power spectrum.

f_{23}, relative energy in the power spectrum, R_1,

$$f_{23} = \frac{E_1}{E_1 + E_2 + E_3 + E_4}$$

where

$$E_i = \sum_{u,v \epsilon R_i} P(u,v)$$
$$R_i = \left\{ \ u, v \text{ such that } \tfrac{i-1}{4} u_m \leq |u| < \tfrac{i}{4} u_m \ and \ \tfrac{i-1}{4} v_m \leq |v| < \tfrac{i}{4} v_m \ \right\}$$

u_m, v_m are maximum frequency components for the local spectrum (i.e. for an N×N texture block, u_m = N and v_m = N). Figure 5.2 illustrates the four regions in the frequency domain.

f_{24}, relative energy in the power spectrum, R_2,

$$f_{24} = \frac{E_2}{E_1 + E_2 + E_3 + E_4}.$$

f_{25}, relative entropy of the power spectrum, R_1,

$$f_{25} = -\sum_{u,v \epsilon R_1} p_1(u,v) \log p_1(u,v) \cdot \frac{1}{\log K_1}$$

where

$$p_i(u,v) = \frac{P(u,v)}{\sum\limits_{u,v \epsilon R_i} P(u,v)}$$

$$K_i = \text{number of distinct frequencies in } R_i.$$

f_{26}, relative entropy of the power spectrum, R_2,

$$f_{26} = -\sum_{u,v \epsilon R_2} p_2(u,v) \log p_2(u,v) \cdot \frac{1}{\log K_2}.$$

f_{27}, relative entropy of the power spectrum, R_3,

$$f_{27} = -\sum_{u,v \epsilon R_3} p_3(u,v) \log p_3(u,v) \cdot \frac{1}{\log K_3}.$$

f_{28}, relative entropy of the power spectrum, R_4,

$$f_{28} = -\sum_{u,v \epsilon R_4} p_4(u,v) \log p_4(u,v) \cdot \frac{1}{\log K_4}.$$

f_{29}, Euclidean distance from the means of $f_{22}, f_{23}, f_{24}, f_{25}$,

$$f_{29} = \sqrt{(\bar{f_{22}} - f_{22})^2 + (\bar{f_{23}} - f_{23})^2 + (\bar{f_{24}} - f_{24})^2 + (\bar{f_{25}} - f_{25})^2}.$$

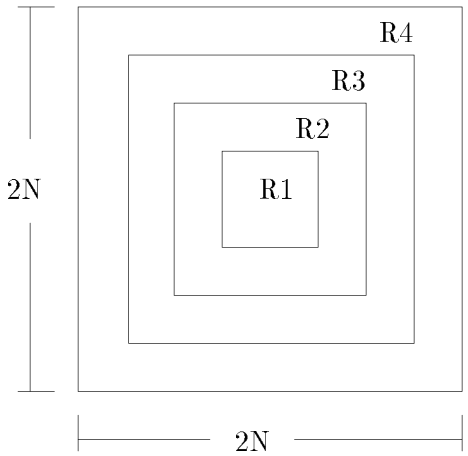


Figure 5.2: The four regions, R_1-R_4, of the 2Nx2N power spectrum for an NxN block of pixels in the spatial domain. The maximum frequency components, $u_m, v_m, = |\mathrm{N}|$.

f_{30}, Euclidean distance from the medians of $f_{22}, f_{23}, f_{24}, f_{25}$,

$$f_{30} = \sqrt{(\hat{f_{22}} - f_{22})^2 + (\hat{f_{23}} - f_{23})^2 + (\hat{f_{24}} - f_{24})^2 + (\hat{f_{25}} - f_{25})^2}.$$

f_{31}, Euclidean distance from the mean of $f_{25}, f_{26}, f_{27}, f_{28}$,

$$f_{31} = \sqrt{(\bar{f_{25}} - f_{25})^2 + (\bar{f_{26}} - f_{26})^2 + (\bar{f_{27}} - f_{27})^2 + (\bar{f_{28}} - f_{28})^2}.$$

f_{32}, Euclidean distance from the mean of $f_{22}, f_{24}, q1, q2$,

$$f_{32} = \sqrt{(\bar{f_{22}} - f_{22})^2 + (\bar{f_{24}} - f_{24})^2 + (\bar{f_{q1}} - f_{q1})^2 + (\bar{f_{q2}} - f_{q2})^2}$$

where

$$q1 = \sum_{u>0} \sum_{v>0} p(u,v)$$

$$q2 = \sum_{u<0} \sum_{v>0} p(u,v)$$

q_1 and q_2 are the percentages of energy in quadrants 1 and 2, respectively.

5.2 The Characterization Process

Using features to classify texture is a common practice. Traditionally, a set of features is used to characterize homogeneous textures from a known texture class. The resulting measures are collected for different classes of textures. A block with an unknown texture class can be classified into one of the known classes by finding the class which is closest to it according to some predefined metric.

It is common to see that techniques on texture characterization place too much emphasis on finding an optimal set of features which can classify unknown texture blocks with a low degree of error. Little consideration has been given to the relationship between the performance of the choice of features with the size of the texture blocks on which the features are used. Certainly, no existing approach takes into consideration the size of the texture blocks when determining the classification accuracy of the features.

A totally different and novel approach to texture characterization is used in the MTS algorithm. The uniqueness of this approach is based on two major points. First, the relationship between the features and the size of the texture blocks to which they are applied is considered. A feature which can characterize a texture class well for a large block may poorly characterize a small block. Because of this inconsistency inherent in textures found in most real images, the texture characterization approach in the MTS algorithm uses only features which can best characterize texture classes at each different resolution, β (where the values of β are predetermined). In this way, it is possible that a number of different features will be considered optimal for characterizing texture classes, depending on the resolution at which the blocks are examined. Second, classification is not used to determine which of the features is optimal for distinguishing among different texture classes. Rather, outlier detection within each texture class is used as a basis for rating the ability of features in characterizing texture (see Section 5.3).

The characterization process in the MTS algorithm is designed so that any number of features can be used to produce measures for homogeneous texture blocks at varying resolutions, β. For each texture class C_m, S (a large number) pixel blocks (or a representative sample, discussed in Section 5.5.1) at resolution β are examined. Each feature, f_i, is applied to each block j, $0 < j \leq S$, in the sample (of size S) resulting in the set F_i, $F_i = \{measure_j\}$ where $0 < j \leq S$ and $measure_j$ is the measure obtained for block j representing feature f_i. Thus, there are S measures, $|F_i|$ = S, for each feature f_i.

At this point in the characterization process, the sets F_i contain all the raw measures representing texture in samples of homogeneous blocks at resolution β for texture class C_m. It is desirable to represent these measures in a compact form so that the feature table (see Section 5.4) can determine how the features rank in characterizing the different classes of textures at varying resolutions. This is accomplished by determining the number of outlier measures in F_i (see Section 5.3). Assuming that each measure in F_i is obtained from blocks in the same texture class, by the same feature, it follows that the measures should be similar. Outliers in F_i suggest that either some of the blocks are not from the same texture class or that feature f_i can not produce consistent measures for blocks in the same texture class. Since it is assumed that the blocks are all samples of the same texture class it is only possible that f_i can not consistently characterize blocks from the same texture class. The number of outliers detected in F_i determines how well f_i characterizes blocks at resolution β belonging to the same texture class, C_m.

Using outliers to describe the feature measures, each set F_i can be reduced to the triplet $\{M, r, o\}$. In the case of a Guassian distribution, $M \pm r$ is the central region in which 98.7% of the measures are located. The acceptable amount of deviation from M is the range r. The number of outlier measures (measures beyond $M \pm r$) is contained in o.

5.3 Robust Statistics and Outlier Detection

Outlier detection has its roots in robust statistics [42] which are a class of techniques used to estimate statistical properties of data contaminated with data called outliers.

Outliers have properties which are inconsistent with the general population. By accounting for these outliers, it is possible to determine the extent to which the general population of data has been corrupted. Accountability of outliers is exactly what is required when examining a set of feature measures produced for a sample of texture blocks from a given texture class (see Section 5.2). For blocks sampled from the same texture class, a feature measure is considered an outlier when it is applied to a block from the sample and the resulting measure is not consistent with the general measure obtained for the majority of blocks in the same sample. Since all blocks are of the same texture class, the presence of outliers indicate that the feature fails to characterize the texture. A set of feature measures corrupted with a large number of outlier measures indicates that the feature may not be appropriate for characterizing blocks of the given texture class. A low number of outliers, however, indicates that the feature may be able to produce a consistent measure for the blocks while attributing outlier measures to random sampling error, provided the sample size is statistically significant.

The Least Median of Squares (LMS) is a robust statistical method which computes a value and range for a data set [42, 55]. The value, M, is the median of the range of values (the range having length $2m_T$) which minimizes the LMS error. The range, r, is set to the commonly used value $2.5 \cdot 1.4826 \cdot m_T$ [55]. The value $1.4826 \cdot m_T$ forces r to have the same value as the standard deviation estimate for a Guassian distribution. Points beyond the range, $M \pm r$, do not fall within 98.7% of the samples in a Gaussian distribution and are thus termed outliers. The description of the set F_i of feature measures using the LMS method results in the triplet $\{M, r, LMS(F_i)\}$ where $LMS(F_i)$ is the number of outliers detected in the set F_i using the LMS method. This representation is used as a basis for ranking, and collecting statistics on, a feature's ability to characterize different classes of textures.

5.4 The Feature Table

The accuracy in which image segmentations are achieved with the MTS algorithm is largely dependent on how well each texture class in an image has been characterized. The feature table is at the heart of this characterization. Figure 5.1 shows the interaction between the feature table and the other main components of the MTS algorithm. The feature table takes the large amount of data collected from the characterization process (see Section 5.2) and ranks each of the features used in characterizing the different texture classes. Only a small subset of the features evaluated in the characterization process are used in the segmentation component of the MTS algorithm because the segmentation component only uses the top ranking feature at each resolution. The feature table also produces statistics on each of the features including the number of within class outliers as well as the number and amount of overlapping regions among classes (see Section 5.5 for examples of these statistics).

Ranking of the features is performed so that the statistically best feature at each resolution is used in the segmentation process. Each feature is ranked according to the following scheme:

$$\begin{aligned} R(\beta, f_i) &= Within\ Class\ Outliers + Among\ Class\ Overlap \\ &= \sum_j outliers(f_i, \beta, j)\ + W \cdot \sum_j \sum_{k=j+1} \delta\{C_j \cap C_k\} \end{aligned} \tag{5.3}$$

where

$outliers(f_i, \beta, j)$ = outliers for feature f_i at resolution β for texture class j

$\delta\{C_j \cap C_k\} = 1$ if C_j intersects C_k, 0 otherwise

$C_m = \{M_m, r_m, o_m\}$ the measures for f_i in terms of its outliers for texture class m

W is a weighting constant.

The value for subscript i in Equation 5.3 is dependent on the number of features used in the texture characterization process. Since at least one feature is required, i is any integer > 0. The value of the subscript j is determined by the number of texture

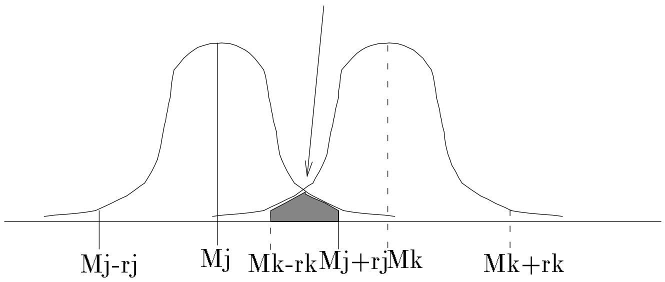


Figure 5.3: Measures obtained by the same feature for blocks from two different texture classes, C_j and C_k. Overlap in the shaded area results when the range of measures $[M_j - r_j, M_j + r_j]$ for blocks from texture class C_j intersects the range of measures $[M_k - r_k, M_k + r_k]$ for blocks from texture class C_k. The texture class of blocks having measures in this region can not be determined.

classes. The function $\delta\{C_j \cap C_k\}$ determines if the feature measures for blocks in one texture class overlap with those in a different texture class. Feature measures existing in the overlapping region indicate that it is impossible to determine the texture class for the blocks having these measures. An overlapping region occurs between two texture classes when $[M_j - r_j, M_j + r_j] \cap [M_k - r_k, M_k + r_k] \neq \emptyset$ (*i.e.* the feature measures, excluding outliers, obtained for blocks in texture class j overlap with those from texture class k). Figure 5.3 provides an example in which measures for the same feature from two different texture classes overlap. Note that the scheme used for detecting overlapping regions is appropriate for a small number of texture classes. If the number of texture classes is larger than 3 or 4, it will be necessary to map the range of measures for each texture class into a larger space to reduce the chances of overlapping regions occurring among the texture classes.

W is chosen so that either within class outliers or among class overlap dominates the ranking R. For all the experiments in Section 5.5, 32 features were used to rank 2 and 3 texture classes with region overlap dominating the feature rankings. Thus, $0 < i \leq 32$, $0 < j \leq 2$ and $0 < j \leq 3$, and $W >> outliers(f_i, \beta, j)$.

Using the ranking scheme in Equation 5.3, features are ranked in the feature table from top (lowest R) to bottom (highest R) for each resolution β, within each texture class C_m. Each entry in the table is stored as a triplet $\{f, r_l, r_u\}$. This triplet is a transformation of the feature measures $\{M, r, o\}$ for feature f_i. The o parameter is not needed since the number of outliers has no consequence in the segmentation component. The parameters M and r are replaced by r_l, $r_l = M - r$, and r_u, $r_u = M + r$. Representing the range of measures for feature f_i in this way allows for quick access by the segmentation component. The feature with which the range of measures was obtained is represented by f.

When the segmentation component is examining a block at resolution β, it consults the feature table to determine which feature to use as well as the ranges of measures obtained by the feature for each texture class. The feature is applied to the block and the resulting measure is compared to each of the ranges obtained from the feature table. If the measure is within one of the ranges, then the class to which the block belongs is known. If the measure is beyond all ranges, then the texture class to which the block belongs can not be determined; in which case the block must be split because it does not have a homogeneous texture. If the measure is within two or more ranges, then splitting or merging of the block must occur because it is impossible to determine which texture class the block belongs to. This happens when all the features could not uniquely characterize each texture class at resolution β.

The segmentation component determines which entries in the feature table to use according to the parameters $\{rank, \beta, C\}$. The $rank$ parameter indicates the ranking, within the feature table, of the feature desired. In almost all cases $rank = 1$, indicating that the top ranking feature is to be selected. The only case where $rank \neq 1$ is when the same feature is to be used at each resolution as in experiment 7 in Section 6.3.5. The value for β is determined by the resolution of the block under examination. C indicates the texture class from which $\{f, r_l, r_u\}$ is calculated. When examining a block during the segmentation process, all values of C are used (*i.e.* one table entry for each texture class C is accessed).

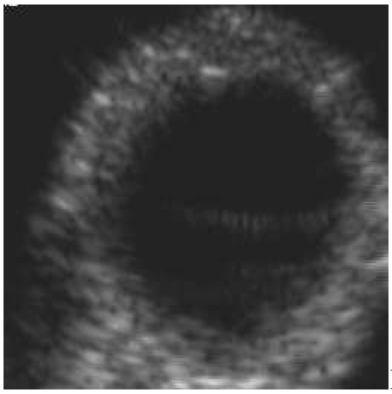
5.5 Experiments

The use of synthetic textures, such as the Brodatz textures [5], is a common practice when evaluating the effectiveness of features in characterizing texture. Such use, however, is only effective for comparing features on a common ground. The absolute effectiveness of a feature must still be judged on its effectiveness in characterizing textures which it will be measuring in a real application. The following sections describe the textures, sampled from a series of real diagnostic ultrasound images of an ovarian follicle, and the experiments performed on them in evaluating the effectiveness of features to characterize texture at various resolutions.

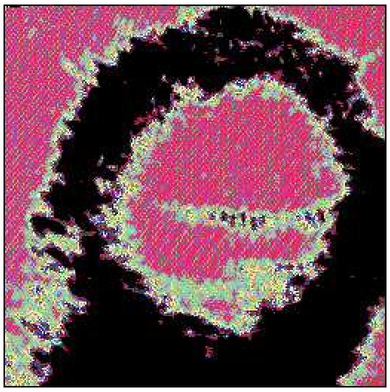
5.5.1 Texture Classes

The importance of using real images in the evaluation of features is apparent when the features are to be used in an application for the segmentation of real images. It is not sufficient to experiment with a set of synthetic textures; and, based on the results, speculate about the ability of features to characterize any given type of texture [29, 51, 16, 22]. This practice is common in most feature evaluations. Imagine the reaction to a researcher releasing a new drug to the public based on the fact that it was tested only on the rats in his lab. It seems inappropriate to assume that such a major decision could be made on such evidence. The researcher is required to test the drug rigorously, including tests on humans for which the drug is intended. Only then can claims about the effectiveness of the drug be made. Admittedly, this analogy is exaggerated, but it does illustrate the point. In texture characterization, there is not much use for a feature which can identify synthetic textures but can not identify textures in real images. Experimenting with real images should always be done to determine the usefulness of the ability of the feature to characterize texture.

The textures used in the experiments were all sampled from a series of diagnostic ultrasound images of an ovarian follicle. Figure 5.4 depicts one of the ultrasound images of an ovarian follicle from which textures were sampled. It is obvious in Figure 5.4(1) that there are two distinct regions; the lightly textured region, representing



1)



2)

Figure 5.4: An example of a real image. (1) In this ultrasound image it is obvious that there are two textures, (2) Enhancing the ultrasound image in (1) illustrates the possibility of a third texture class surrounding the lightly textured region in (1).

layers of tissue, and a darker region, representing fluid (the follicle is the dark region appearing in the center of the image). Figure 5.4(2) shows that by enhancing the ultrasound image in Figure 5.4(1), there is a possibility of a third texture surrounding the lightly textured region. As a result, three texture classes were assumed to exist in the ultrasound images. The texture classes are described as follows:

- C_1 - texture class 1: the fluid region,
- C_2 - texture class 2: a thin region separating fluid from surrounding tissue,
- C_3 - texture class 3: the surrounding tissue.

For each of the three texture classes, a *representative sample* at resolution β was drawn. A representative sample is a set of homogeneous blocks from the same texture class. The blocks were sampled from different areas but of the same texture class in the image and from different images. The number of blocks sampled is large enough to sufficiently represent texture variations within the same class of textures. For all experiments, representative samples at resolution β = 4, 8, 16, 32, 64, 128 were collected for texture classes C_1 and C_3. For C_2, it was not possible to obtain a sample with β = 128. For C_1 the sample sizes were 2368 (β=4), 2192 (β=8), 932 (β=16), 233 (β=32), 58 (β=64), and 9 (β=128). For C_3 the sample sizes were 2304 (β=4), 2176 (β=8), 928 (β=16), 232 (β=32), 58 (β=64), and 16 (β=128). For C_2 the sample sizes were 2304 (β=4), 576 (β=8), 144 (β=16), 36 (β=32), and 8 (β=64).

5.5.2 Experiment 1: Class Distinguishability

The purpose of Experiment 1 was to determine the effectiveness of the 32 features described in Section 5.1 in characterizing the three texture classes present in the ultrasound images.

Table 5.1(1) shows the percentage of outliers detected using the LMS method, for each of the 32 feature among the three texture classes $C_1, C_2,$ andC_3. The percentage of outliers was determined by adding the number of outlier measures detected for each sample from the three texture classes and dividing by the total number of blocks in the three samples. Table 5.1(2) shows the number of overlapping regions among the ranges of measures based on the LMS method, for each feature. Recall that the range of measures for a feature is due to the variability in texture for a representative sample from a given texture class. Note that for three texture classes, the maximum number of overlapping regions is 3. It can be seen from Table 5.1(2) that for most features, it is not possible to produce a range of measures for each of the three texture classes that do not result in overlapping regions.

Since both methods produced a large number of overlapping regions, it was necessary to examine the range of measures produced by each feature more closely. Table 5.2 depicts the feature space overlap graphically for each of the 32 features at different resolutions. The feature space is defined to be

$$B = [MIN(r_{l1}, r_{l2}, r_{l3}), MAX(r_{u1}, r_{u2}, r_{u3})]$$

where r_{lm} is the lower value of the range of measures for texture class m and r_{um} is the upper value of the range of measures for texture class m (see Section 5.4 for a description of r_l, r_u and the range of measures). The feature space is simply the total one dimensional space occupied by the range of measures for all three textures. The portion of the feature space occupied by the range of measures for each texture class is normalized and displayed as a bar in a cell of the table. Overlapping regions are indicated in a cell wherever two or more bars overlap vertically. The amount of vertical overlap among the bars indicates the relative amount of overlap among the ranges

Total Outliers						
	β					
Feature	4	8	16	32	64	128
1	9.2	9.5	15.6	19.2	17.7	8.7
2	7.5	13.2	22.3	26.3	21.8	26.1
3	9.8	11.2	10.9	17.2	13.7	8.7
4	11.6	12.8	14.9	15.2	15.3	8.7
5	23.9	16.8	15.2	19.0	17.7	17.4
6	13.0	14.7	18.0	17.0	12.9	13.0
7	17.1	23.8	25.9	25.0	16.9	17.4
8	9.1	6.3	10.0	10.4	9.7	17.4
9	9.1	13.2	17.4	17.8	17.7	17.4
10	29.1	5.0	4.1	4.4	4.8	8.7
11	31.1	6.8	7.2	7.4	8.1	13.0
12	11.2	16.0	18.4	21.2	19.4	17.4
13	18.8	21.0	25.0	26.1	23.4	21.7
14	22.5	29.1	32.8	33.3	29.0	26.1
15	7.2	6.1	9.4	10.6	15.3	13.0
16	8.3	15.0	19.4	18.8	16.1	0.0
17	28.2	4.1	5.2	7.2	8.1	13.0
18	26.0	5.9	7.5	7.2	5.6	8.7
19	11.0	16.6	20.3	20.2	16.9	13.0
20	18.1	20.9	26.0	19.0	21.0	17.4
21	22.0	29.8	33.9	33.7	24.2	17.4
22	2.4	4.1	7.4	9.6	9.7	26.1
23	14.1	20.6	21.0	0.4	12.9	0.0
24	14.1	20.0	19.4	0.2	14.5	17.4
25	3.5	6.0	9.6	11.8	6.5	13.0
26	6.3	4.3	4.9	8.6	11.3	8.7
27	0.0	1.4	2.8	4.0	7.3	13.0
28	0.0	28.3	3.2	8.8	12.9	13.0
29	21.9	20.1	14.3	4.6	10.5	13.0
30	24.2	29.5	23.2	9.2	17.7	13.0
31	18.7	12.9	15.6	10.0	8.9	30.4
32	13.7	17.6	16.1	15.6	8.9	8.7

(1)

Overlap Regions						
	β					
Feature	4	8	16	32	64	128
1	3	3	3	3	3	1
2	3	3	3	3	3	1
3	3	3	3	3	3	1
4	3	3	3	3	3	1
5	1	2	3	3	2	1
6	3	3	3	3	3	1
7	3	3	3	3	3	1
8	3	3	2	1	0	0
9	3	3	1	1	0	0
10	3	3	3	1	1	0
11	3	3	3	2	1	0
12	3	3	2	1	1	0
13	3	3	3	3	3	1
14	3	3	3	3	3	1
15	3	3	2	2	0	0
16	3	3	1	1	1	0
17	3	3	3	2	1	0
18	3	3	3	2	1	0
19	3	3	1	1	1	0
20	3	3	3	3	3	1
21	3	3	3	3	3	1
22	3	3	3	3	3	1
23	3	1	1	3	3	1
24	3	1	1	3	3	1
25	3	3	3	3	3	1
26	3	3	3	2	0	0
27	3	3	3	2	0	0
28	3	3	3	3	3	0
29	3	3	3	3	3	1
30	3	3	3	3	3	1
31	3	3	3	3	3	1
32	3	3	3	3	3	1

(2)

Table 5.1: LMS results for textures at original orientations; (1) total percentage of outliers for texture classes 1, 2, and 3, (2) number of overlapping regions for texture classes 1, 2, and 3 for each value of β.

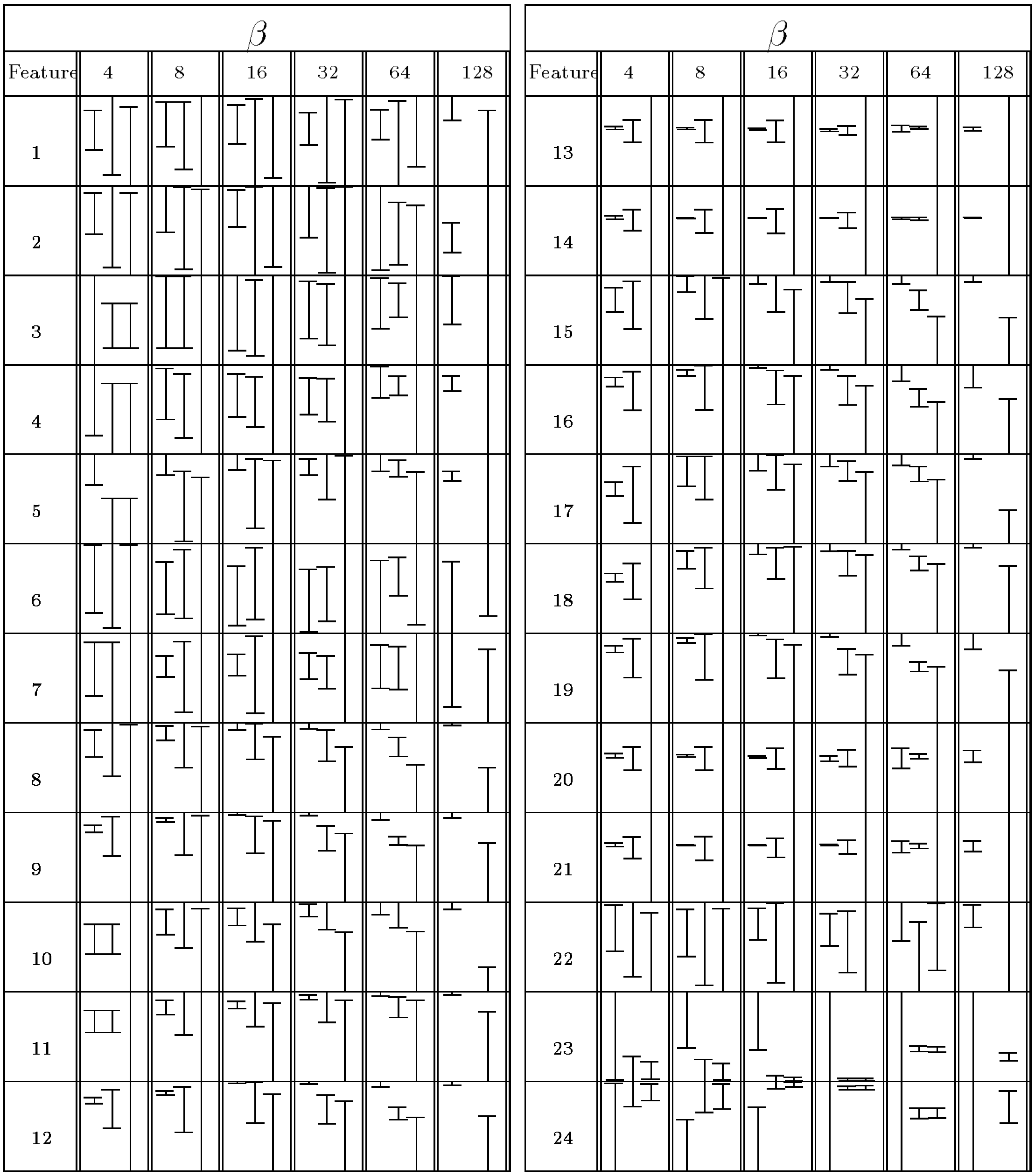


Table 5.2: Feature space overlap for the 32 features. Each cell depicts the portion of the normalized feature space occupied by the range of the feature's measures for (from left to right) texture classes 1, 2, and 3 for each value of β (*continued next page*)

β						
Feature	4	8	16	32	64	128
25						
26						
27						
28						
29						
30						
31						
32						

Table 5.2 (*continued from previous page*).

of texture class measures that each overlapping bar represents. It is obvious from Table 5.2 that the ranges of measures obtained for texture class C_2 are consistently overlapping with those in texture classes C_1 and C_3. It is apparent that texture class C_2 can not be uniquely characterized. Based on this observation, it was decided that texture class C_2 was not distinguishable from classes C_1 or C_3 and was no longer evaluated in the remaining experiments.

5.5.3 Experiment 2: Texture Orientation

The purpose of this set of experiments was to determine the effect of texture orientation on the ability of the features in uniquely characterizing texture classes C_1 and C_3. The sample blocks from texture classes C_1 and C_3 described in Section 5.5.1 were rotated by 90, 180, and 270 degrees, as well as at their original orientation. A fifth texture orientation was used in which the orientation of the sample blocks was **varied** at 0, 90, 180, and 270 degrees. The blocks within each sample were consecutively oriented at 90 degree intervals such that $\frac{1}{4}$ of the blocks were at the original orientation, $\frac{1}{4}$ of the blocks were at 90 degree orientation, $\frac{1}{4}$ of the blocks were at the 180 degree orientation and $\frac{1}{4}$ of the blocks were at 270 degree orientation. This varied orientation was used to determine the ability of the features to characterize blocks at varied orientations within the same sample. The 32 features were applied to the textures at each orientation using the LMS method to detect outliers in each representative sample.

Tables 5.3-5.5 show the results for the 32 features in characterizing the samples from texture classes C_1 and C_3 at their original orientations. Table 5.3(1) shows the number of outliers detected within each sample for texture class C_1 and Table 5.3(2) shows the outliers detected for class C_3. The total percentage of outliers is shown in Table 5.4(1). The number of overlapping regions is shown in Table 5.4(2). It is interesting to note that at each resolution β, there is at least one feature which has no overlapping regions. This is significant improvement over the results obtained in Experiment 1 in which it was impossible to uniquely characterize all three texture

Texture 1: Outliers						
	β					
Feature	4	8	16	32	64	128
1	220	237	192	63	15	2
2	269	448	326	90	16	3
3	132	235	86	39	8	2
4	268	356	156	43	11	2
5	672	732	304	83	19	4
6	488	384	169	46	12	2
7	629	757	303	79	16	2
8	246	149	150	45	8	3
9	322	528	274	73	16	3
10	704	92	19	9	4	1
11	721	138	51	17	6	2
12	384	602	301	87	18	3
13	397	308	185	56	10	2
14	579	688	330	87	18	3
15	201	143	140	46	15	2
16	275	570	317	81	14	0
17	1026	39	63	22	8	3
18	984	106	89	31	7	2
19	407	650	335	85	14	2
20	383	295	207	31	9	2
21	581	715	354	85	13	2
22	158	164	114	28	4	3
23	1	626	307	0	16	0
24	1	584	280	0	18	4
25	139	177	176	42	1	1
26	177	35	53	20	7	0
27	0	29	33	14	3	3
28	0	590	20	24	13	2
29	428	352	149	1	9	2
30	479	716	324	21	16	1
31	545	278	164	33	8	2
32	344	411	138	36	3	1
Samples	2368	2192	932	233	58	9

(1)

Texture 3: Outliers						
	β					
Feature	4	8	16	32	64	128
1	161	184	109	30	7	0
2	113	158	104	39	10	3
3	328	260	119	40	6	0
4	320	234	122	28	6	0
5	520	98	0	1	0	0
6	155	253	169	35	3	1
7	259	337	193	34	4	2
8	196	134	42	6	4	1
9	113	92	66	11	5	1
10	380	120	51	7	1	1
11	338	150	77	20	3	1
12	172	162	55	16	5	1
13	503	576	268	64	16	3
14	508	596	284	69	15	3
15	142	126	40	5	3	1
16	110	132	56	9	5	0
17	495	132	30	7	1	0
18	286	151	51	5	0	0
19	137	137	63	12	5	1
20	503	580	266	55	15	2
21	501	598	276	72	15	2
22	3	22	27	17	8	3
23	512	312	101	1	0	0
24	512	325	96	1	0	0
25	12	95	10	16	7	2
26	131	148	43	21	5	2
27	0	28	19	5	4	0
28	0	634	39	17	2	1
29	561	518	119	19	4	1
30	608	591	121	22	6	2
31	391	301	135	16	3	5
32	265	370	162	38	8	1
Samples	2304	2176	928	232	58	14

(2)

Table 5.3: Results for textures at original orientation. Number of outliers detected for (1) texture class C_1, (2) texture class C_3.

classes. The range of measures for each sample is illustrated in more detail in Table 5.5. The feature space overlap for each of the 32 features shows where each feature fails in producing a unique characterization for the representative samples.

Using the ranking scheme in Equation (5.3) in Section 5.4, the feature rankings at each resolution β are displayed in Table 5.6. The higher the position of the feature in the table, the higher is the ranking. It is interesting to note that the top feature was consistently one from the spatial domain.

The results obtained when the features were applied to the textures at 90, 180, and 270 degrees were almost identical to those obtained when the original orientation of the textures was used. Figure 5.5 shows the percentage of outliers produced for

Total Outliers						
	β					
Feature	4	8	16	32	64	128
1	8.2	9.6	16.2	20.0	19.0	8.7
2	8.2	13.9	23.1	27.7	22.4	26.1
3	9.8	11.3	11.0	17.0	12.1	8.7
4	12.6	13.5	14.9	15.3	14.7	8.7
5	25.5	19.0	16.3	18.1	16.4	17.4
6	13.8	14.6	18.2	17.4	12.9	13.0
7	19.0	25.0	26.7	24.3	17.2	17.4
8	9.5	6.5	10.3	11.0	10.3	17.4
9	9.3	14.2	18.3	18.1	18.1	17.4
10	23.2	4.9	3.8	3.4	4.3	8.7
11	22.7	6.6	6.9	8.0	7.8	13.0
12	11.9	17.5	19.1	22.2	19.8	17.4
13	19.3	20.2	24.4	25.8	22.4	21.7
14	23.3	29.4	33.0	33.5	28.4	26.1
15	7.3	6.2	9.7	11.0	15.5	13.0
16	8.2	16.1	20.1	19.4	16.4	0.0
17	32.6	3.9	5.0	6.2	7.8	13.0
18	27.2	5.9	7.5	7.7	6.0	8.7
19	11.6	18.0	21.4	20.9	16.4	13.0
20	19.0	20.0	25.4	18.5	20.7	17.4
21	23.2	30.1	33.9	33.8	24.1	17.4
22	3.4	4.3	7.6	9.7	10.3	26.1
23	11.0	21.5	21.9	0.2	13.8	0.0
24	11.0	20.8	20.2	0.2	15.5	17.4
25	3.2	6.2	10.0	12.5	6.9	13.0
26	6.6	4.2	5.2	8.8	10.3	8.7
27	0.0	1.3	2.8	4.1	6.0	13.0
28	0.0	28.0	3.2	8.8	12.9	13.0
29	21.2	19.9	14.4	4.3	11.2	13.0
30	23.3	29.9	23.9	9.2	19.0	13.0
31	20.0	13.3	16.1	10.5	9.5	30.4
32	13.0	17.9	16.1	15.9	9.5	8.7

(1)

Overlap Regions						
	β					
Feature	4	8	16	32	64	128
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	0	0	1	1	0	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	1	0	0	0	0
9	1	1	0	0	0	0
10	1	1	1	0	0	0
11	1	1	1	0	0	0
12	1	1	0	0	0	0
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	1	0	0	0	0
16	1	1	0	0	0	0
17	1	1	1	0	0	0
18	1	1	1	0	0	0
19	1	1	0	0	0	0
20	1	1	1	1	1	1
21	1	1	1	1	1	1
22	1	1	1	1	1	1
23	1	0	0	1	1	1
24	1	0	0	1	1	1
25	1	1	1	1	1	1
26	1	1	1	0	0	0
27	1	1	1	0	0	0
28	1	1	1	1	1	0
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1

(2)

Table 5.4: Results for textures at original orientation; (1) total percentage of outliers for texture classes C_1 and C_3, (2) number of overlapping regions for texture classes C_1 and C_3 for each value of β.

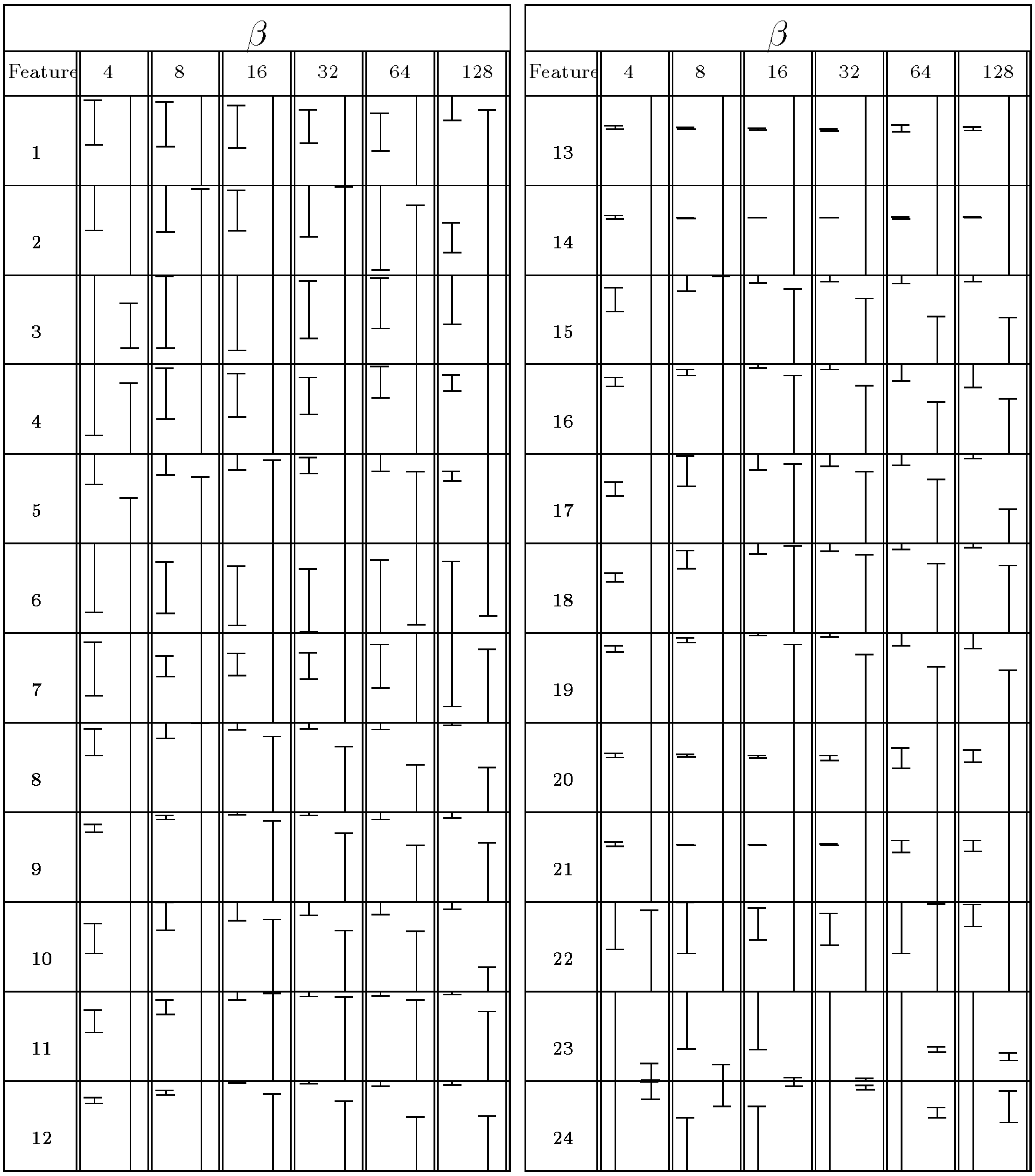


Table 5.5: Feature space overlap for the 32 features. Each cell depicts the portion of the normalized feature space occupied by the distribution of the features' measures for (from left to right) texture classes C_1 and C_3 for each value of β (*continued next page*)

	β					
Feature	4	8	16	32	64	128
25						
26						
27						
28						
29						
30						
31						
32						

Table 5.5 (*continued from previous page*).

Feature Detector Ranking						
	β					
Rank	4	8	16	32	64	128
1	5	5	15	10	10	16
2	27	24	8	27	18	10
3	28	23	9	17	27	18
4	25	27	12	18	11	26
5	22	17	16	11	17	11
6	26	26	24	26	8	15
7	15	22	19	8	26	17
8	1	10	23	15	15	19
9	2	18	27	9	5	27
10	16	15	28	16	16	28
11	9	25	10	19	19	8
12	8	8	17	12	9	9
13	3	11	26	23	12	12
14	23	1	11	24	25	23
15	24	3	18	29	31	1
16	19	31	22	28	32	3
17	12	4	25	30	22	4
18	4	2	3	22	29	32
19	32	9	29	31	3	6
20	6	6	4	25	6	25
21	20	16	31	4	28	29
22	7	12	32	32	23	30
23	13	32	1	3	4	5
24	31	19	5	6	24	7
25	29	29	6	5	7	20
26	11	20	2	20	1	21
27	21	13	30	1	30	24
28	10	7	13	7	20	13
29	14	28	20	13	2	2
30	30	14	7	2	13	14
31	18	30	14	14	21	22
32	17	21	21	21	14	31

Table 5.6: Feature ranking for textures at original orientation for each β.

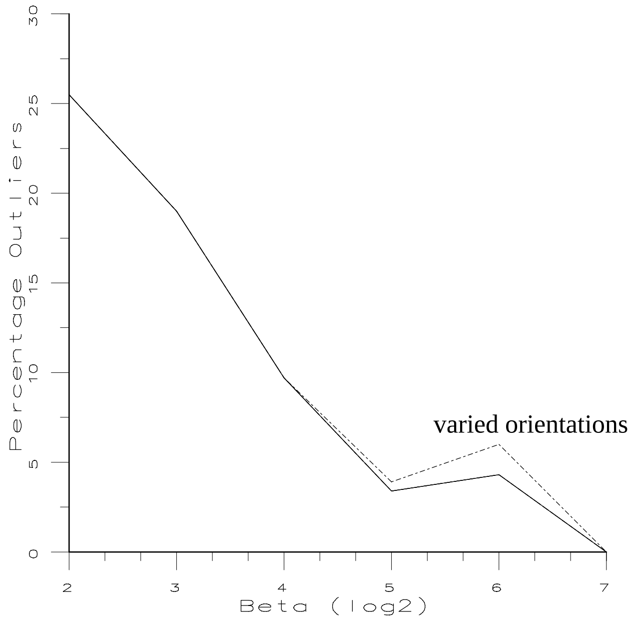


Figure 5.5: Total within class outliers for texture classes C_1 and C_3. Texture orientations at 0, 90, 180, 270 degrees produce the same outlier curve.

each orientation. The percentage of outliers is exactly the same for each orientation except when the orientation is varied (varied as discussed previously in this section). At this orientation, the percentage of outliers increases slightly for $\beta = 32$ and $\beta = 64$.

The top ranking feature for each β is shown in Figure 5.6. In all cases, the top ranking features were the same for $\beta = 4$, $\beta = 8$, $\beta = 16$, and $\beta = 128$, independent of the orientation of the textures. The exception to this was when the texture was at its original orientation (0 degrees) where feature 15, instead of feature 8, was top ranked. Closer examination of this discrepancy reveals some interesting points. First, feature 15 is based on the standard deviation of the gradient magnitude while feature 8 is based on the average deviation of the gradient magnitude (see Section 5.1 for these feature definitions). Both are similar measures. Second, Table 5.6 shows that feature 8 was ranked second only to feature 15. Examining Tables 5.4(1) and 5.4(2) shows that both features had no overlapping regions and that the total percentage of outliers differed by only .5%.

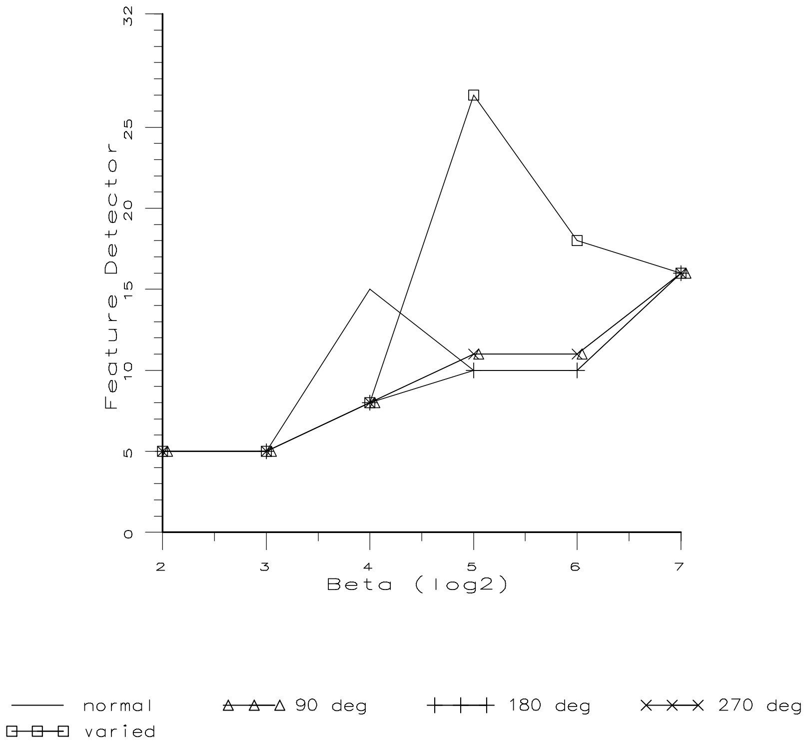


Figure 5.6: Top ranking feature for each value of β at 0, 90, 180, 270 degrees and varied orientations.

Another interesting point can be made about the feature rankings in Figure 5.6. For the 90 and 270 degree orientations, the rankings were exactly the same. For the original and 180 degree orientations, the rankings were also the same (with the noted exception above). In fact, for all of these orientations, the rankings were very similar. Based on these observations it was decided that the 32 features were fairly independent of texture orientations at 0, 90, 180, and 270 degrees. The remaining experiments did not consider these orientations.

Tables 5.7-5.9 show the results for the 32 features in characterizing the samples from texture classes C_1 and C_3 at varied orientations. Table 5.7(1) shows the number of outliers detected within each sample for texture class C_1 and Table 5.7(2) shows the outliers detected for class C_3. The total percentage of outliers is shown in Table 5.8(1). The number of overlapping regions is shown in Table 5.8(2). The range of measures for each sample is illustrated in more detail in Table 5.9. The feature space overlap for each of the 32 features shows where each feature failed to produce a unique characterization for the representative samples.

Using the ranking scheme in Equation (5.3) in Section 5.4, the feature rankings at each resolution β are displayed in Table 5.10. Note that for $\beta = 32$, the top feature is one from the frequency domain. These results show that orientation must be considered when the texture orientation is varied among the blocks in a representative sample from the same texture class. With ultrasound images, however, the textures always appear at the same orientation. This orientation may be unknown but it never varies among images. Therefore, subsequent experiments only considered the original orientation of the textures in the ultrasound images from which they were obtained.

5.5.4 Experiment 3: Texture Processing

The purpose of this set of experiments was to investigate preprocessing techniques which could decrease the number of detected outliers within each representative sample. The previous two experiments showed that the percentage of outliers for small values of β were larger than expected. This could simply mean that the proper

Texture 1: Outliers						
	β					
Feature	4	8	16	32	64	128
1	232	220	206	60	16	2
2	269	448	326	90	16	3
3	434	424	82	44	14	3
4	152	303	38	25	2	4
5	672	732	304	83	19	4
6	440	396	167	46	12	2
7	547	715	311	77	16	2
8	263	151	140	50	14	2
9	322	528	274	73	16	3
10	682	98	36	9	3	2
11	743	121	45	12	3	0
12	384	602	301	87	18	3
13	369	318	190	55	10	2
14	570	682	333	86	18	3
15	412	127	137	52	15	2
16	275	570	317	81	14	0
17	969	74	43	4	2	0
18	1041	96	43	12	2	2
19	407	650	335	85	14	2
20	418	307	199	31	9	2
21	557	696	354	85	13	2
22	158	164	114	28	4	3
23	1	581	307	0	16	0
24	1	570	281	0	18	4
25	129	162	176	42	1	1
26	131	23	54	20	7	1
27	0	18	25	15	3	1
28	0	591	25	22	12	2
29	428	356	149	1	9	2
30	464	696	324	21	16	1
31	505	237	161	34	8	2
32	329	383	130	33	3	1
Samples	2368	2192	932	233	58	9

(1)

Texture 3: Outliers						
	β					
Feature	4	8	16	32	64	128
1	151	154	117	24	6	0
2	113	158	104	39	10	3
3	324	216	171	46	11	4
4	324	260	89	12	7	2
5	520	98	0	1	0	0
6	189	241	177	38	4	2
7	254	342	176	39	4	2
8	237	126	43	10	4	1
9	113	92	66	11	5	1
10	267	147	105	31	11	4
11	304	159	44	12	4	2
12	172	162	55	16	5	1
13	501	574	267	64	16	3
14	500	605	284	69	15	3
15	179	132	49	8	5	1
16	110	132	56	9	5	0
17	165	149	89	30	12	4
18	205	150	41	9	2	2
19	137	137	63	12	5	1
20	476	577	266	56	15	2
21	506	585	277	73	15	2
22	3	22	27	17	8	3
23	505	333	90	1	0	0
24	505	348	112	1	0	0
25	24	63	9	16	7	2
26	134	136	78	27	10	1
27	0	37	15	3	4	0
28	0	597	24	12	3	1
29	539	502	125	19	4	1
30	605	575	125	22	6	2
31	400	275	118	17	3	4
32	352	322	140	37	9	2
Samples	2304	2176	928	232	58	14

(2)

Table 5.7: Results for textures at varied orientations. Number of outliers detected for (1) texture class C_1, (2) texture class C_3.

Total Outliers						
	β					
Feature	4	8	16	32	64	128
1	8.2	8.6	17.4	18.1	19.0	8.7
2	8.2	13.9	23.1	27.7	22.4	26.1
3	16.2	14.7	13.6	19.4	21.6	30.4
4	10.2	12.9	6.8	8.0	7.8	26.1
5	25.5	19.0	16.3	18.1	16.4	17.4
6	13.5	14.6	18.5	18.1	13.8	17.4
7	17.1	24.2	26.2	24.9	17.2	17.4
8	10.7	6.3	9.8	12.9	15.5	13.0
9	9.3	14.2	18.3	18.1	18.1	17.4
10	20.3	5.6	7.6	8.6	12.1	26.1
11	22.4	6.4	4.8	5.2	6.0	8.7
12	11.9	17.5	19.1	22.2	19.8	17.4
13	18.6	20.4	24.6	25.6	22.4	21.7
14	22.9	29.5	33.2	33.3	28.4	26.1
15	12.6	5.9	10.0	12.9	17.2	13.0
16	8.2	16.1	20.1	19.4	16.4	0.0
17	24.3	5.1	7.1	7.3	12.1	17.4
18	26.7	5.6	4.5	4.5	3.4	17.4
19	11.6	18.0	21.4	20.9	16.4	13.0
20	19.1	20.2	25.0	18.7	20.7	17.4
21	22.8	29.3	33.9	34.0	24.1	17.4
22	3.4	4.3	7.6	9.7	10.3	26.1
23	10.8	20.9	21.3	0.2	13.8	0.0
24	10.8	21.0	21.1	0.2	15.5	17.4
25	3.3	5.2	9.9	12.5	6.9	13.0
26	5.7	3.6	7.1	10.1	14.7	8.7
27	0.0	1.3	2.2	3.9	6.0	4.3
28	0.0	27.2	2.6	7.3	12.9	13.0
29	20.7	19.6	14.7	4.3	11.2	13.0
30	22.9	29.1	24.1	9.2	19.0	13.0
31	19.4	11.7	15.0	11.0	9.5	26.1
32	14.6	16.1	14.5	15.1	10.3	13.0

(1)

Overlap Regions						
	β					
Feature	4	8	16	32	64	128
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	0	0	1	1	0	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	1	0	0	0	0
9	1	1	0	0	0	0
10	1	1	1	0	0	0
11	1	1	1	0	0	0
12	1	1	0	0	0	0
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	1	0	0	0	0
16	1	1	0	0	0	0
17	1	1	1	1	0	0
18	1	1	1	1	0	0
19	1	1	0	0	0	0
20	1	1	1	1	1	1
21	1	1	1	1	1	1
22	1	1	1	1	1	1
23	1	0	0	1	1	1
24	1	0	0	1	1	1
25	1	1	1	1	1	1
26	1	1	1	1	0	0
27	1	1	1	0	0	0
28	1	1	1	1	1	0
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1

(2)

Table 5.8: Results for textures at varied orientations; (1) total percentage of outliers for texture classes C_1 and C_3, (2) number of overlapping regions for texture classes C_1 and C_3 for each value of β.

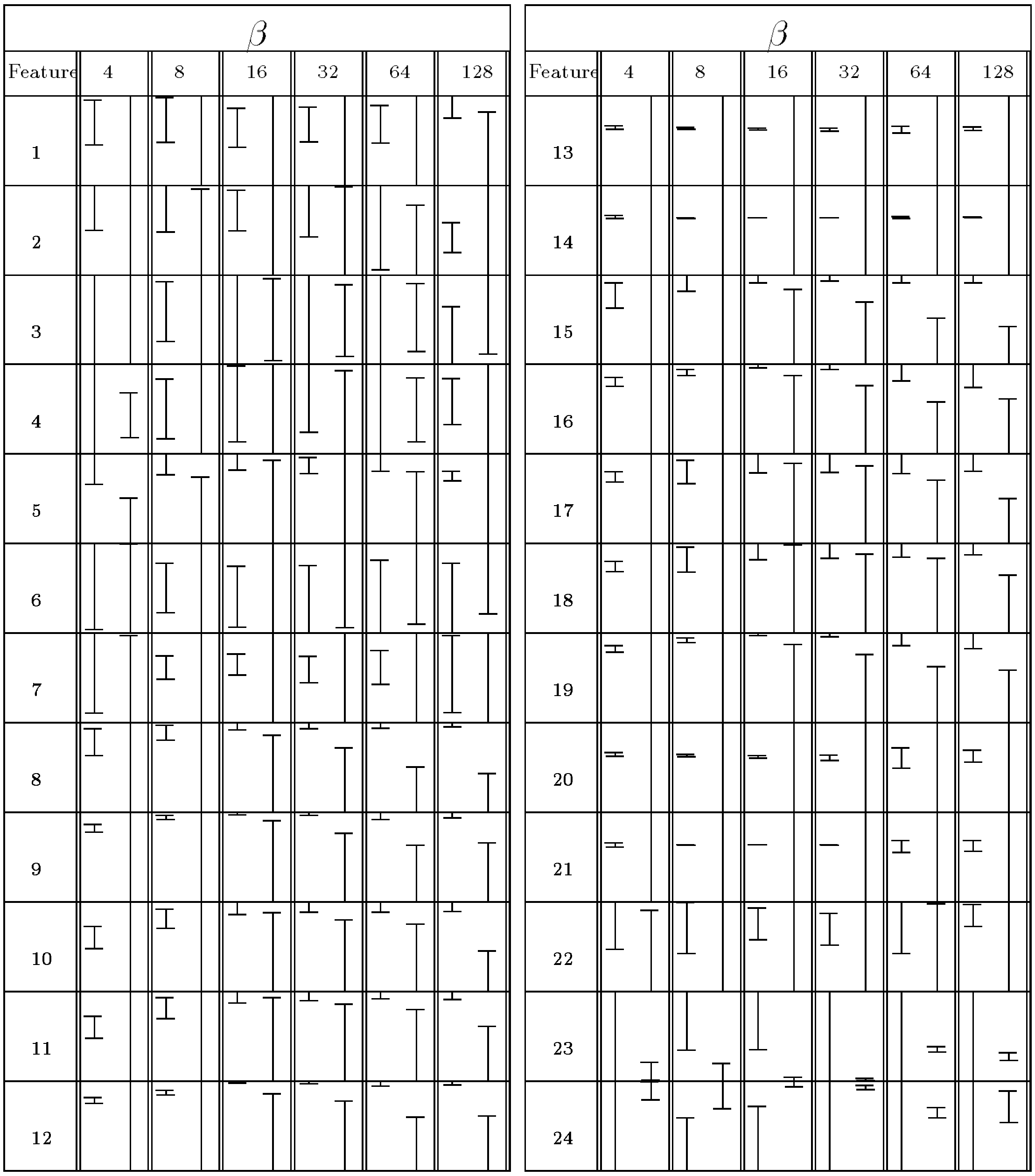


Table 5.9: Feature space overlap for the 32 features. Each cell depicts the portion of the normalized feature space occupied by the distribution of the features' measures for texture classes C_1 and C_3 (from left to right) for each value of β (*continued next page*)

β						
Feature	4	8	16	32	64	128
25						
26						
27						
28						
29						
30						
31						
32						

Table 5.9 (*continued from previous page*).

Feature Detector Ranking						
	β					
Rank	4	8	16	32	64	128
1	5	5	8	27	18	16
2	27	23	15	11	11	27
3	28	24	9	10	27	11
4	25	27	12	8	10	26
5	22	26	16	15	17	8
6	26	22	24	9	26	15
7	2	17	23	16	8	19
8	1	25	19	19	5	28
9	16	10	27	12	16	9
10	9	18	28	23	19	12
11	4	15	18	24	15	17
12	8	8	11	29	9	18
13	23	11	4	18	12	10
14	24	1	17	17	25	23
15	19	31	26	28	4	1
16	12	4	10	4	31	25
17	15	2	22	30	22	29
18	6	9	25	22	32	30
19	32	6	3	26	29	32
20	3	3	32	31	28	5
21	7	16	29	25	6	6
22	13	32	31	32	23	7
23	20	12	5	1	24	20
24	31	19	1	5	7	21
25	10	29	6	6	1	24
26	29	20	2	20	30	13
27	11	13	30	3	20	2
28	21	7	13	7	3	4
29	30	28	20	13	2	14
30	14	30	7	2	13	22
31	17	21	14	14	21	31
32	18	14	21	21	14	3

Table 5.10: Feature rankings for textures at varied orientations for each β.

features were not being used to characterize the ultrasound textures. Such a problem could be easily solved by incorporating more features into the characterization framework. Another way to approach the problem is to process each block of the representative samples with some operators and in so doing, modifying the characteristics of each texture class. Thus, the characteristics of each texture class are modified to suit the features, rather than using more features to try to characterize the textures in their original form. For this set of experiments it was decided to perform the later of the two approaches.

Two methods were used in processing the blocks. The first method involved stretching a range of pixel intensities so that subtle intensity changes would become more apparent. For each block, the pixel intensities were stretched between a lower, l, and an upper, u, value by mapping the lower value to 0 and the upper value to 255. Intensities in between these values were interpolated linearly. The second, and less *ad hoc*, method is to convolve each block with the Sobel edge operator.

Tables 5.11-5.14 show the results of the 32 features when applied to the processed representative samples from texture classes C_1 and C_3. To obtain these results, the pixel intensities were stretched with l=20 and u=40. A number of other l,u combinations were evaluated. These included l=0 u=60, l=0 u=100, l=20 u=60, and l=20 u=100. Their results on the percentage of outliers detected are summarized in Figure 5.7. Table 5.11 shows the number of outliers detected within each sample, for texture classes C_1 and C_3. There is a substantial decrease in the number of outliers for $\beta = 4$ and $\beta = 8$, especially for samples in texture class C_3. Table 5.12(1) shows the total percentage of outliers detected among the texture classes. It is apparent from this table that the percentage of outliers decreased for almost all of the 32 features. The number of overlapping regions also decreased in Table 5.12(2). This implies that more features are able to uniquely characterize the samples from texture classes C_1 and C_3. The relationship between the ranges of feature measures for texture classes C_1 and C_3 is shown in Table 5.13. Note the large amount of separability among many of the ranges. It is especially interesting to note that this occurs quite frequently for the frequency domain features f_{23}, f_{25}, f_{26}, f_{27}, and f_{28}. This increased performance

Texture 1: Outliers						
	β					
Feature	4	8	16	32	64	128
1	254	257	163	35	6	0
2	234	132	9	21	7	3
3	484	224	53	3	0	0
4	288	318	84	9	3	0
5	672	732	304	83	17	2
6	456	359	130	33	7	0
7	595	565	181	47	8	0
8	243	148	137	42	8	4
9	240	471	283	77	14	3
10	193	66	58	24	8	4
11	182	66	60	23	5	2
12	368	584	299	85	17	3
13	429	364	145	37	9	2
14	534	692	326	82	16	3
15	358	121	132	43	10	4
16	377	651	337	89	2	2
17	222	57	55	28	7	4
18	277	114	96	32	6	2
19	393	638	332	86	11	2
20	431	353	150	38	7	2
21	597	739	349	84	11	2
22	161	176	71	11	1	0
23	5	620	307	0	18	0
24	1	580	279	0	17	2
25	143	173	183	43	1	0
26	149	43	50	8	1	2
27	0	31	35	5	0	0
28	0	577	22	21	7	2
29	127	353	150	1	9	0
30	155	641	303	22	16	0
31	142	22	8	3	0	3
32	134	68	27	7	0	0
Samples	2368	2192	932	233	58	9

(1)

Texture 3: Outliers						
	β					
Feature	4	8	16	32	64	128
1	5	83	39	12	1	0
2	5	135	98	56	21	0
3	4	86	39	12	1	0
4	4	85	40	12	1	0
5	3	82	35	11	1	0
6	5	135	98	56	22	0
7	5	135	98	56	22	0
8	5	115	87	50	26	2
9	5	135	89	50	24	3
10	4	116	86	45	22	3
11	4	115	91	46	23	2
12	5	135	98	51	24	3
13	5	134	98	56	26	1
14	5	135	92	48	23	3
15	5	115	87	52	27	0
16	5	135	98	56	24	1
17	4	116	90	53	26	1
18	4	115	93	56	27	0
19	5	135	98	56	24	1
20	5	135	98	56	19	1
21	5	135	98	56	26	4
22	3	65	23	7	0	0
23	5	135	98	56	0	5
24	4	108	63	33	16	3
25	5	135	98	56	27	2
26	5	135	98	56	27	0
27	0	135	98	56	27	2
28	0	122	98	56	27	2
29	5	135	98	56	1	2
30	5	135	98	56	27	4
31	5	135	98	56	22	3
32	5	135	98	56	10	2
Samples	2304	2176	928	232	58	14

(2)

Table 5.11: Results for stretched textures with l=20 and u=40. Number of outliers detected for (1) texture class C_1, (2) texture class C_3.

Total Outliers						
	β					
Feature	4	8	16	32	64	128
1	5.5	7.8	10.9	10.1	6.0	0.0
2	5.1	6.1	5.8	16.6	24.1	13.0
3	10.4	7.1	4.9	3.2	0.9	0.0
4	6.2	9.2	6.7	4.5	3.4	0.0
5	14.4	18.6	18.2	20.2	15.5	8.7
6	9.9	11.3	12.3	19.1	25.0	0.0
7	12.8	16.0	15.0	22.2	25.9	0.0
8	5.3	6.0	12.0	19.8	29.3	26.1
9	5.2	13.9	20.0	27.3	32.8	26.1
10	4.2	4.2	7.7	14.8	25.9	30.4
11	4.0	4.1	8.1	14.8	24.1	17.4
12	8.0	16.5	21.3	29.2	35.3	26.1
13	9.3	11.4	13.1	20.0	30.2	13.0
14	11.5	18.9	22.5	28.0	33.6	26.1
15	7.8	5.4	11.8	20.4	31.9	17.4
16	8.2	18.0	23.4	31.2	22.4	13.0
17	4.8	4.0	7.8	17.4	28.4	21.7
18	6.0	5.2	10.2	18.9	28.4	8.7
19	8.5	17.7	23.1	30.5	30.2	13.0
20	9.3	11.2	13.3	20.2	22.4	13.0
21	12.9	20.0	24.0	30.1	31.9	26.1
22	3.5	5.5	5.1	3.9	0.9	0.0
23	0.2	17.3	21.8	12.0	15.5	21.7
24	0.1	15.8	18.4	7.1	28.4	21.7
25	3.2	7.1	15.1	21.3	24.1	8.7
26	3.3	4.1	8.0	13.8	24.1	8.7
27	0.0	3.8	7.2	13.1	23.3	8.7
28	0.0	16.0	6.5	16.6	29.3	17.4
29	2.8	11.2	13.3	12.3	8.6	8.7
30	3.4	17.8	21.6	16.8	37.1	17.4
31	3.1	3.6	5.7	12.7	19.0	26.1
32	3.0	4.6	6.7	13.5	8.6	8.7

(1)

Overlap Regions						
	β					
Feature	4	8	16	32	64	128
1	1	1	1	1	1	1
2	0	0	1	0	0	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1
6	1	1	1	1	1	0
7	1	1	1	1	1	0
8	1	0	0	0	0	0
9	1	1	0	1	1	1
10	1	0	0	0	0	0
11	1	0	0	0	0	0
12	0	0	0	0	0	0
13	1	1	1	1	1	1
14	1	1	1	1	1	1
15	1	0	0	0	0	1
16	0	0	0	0	1	1
17	1	0	0	0	0	0
18	1	0	0	0	0	0
19	0	0	0	0	1	1
20	1	1	1	1	1	1
21	1	1	1	1	1	1
22	1	1	1	1	1	1
23	0	1	1	1	1	1
24	1	0	0	1	1	1
25	0	0	0	0	1	1
26	0	0	0	0	1	1
27	1	0	0	0	0	0
28	1	0	0	0	1	0
29	1	1	1	1	0	1
30	1	1	1	1	1	1
31	1	1	1	1	0	1
32	1	1	1	1	0	1

(2)

Table 5.12: Results for stretched textures with l=20 and u=40; (1) total percentage of outliers for texture classes C_1 and C_3, (2) number of overlapping regions for texture classes C_1 and C_3 for each value of β.

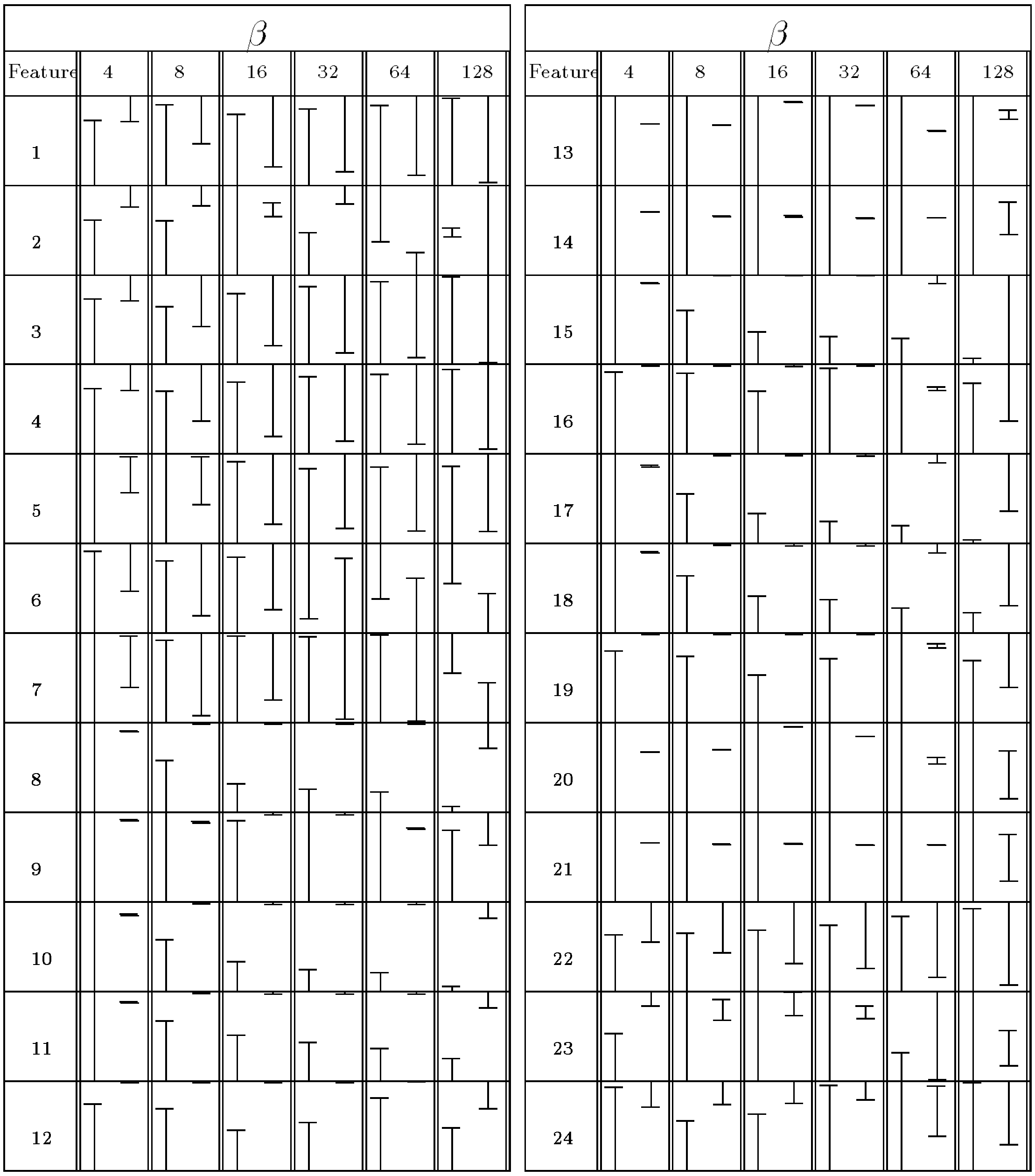


Table 5.13: Feature space overlap for the 32 features. Each cell depicts the portion of the normalized feature space occupied by the distribution of the features' measures for (from left to right) texture classes C_1 and C_3 for each value of β (*continued next page*)

β						
Feature	4	8	16	32	64	128
25						
26						
27						
28						
29						
30						
31						
32						

Table 5.13 (*continued from previous page*).

Feature Detector Ranking						
	β					
Rank	4	8	16	32	64	128
1	23	27	28	27	29	6
2	25	17	27	26	32	7
3	26	26	10	10	31	18
4	2	11	17	11	27	27
5	12	10	26	2	2	11
6	16	18	11	28	11	28
7	19	15	18	17	10	17
8	27	8	15	18	17	8
9	28	2	8	8	18	12
10	24	25	25	15	8	10
11	29	24	24	25	15	1
12	32	28	9	12	12	3
13	31	12	12	19	3	4
14	30	19	19	16	22	22
15	22	16	16	3	4	5
16	11	31	3	22	1	25
17	10	32	22	4	5	26
18	17	22	31	24	23	29
19	9	3	2	1	16	32
20	8	1	4	23	20	2
21	1	4	32	29	25	13
22	18	20	1	31	26	16
23	4	29	6	32	6	19
24	15	6	13	30	7	20
25	13	13	20	6	24	15
26	20	9	29	13	28	30
27	6	7	7	5	13	23
28	3	23	5	20	19	24
29	14	30	30	7	21	9
30	7	5	23	9	9	14
31	21	14	14	14	14	21
32	5	21	21	21	30	31

Table 5.14: Feature ranking for stretched textures with l=20 and u=40 for each β.

in the frequency domain features resulted in their dominance at the top of the feature rankings in Table 5.14. Recall that in the previous two experiments the spatial domain features were consistently top ranked.

Tables 5.15-5.18 show the results for the 32 features when applied to the processed representative samples from texture classes C_1 and C_3. To obtain these results, the blocks in the representative samples were convolved with the Sobel edge operator. The representative sample sizes used were 1024 ($\beta = 4$), 512 ($\beta = 8$), 128 ($\beta = 16$), 32 ($\beta = 32$), 36 ($\beta = 64$), and 9 ($\beta = 128$).

Table 5.15 shows the number of outliers detected within each sample for texture classes C_1 and C_3. Table 5.16(1) shows the total percentage of outliers detected among the texture classes. The number of overlapping regions is shown in Table 5.16(2). For the frequency domain features there was only one resolution ($\beta = 128$) at which features f_{27} and f_{28} produced non-overlapping regions. Examining the feature space occupied by the ranges of measures for the frequency domain features in Table 5.17 reveals that this overlap is quite substantial for most these features. As a result, the feature rankings in Table 5.18 indicate that the spatial domain features are more effective in characterizing the texture classes C_1 and C_3 when processing the representative samples with the Sobel edge operator.

A comparison of the ability of the 32 features to characterize texture classes C_1 and C_3 when processed by different methods reveals a number of points. First, examination of the total percentage of outliers for the top ranking features at each resolution β in Figure 5.7(2) indicates that with both methods the percentage of outliers for small values of β (4 and 8) was decreased. Both the stretched and convolved samples had substantially less outliers than the original samples. However, for $\beta = 32$, the top ranking features for the stretched and convolved samples resulted in a higher percentage of outliers. The stretched samples $\beta = 64$ also produced a higher percentage of outliers.

Second, the percentage of outliers detected in the stretched samples of Figure 5.7(1), in general, did not result in much improvement over the original. For some values of β the percentage of outliers using the original samples was less while for

Texture 1: Outliers						
	β					
Feature	4	8	16	32	64	128
1	121	44	12	2	0	0
2	126	72	14	4	9	0
3	283	44	8	2	0	0
4	349	38	4	2	0	0
5	150	82	20	4	0	0
6	214	32	6	2	0	0
7	238	38	10	2	0	0
8	76	46	20	6	9	0
9	94	74	24	10	9	0
10	102	46	14	8	0	0
11	103	38	8	4	0	0
12	111	78	28	10	9	0
13	143	70	12	4	9	0
14	207	128	30	6	9	0
15	71	54	20	6	9	0
16	93	74	30	10	9	0
17	91	46	20	6	0	0
18	82	36	8	6	9	0
19	104	78	28	10	9	0
20	163	76	10	4	9	0
21	208	140	36	8	9	0
22	105	30	4	0	0	0
23	41	26	6	6	9	0
24	41	10	10	6	0	0
25	34	18	6	4	0	0
26	36	32	4	0	0	0
27	0	16	4	0	0	0
28	0	136	2	0	0	0
29	107	26	6	4	0	0
30	118	30	6	6	9	0
31	29	6	0	0	0	0
32	84	2	0	0	0	0
Samples	1024	512	128	32	36	9

(1)

Texture 3: Outliers						
	β					
Feature	4	8	16	32	64	128
1	64	20	20	4	0	0
2	96	78	14	2	0	0
3	52	28	2	4	0	0
4	60	20	14	6	9	0
5	124	56	8	2	0	0
6	104	36	4	2	0	0
7	155	52	8	2	0	0
8	85	36	12	4	0	0
9	8	44	4	8	9	0
10	269	0	2	2	0	0
11	173	2	16	2	0	0
12	5	10	14	2	0	0
13	146	102	24	8	0	0
14	230	120	32	6	9	0
15	81	80	14	2	0	0
16	26	44	2	8	9	0
17	269	0	12	2	9	0
18	165	4	12	2	0	0
19	5	18	10	2	0	0
20	162	106	22	8	0	0
21	231	134	30	4	9	0
22	84	44	4	0	0	0
23	102	0	8	2	0	0
24	102	2	6	2	0	0
25	60	22	6	0	0	0
26	82	22	0	0	0	0
27	0	14	10	0	0	0
28	0	92	14	0	0	0
29	71	12	0	0	0	0
30	98	12	4	0	0	0
31	39	0	0	0	0	0
32	99	0	0	0	0	0
Samples	1024	512	128	32	36	9

(2)

Table 5.15: Results for textures convolved with the Sobel operator. Number of outliers detected for (1) texture class C_1, (2) texture class C_3.

Total Outliers						
	β					
Feature	4	8	16	32	64	128
1	9.0	6.2	12.5	9.4	0.0	0.0
2	10.8	14.6	10.9	9.4	12.5	0.0
3	16.4	7.0	3.9	9.4	0.0	0.0
4	20.0	5.7	7.0	12.5	12.5	0.0
5	13.4	13.5	10.9	9.4	0.0	0.0
6	15.5	6.6	3.9	6.2	0.0	0.0
7	19.2	8.8	7.0	6.2	0.0	0.0
8	7.9	8.0	12.5	15.6	12.5	0.0
9	5.0	11.5	10.9	28.1	25.0	0.0
10	18.1	4.5	6.2	15.6	0.0	0.0
11	13.5	3.9	9.4	9.4	0.0	0.0
12	5.7	8.6	16.4	18.8	12.5	0.0
13	14.1	16.8	14.1	18.8	12.5	0.0
14	21.3	24.2	24.2	18.8	25.0	0.0
15	7.4	13.1	13.3	12.5	12.5	0.0
16	5.8	11.5	12.5	28.1	25.0	0.0
17	17.6	4.5	12.5	12.5	12.5	0.0
18	12.1	3.9	7.8	12.5	12.5	0.0
19	5.3	9.4	14.8	18.8	12.5	0.0
20	15.9	17.8	12.5	18.8	12.5	0.0
21	21.4	26.8	25.8	18.8	25.0	0.0
22	9.2	7.2	3.1	0.0	0.0	0.0
23	7.0	2.5	5.5	12.5	12.5	0.0
24	7.0	1.2	6.2	12.5	0.0	0.0
25	4.6	3.9	4.7	6.2	0.0	0.0
26	5.8	5.3	1.6	0.0	0.0	0.0
27	0.0	2.9	5.5	0.0	0.0	0.0
28	0.0	22.3	6.2	0.0	0.0	0.0
29	8.7	3.7	2.3	6.2	0.0	0.0
30	10.5	4.1	3.9	9.4	12.5	0.0
31	3.3	0.6	0.0	0.0	0.0	0.0
32	8.9	0.2	0.0	0.0	0.0	0.0

(1)

Overlap Regions						
	β					
Feature	4	8	16	32	64	128
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	0	0	0	0	0
9	0	0	0	0	0	0
10	1	1	0	0	0	0
11	1	1	0	0	0	0
12	1	0	0	0	0	0
13	0	1	1	1	1	1
14	0	1	1	1	1	1
15	1	0	0	0	0	0
16	0	0	0	0	0	0
17	1	1	0	0	0	0
18	1	0	0	0	0	0
19	0	0	0	0	0	0
20	0	1	1	1	1	1
21	0	1	1	1	1	1
22	1	1	1	1	1	1
23	1	1	1	1	1	0
24	1	1	1	1	1	1
25	1	1	1	1	1	1
26	1	1	1	1	1	0
27	1	1	1	1	1	0
28	1	1	1	1	1	1
29	1	1	1	1	1	1
30	1	1	1	1	1	1
31	1	1	1	1	1	1
32	1	1	1	1	1	1

(2)

Table 5.16: Results for textures convolved with the Sobel operator; (1) total percentage of outliers for texture classes C_1 and C_3, (2) number of overlapping regions for texture classes C_1 and C_3 for each value of β.

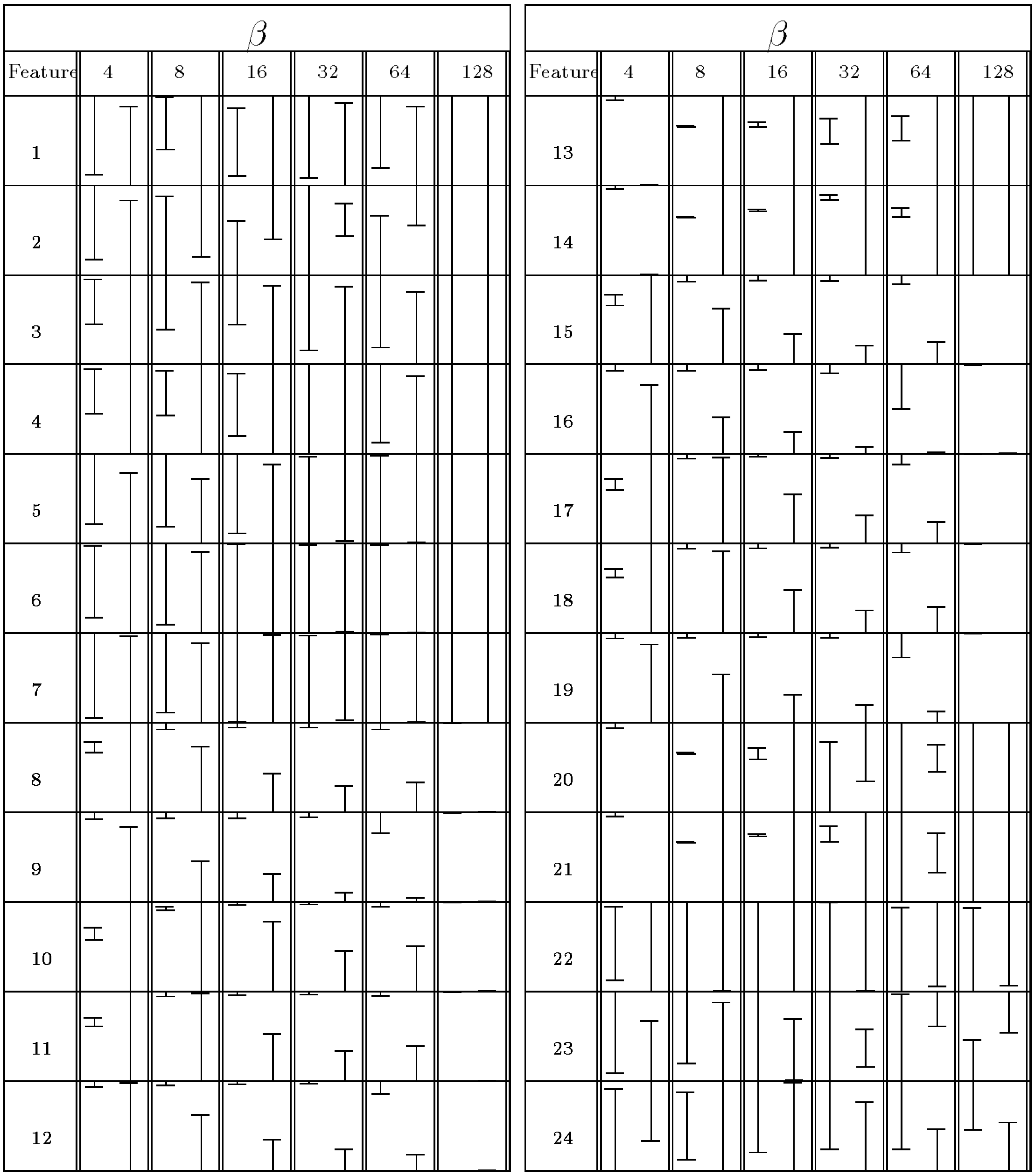


Table 5.17: Feature space overlap for the 32 features. Each cell depicts the portion of the normalized feature space occupied by the distribution of the features' measures for (from left to right) texture classes C_1 and C_3 for each value of β (*continued next page*)

	β					
Feature	4	8	16	32	64	128
25						
26						
27						
28						
29						
30						
31						
32						

Table 5.17 (*continued from previous page*).

Feature Detector Ranking						
	β					
Rank	4	8	16	32	64	128
1	9	18	10	11	10	8
2	19	8	18	15	11	9
3	16	12	11	17	8	10
4	13	19	9	18	12	11
5	20	9	8	8	15	12
6	14	16	16	10	17	15
7	21	15	17	12	18	16
8	27	32	15	19	19	17
9	28	31	19	9	9	18
10	31	24	12	16	16	19
11	25	23	31	22	1	23
12	12	27	32	26	3	26
13	26	29	26	27	5	27
14	23	11	29	28	6	1
15	24	25	22	31	7	2
16	15	30	3	32	22	3
17	8	10	6	6	24	4
18	29	17	30	7	25	5
19	32	26	25	25	26	6
20	1	4	23	29	27	7
21	22	1	27	1	28	13
22	30	6	24	2	29	14
23	2	3	28	3	31	20
24	18	22	4	5	32	21
25	5	7	7	30	2	22
26	11	5	2	4	4	24
27	6	2	5	23	13	25
28	3	13	1	24	20	28
29	17	20	20	13	23	29
30	10	28	13	14	30	30
31	7	14	14	20	14	31
32	4	21	21	21	21	32

Table 5.18: Feature ranking for textures convolved with the Sobel operator for each β.

other values of β the percentage of outliers in the stretched samples was less.

Finally, the top ranking features for the original, stretched (l=20,u=40), and the convolved samples in Figure 5.8 indicate that the frequency domain features are capable of uniquely characterizing textures classes C_1 and C_3 better than the spatial domain features when the representative samples are stretched with l=20 and u=40. In their original form, and when convolved with the Sobel operator, the spatial domain features characterize texture classes more effectively.

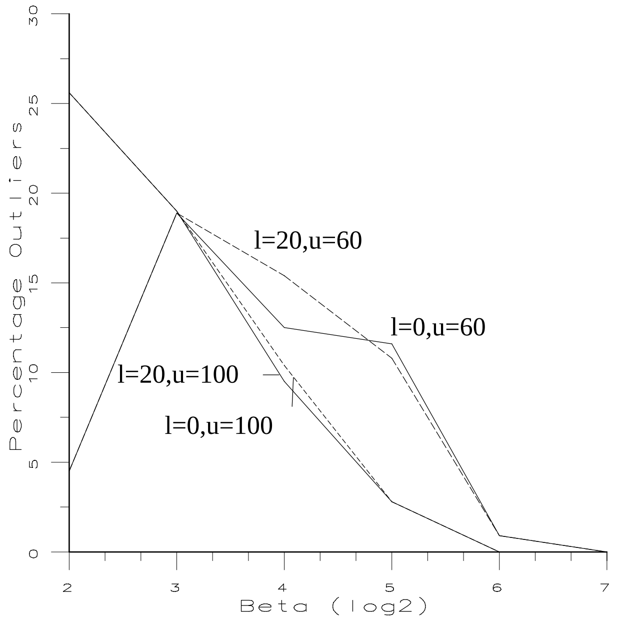
5.6 Summary

In Section 5.1 the use of features in characterizing texture was discussed. 32 spatial and frequency domain features were defined. These features were based on both the raw pixel intensities of the textures as well as transformations of the raw intensities. The four transformations used, in addition to the raw intensities, were the gradient magnitude, the residual, the horizontal directional residual and the vertical directional residual.

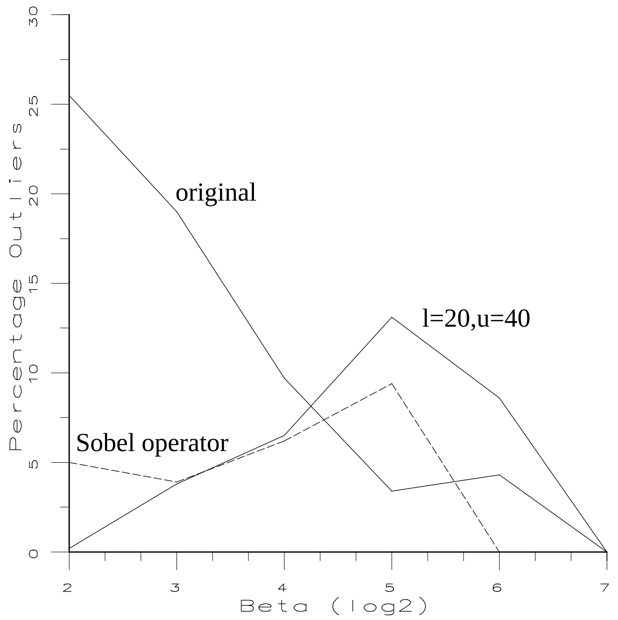
The relationship between features and the size (resolution) of the texture blocks that they are applied to is examined in Section 5.2. Because of the inherent variability within the same types of texture in most real images the MTS algorithm uses only the features which can best characterize texture classes at each different resolution, rather than trying to find an optimal feature independent of the resolution. As well, the use of outlier detection, rather than texture classification, was discussed as a method of determining the effectiveness of features in characterizing different texture classes.

In Section 5.3 the use of robust statistics in detecting outliers was described. An effective feature is one which produces a low number of outliers from the measures obtained by the feature when it is applied to a set of homogeneous blocks from the same texture class. The Least Median of Squares (LMS) was a method described to detect outliers. This statistically robust method can detect outliers in a data set which is corrupted with up to 50% outliers.

Section 5.4 described how the large amount of feature measures obtained during



(1)



(2)

Figure 5.7: Total within class outliers for texture classes C_1 and C_3 for (1) stretched textures with l=0 u=60, l=0 u=100, l=20 u=60, and l=20 u=100, (2) original textures, stretched textures with l=20 u=40 and original textures convolved with the Sobel operator.

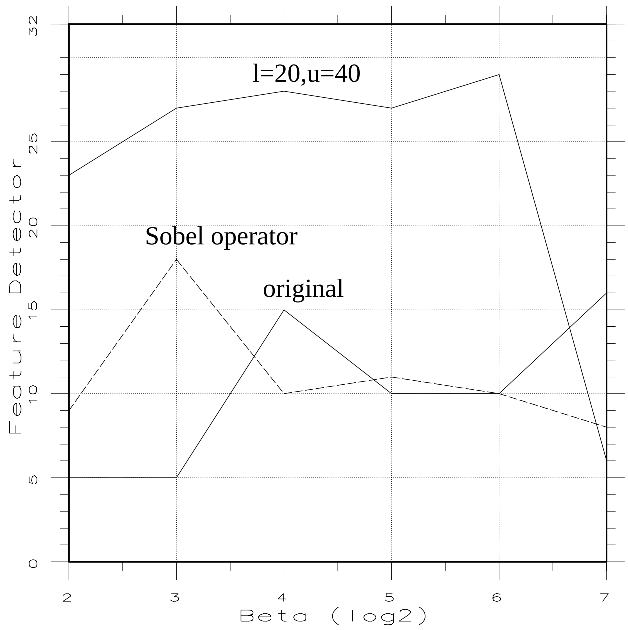


Figure 5.8: Top ranking features for each value of β for original, stretched (l=20, u=40) and convolved textures.

the characterization process can be reduced to a form which could be effectively used by the segmentation component of the MTS algorithm. This reduced form was summarized in the feature table. A ranking scheme was presented which ranks the features according to the number of outliers it produces as well as the number of overlapping regions it produces in the range of feature measures for each texture class. At each resolution each feature is ranked and recorded in the feature table. The feature table is then used by the segmentation component of the MTS algorithm to determine the appropriate feature to apply to a block at a given resolution.

The purpose of Experiment 1 was to determine the number of distinct texture classes present in a series of diagnostic ultrasound images of an ovarian follicle. Using the LMS method to detect outliers it was found that the number of overlapping regions produced with the 32 features indicated that it was not possible to uniquely characterize the three textures believed to exist in the ultrasound images. Closer examination of the overlapping regions showed that the texture class C_2 was consistently overlapping with the other texture classes, C_1 and C_3. Based on this, it was determined that texture class C_2 was not a distinct texture class in the ultrasound images, so the two texture classes, C_1 and C_3 were used in further experimentation.

In Experiment 2 , texture orientation was examined to determine the effects on the features in characterizing texture classes C_1 and C_3. Orientations of 0, 90, 180, and 270 degrees produced similar results, indicating that the 32 features were fairly independent of the four orientations. When the orientation of the blocks in a sample was varied among these four orientations the results indicated that there was an effect on the resulting feature measures. However, since the texture orientation among the ultrasound images is constant, it was not deemed necessary to consider the effects of orientation on the resulting feature rankings for texture classes C_1 and C_3.

Experiment 3 used two methods to process the representative samples; stretching the pixel intensities within a certain range and convolving every block in the samples with the Sobel edge operator. This was performed in order to reduce the number of detected outliers for the 32 features at small sample resolutions. It was found that by processing the representative samples, thus modifying the characteristics of the texture classes C_1 and C_3, the total percentage of outliers detected was reduced. In addition, the number of overlapping regions was decreased. This shows that the features could uniquely characterize the texture classes C_1 and C_3 effectively at all resolutions.

Chapter 6

Experiments

This chapter details the experimental method used to evaluate the MTS and CO algorithms and discusses the results obtained with these algorithms. Section 6.1 identifies the control parameters which affect the MTS and CO algorithms as well as the control parameters determining the types of synthetic test images used in the experiments. Section 6.2 describes the experimental environment in which the MTS and CO algorithms were evaluated. Section 6.3 evaluates the performance of the MTS algorithm while Section 6.4 compares the performance of the MTS and CO algorithms. A summary of the chapter is discussed in Section 6.5.

6.1 Identification of Control Parameters

Control parameters are those variables in an algorithm which affect the ability of the algorithm to segment an image. The CO algorithm has a number of control parameters to consider. They are summarized in the following list:

1. β - the block size used to sample texture,

2. σ - the label lattice resolution,

3. η - the neighborhood size,

4. μ - site visitation schedule,

5. λ - the penalty schedule,

6. ρ - number of labels.

7. c_i, the thresholds.

β affects how much detail in the image is detected as well as how much computation is needed at each site in the label lattice. As the block size increases the detail level decreases and the amount of computation increases. σ controls the resolution of the label lattice and thus determines how many label sites are used. As σ increases, the distance between label sites increases, resulting in fewer label sites and less (if any) overlap of adjacent blocks. The amount of computation is also affected by σ. When σ is small there is a large number of label sites. This requires more computation because there are more sites to compute the texture measures. η controls how many sites are involved in assigning a label to a particular label site. The neighborhood also determines which sites in the lattice are used when calculating the KS measure. A neighborhood does not have to consist of physically adjacent sites in the lattice. In fact, the neighborhood in the CO algorithm consists of randomly selected sites. This is done to achieve a more global estimate of texture in the image.

μ and λ are schedules which affect how sites are visited and how much of an effect illegal label formations have on the labeling process, respectively. The site visitation schedule determines the order in which the label sites have their labels evaluated at each iteration. This schedule could call for some regular pattern of site visitation or it could simply visit sites randomly. The penalty schedule λ determines how quickly constraints are applied to the minimization function. The faster the schedule, the sooner illegal label formations are ruled out. However, if λ is increased too quickly, it is possible to get stuck in a local minimum. This will result in a segmentation which does not accurately reflect the regions existing in the image.

ρ determines how many different regions can exist in the image. A small number of labels will keep the amount of computation per label site lower. The number of differentiable regions will also be low. This is advantageous in images with a small number of differentiable patterns but results in loss of detail if different regions have to be given the same label.

The thresholds, $\{c_i\}$ are perhaps the most important parameters. They control the sensitivity of the KS measure to differentiate among blocks (see Section 4.3).

The MTS algorithm has two control parameters to consider. They are summarized in the following list:

1. α - the cooling schedule
2. Ω - the number of configurations per iteration

α determines the rate at which the temperature is lowered at each iteration of the annealing process. When the temperature is cooled too fast it is possible to get stuck in a local minimum. When the temperature is cooled very slowly unnecessary computation (excessive splitting and merging) is incurred. Ω determines how many state changes are possible at each temperature. When this number is too low it is possible to again get stuck in a local minimum. As with the cooling schedule, when the number is large, unnecessary computation is incurred.

Experiments were performed on both synthesized images and real ultrasound images. Real ultrasound images were used to determine if the algorithms could segment the types of images for which their application is intended. Synthesized images were used because they provided a controlled environment in which the algorithms could be evaluated. The controlled environment is described in terms of the *ground truth* of the synthesized image. In the ground truth, each pixel's location and intensity is known. As well, the region to which each pixel belongs is also known. Thus, it is possible to determine which pixels (or block of pixels) have been partitioned into the proper regions in the segmented image.

There are several parameters which affect the ground truth of synthesized images:

1. I_1 - the size of the texture region,
2. I_2 - the number of distinct texture regions,
3. I_3 - the texture in a region,

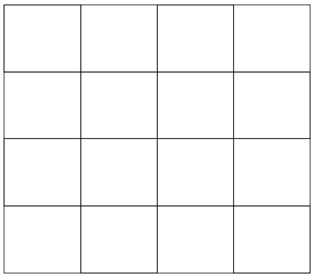


Figure 6.1: Synthesized image with checkerboard texture regions.

4. I_4 - the orientation of the texture,

5. ξ - the amount of noise in the image.

The synthesized images can be created in a variety of forms. One way is to synthesize a checkerboard type image as in Figure 6.1. Each block can represent a different texture region. I_1 controls the size of the blocks in the image. Increasing I_1 will result in an increase in the number of smaller regions in the checkerboard image. I_2 controls how many distinct blocks of texture exist in the checkerboard. I_3 determines the type of texture for a block in the checkerboard. Textures can be generated (eg. all black or all white) or a real ultrasound image can be cropped in different regions to obtain actual textures. These textures are then placed in different blocks of the checkerboard image. I_4 controls the orientation of the texture within a block. Changing I_4 results in rotating the texture. Rotating a texture is used to determine which texture features are least affected by texture orientation. ξ controls the amount of Gaussian noise present in an image. It is measured in the ratio of signal to noise (SNR). The lower the SNR value, the more noise present in the image. ξ is varied to determine how effective the algorithms are at segmenting images in the presence of noise.

The series of real diagnostic ultrasound images of an ovarian follicle were obtained from actual scans of an ovary *in vitro*. The ovary was fixed to the bottom of a tank which was filled with degassed water. The ultrasound probe was attached to a rod above the tank so that the tip of the probe was in the water, just above the ovary. Cross sectional images of the ovary were obtained at 1 mm intervals by moving the tank, with a translational stage, along the scanning axis (perpendicular to the orientation of the probe). At each interval, the ultrasound image was stored to disk using a frame grabber to capture the video out signal from the ultrasound device. Each ultrasound image was 512×512 pixels but only the 256×256 central region, containing the follicle, was cropped out of the image and used for segmentation.

6.2 Experimental Environment

The MTS and CO algorithms were evaluated in a common computing environment. Both algorithms were implemented in the C programming language and run on a Sun Microsystems Sparc Station 1+. The Unix "time" command was used in those experiments which used time as an evaluation criterion. The ground truth of each image was determined so that the texture region to which each pixel belonged was known. A region map was generated for each image which contained the texture region identity for each pixel. Mislabeling error was calculated by comparing the texture region identity of each pixel in the region map to the texture region identity of the corresponding pixels in the resulting segmentations. Since mislabeling error could not be used to determine the segmentation accuracy for real ultrasound images it was necessary to use some qualitative measures. An accurate segmentation is defined to be one which can clearly distinguish the darker central portion of the image (the follicular region) from the rest of the image and appear to produce a homogeneous region of the same shape as the darker central portion of the ultrasound image.

6.3 Evaluating the MTS algorithm

This section evaluates the performance of the MTS algorithm in segmenting both synthetic and real images. Section 6.3.1 discusses the set of test images used in the experiments, Sections 6.3.2-6.3.4 describe the actual experiments performed, and Section 6.3.6 discusses the results of the experiments.

6.3.1 Test Images

Based on the checkerboard format described in Section 6.1, synthetic images consisting of two distinct textures ($I_2 = 2$, the number of distinct texture regions) were used in the experimental procedure. The textures were obtained by sampling homogeneous regions from the actual ultrasound images. Since it was found in Chapter 5 that there were consistently two distinguishable textures in the ultrasound images, those textures were used in the synthetic images. The size of the regions in the image varied from 4×4 pixels up to 64×64 pixels ($I_1 = [4,64]$, the size of the texture region). The 64×64 region was chosen because it is the largest homogeneous region that appears in the ultrasound images (which are 512×512). The 4×4 region was chosen because it is the smallest size possible that can be recognized by the MTS algorithm (4×4 blocks of pixels are the smallest blocks that can characterizing texture with some statistical significance).

In Chapter 5 it was found that most of the features used to characterize the ultrasound textures are rotationally invariant. Examining the textures at different rotations in the synthetic images will generally provide the same results so the original orientation of the textures was used ($I_4 = 0$, the orientation of the texture).

It was also found in Chapter 5 that it was not necessary to increase the noise parameter, ξ, because of the inherently large amount of noise already present in the ultrasound images. It was, however, necessary to preprocess the original ultrasound images to obtain consistently distinguishable textures at different resolutions. Based on this observation, the synthetic images were preprocessed in the same manner.

A number of synthetic images were segmented. Examples are illustrated in Figures 6.2 and 6.3. Figure 6.2(1) shows the original unprocessed synthetic image. This image was then processed by stretching different ranges of pixel intensities so that subtle intensities changes would become noticeable. Figures 6.2(2) - 6.2(6), show the results of stretching the grey values between a lower, l, and upper, u, value by mapping l to 0 and u to 255. Intensities in between these values were interpolated linearly.

The original synthetic image was also processed with the Sobel edge detector.

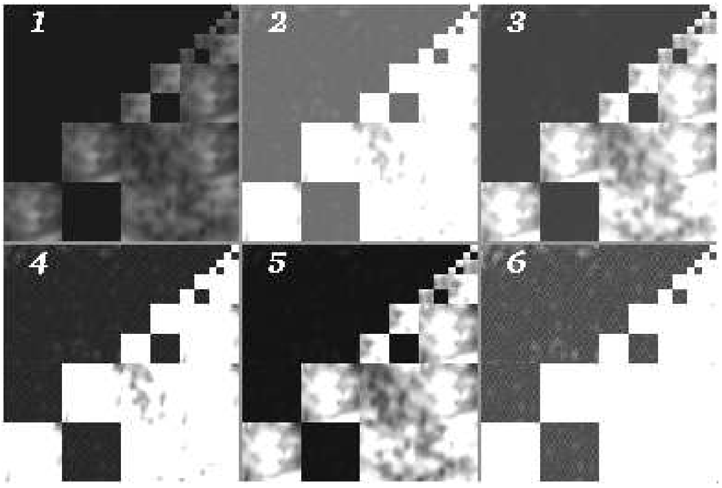


Figure 6.2: Synthetic images: (1) original, (2) l=0,u=60, (3) l=0,u=100, (4) l=20,u=60, (5) l=20,u=100, (6)l=20,u=40.

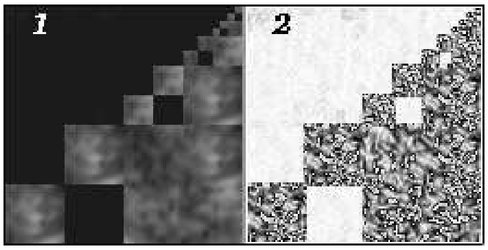


Figure 6.3: Synthetic images: (1) original, (2) convolution with the Sobel operator.

Figure 6.3(2) shows the results.

In some experiments, different versions of the synthetic images were used in order to determine possible experimental bias on the segmentation results. The images in Figures 6.2 and 6.3 have regions with borders at locations which are divisible by 4 and the size of each region is a power of 2. For example, a homogeneous texture region of 32×32 pixels would start at pixel (0,0) and extend for 32 pixels horizontally and vertically. Since the MTS algorithm examines blocks that are confined at exactly these locations and have sizes which are a power of 2, the results of the segmentation would be favorably biased. Hence, it was necessary to shift the image in a given direction by a few pixels. Shifting the images horizontally in Figures 6.2 and 6.3 by 2, 4, and 6 pixels produced the images that would reduce the bias in the segmentation results. In fact, shifting by 2 pixels produces the most mislabelling error since there is a 50% chance that all 4×4 block sizes are labeled incorrectly.

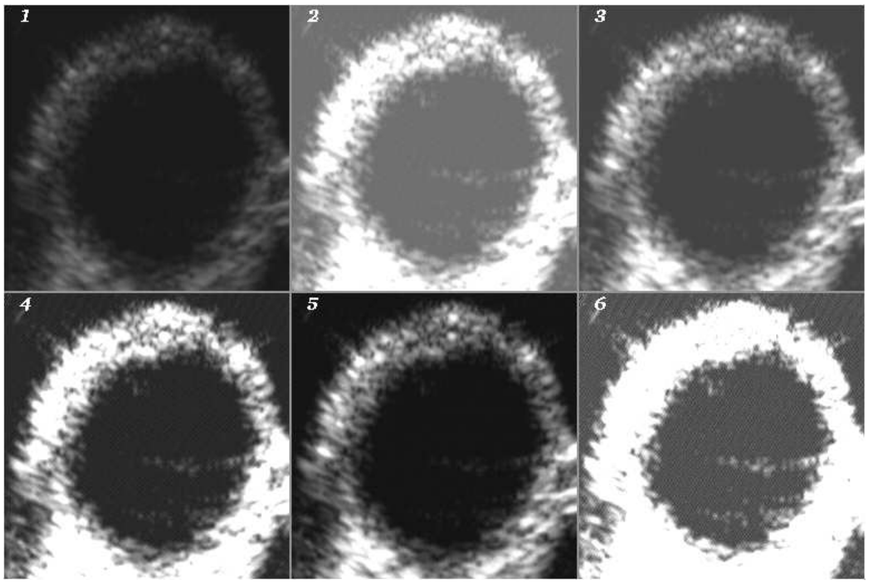


Figure 6.4: Real images: (1) original, (2) l=0,u=60, (3) l=0,u=100, (4) l=20,u=60, (5)l=20,u=100, (6)l=20,u=40.

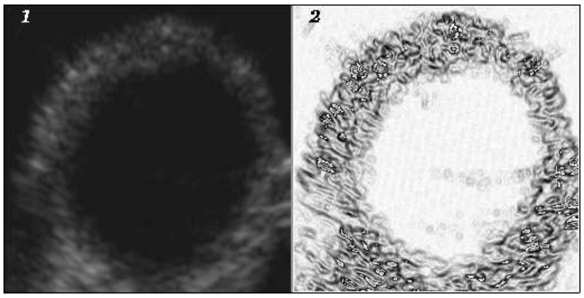


Figure 6.5: Real image of (1) an ovarian follicle, (2) convolution with the Sobel operator.

A number of real ultrasound images were segmented to determine the accuracy of the MTS algorithm in segmenting the images it was. Figure 6.4 shows the original ultrasound image along with its processed versions for different upper and lower range values. The range values are the same as those used for the synthetic images. Figure 6.5 shows the original ultrasound image convolved with the Sobel operator.

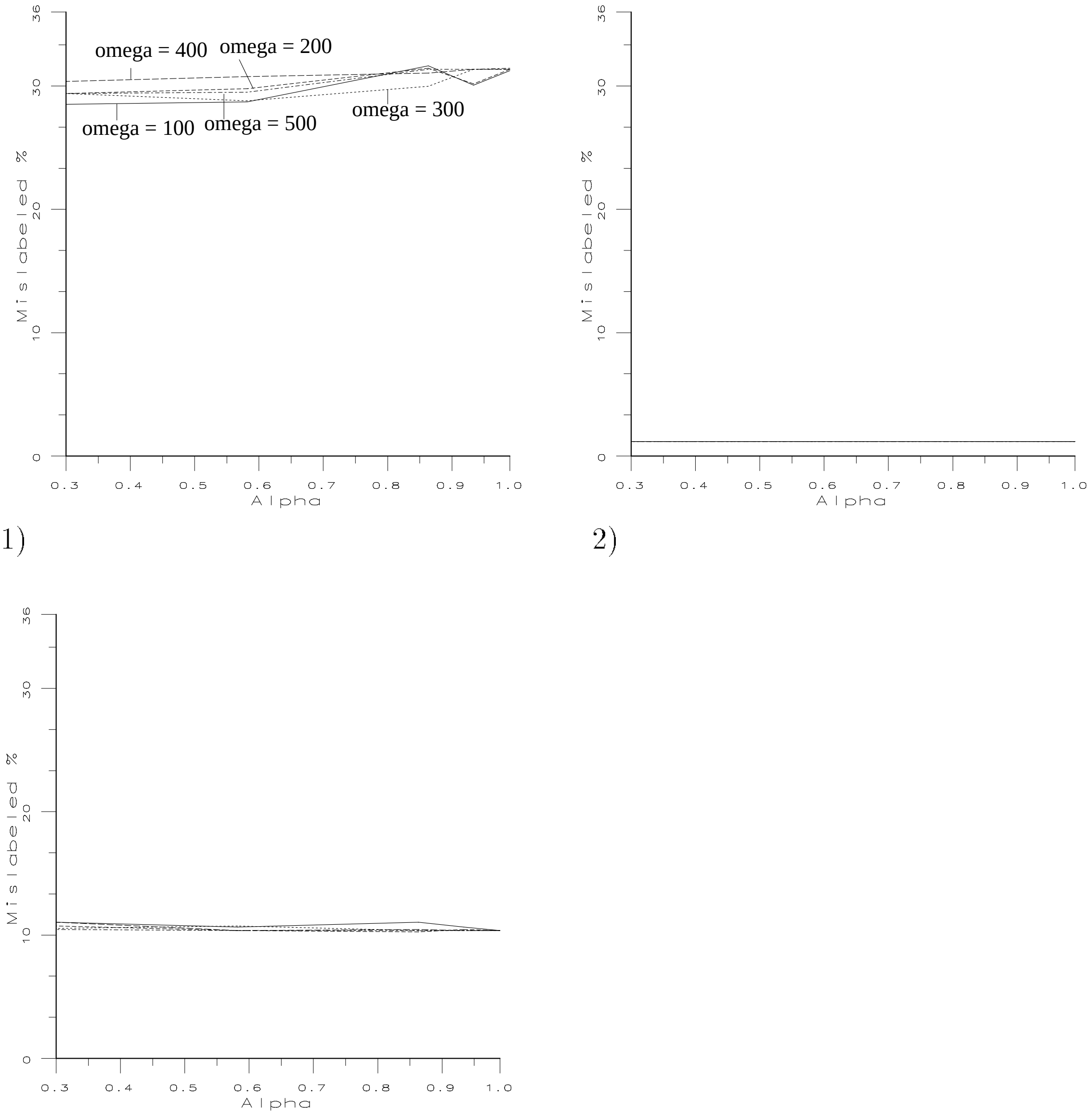
6.3.2 Experiment 4: Scheduling

The purpose of this set of experiments was to determine the most effective schedule for controlling the visitation frequency at which blocks of pixels are examined during the segmentation process. The parameters controlling the schedule are α, the rate of cooling, and Ω, the number of blocks examined at each temperature. By varying the values of α and Ω, different schedules are obtained. The most effective schedule is the one which produces the most accurate segmentation in the shortest amount of time.

Using the controlled environment available with the synthetic images, the image in Figure 6.2(1), its five processed versions in Figures 6.2(2) - Figure 6.2(6), and the image convolved with the Sobel operator in Figure 6.3(2) were segmented thirty times, each with a different setting of α and Ω. In this thesis, $\alpha \in \{.3, .5, .7, .9, .95, .99\}$ and and $\Omega \in \{100, 200, 300, 400, 500\}$. The values for α were chosen so that the effect of very fast and very slow cooling rates. In addition, the rates in between could be evaluated. Since α is defined to be in the range (0.0,1.0), $\alpha = .3$ results in a rapid cooling rate while $\alpha = .99$ results in a very slow cooling rate. The values for Ω were chosen so that both small and large numbers of pixel blocks would be visited per temperature interval. When $\Omega = 100$, a small number of blocks are visited during the segmentation process as compared to the number of blocks visited when $\Omega = 500$.

The mislabeling error and cpu times for the resulting segmentations are shown in Figures 6.6 - 6.8. The graphs in Figure 6.6 show the mislabeled error curves resulting in the segmentations of the original image, the stretched image (l=20,u=40), and the convolved image for each value of Ω. The graphs in Figure 6.7 show the comparison of the mislabeling errors for the three images for each Ω value. In Figure 6.8, the average segmentation time is shown for each value of Ω. The average segmentation time is the averaged sum of the segmentation times for each of the three images with the same α,Ω values. The average of the segmentation times was used since the type of image being segmented was found to have little effect on the individual segmentation times. The fastest segmentations were achieved with $\Omega = 100$ (the bottom curve) while the slowest segmentations were achieved with $\Omega = 500$ (the top curve).

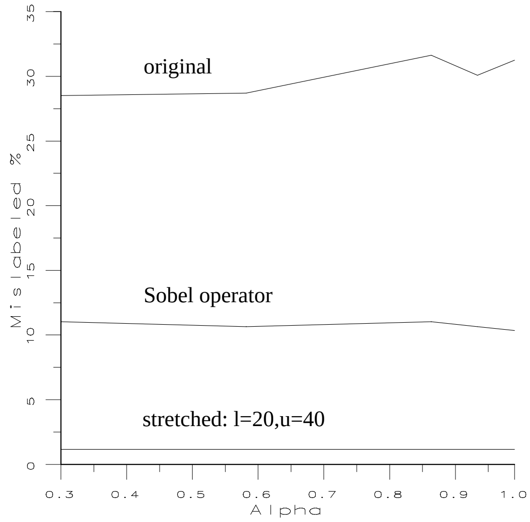
The results obtained from the other four stretched images did not approach those obtained by the two processed images. Therefore, the only stretched image used for subsequent experimentation was the one with l=20 and u=40. Figures 6.18 - 6.20 provide some example segmentations produced for the original, stretched, and



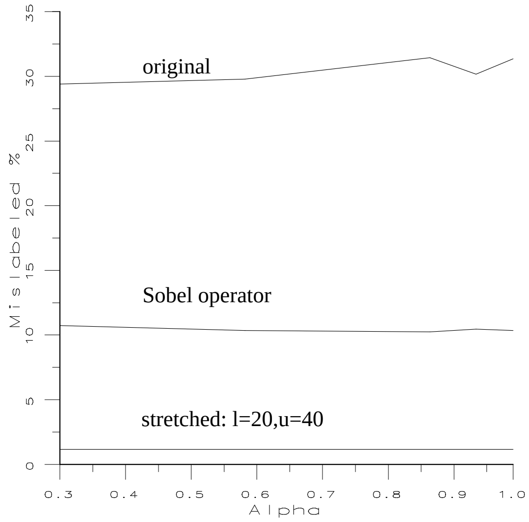
1) 2)

3)

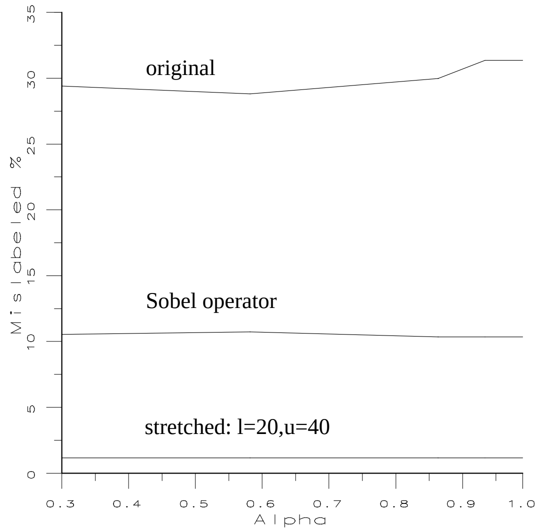
Figure 6.6: Mislabeling error; Ω varying from 100 to 500 for (1) original test image, (2) l=20,u=40, (3) Sobel operator.



1)

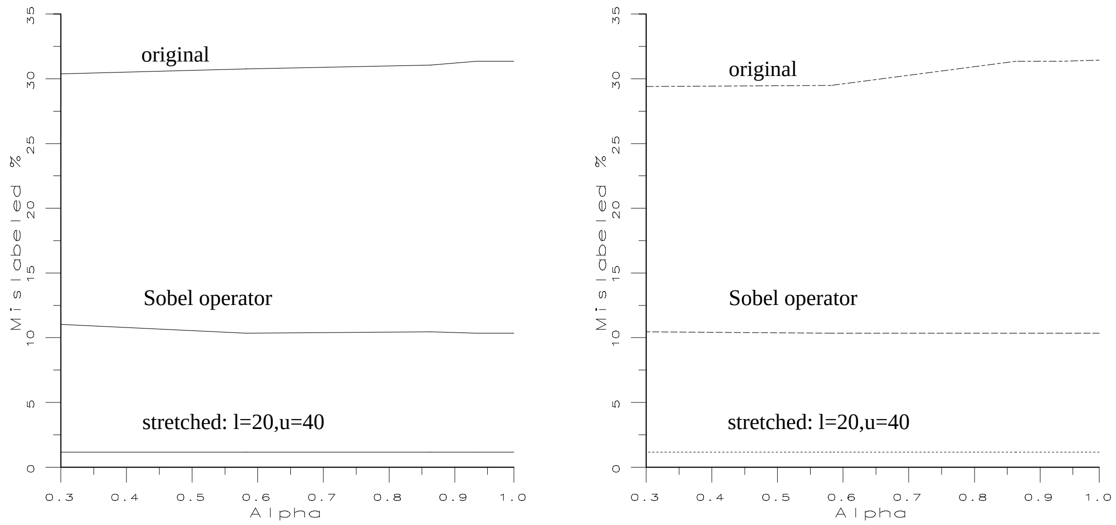


2)



3)

Figure 6.7: Mislabeling error for (1) Ω=100, (2) Ω=200, (3) Ω=300 (*continued next page*)



4) 5)

(*continued*) Figure 6.7 (4) Ω=400, (5) Ω=500.

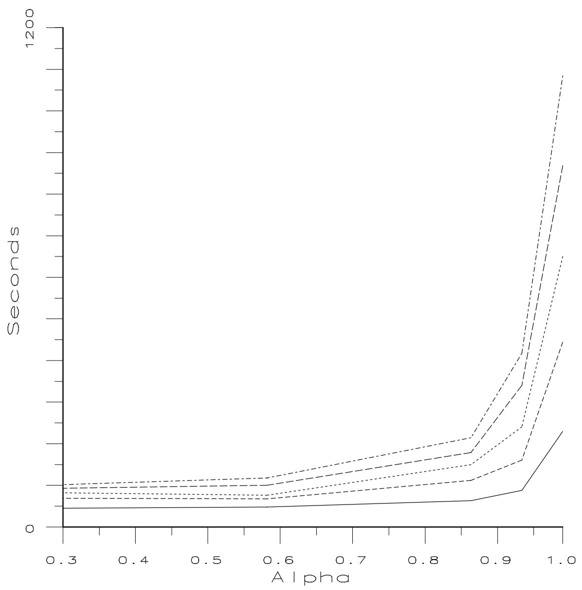


Figure 6.8: Average time to segment the synthetic images with Ω varying from 100 (bottom curve) to 500 (top curve).

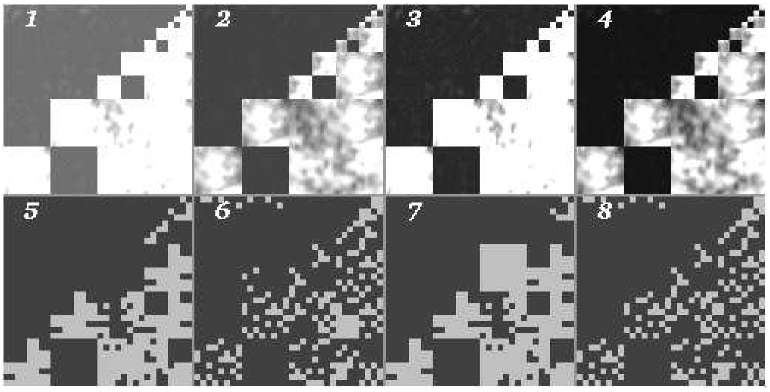


Figure 6.9: Segmentation results for synthetic images,Ω=300,α=.9, (1) l=0,u=60, (2) l=0,u=100, (3) l=20,u=60, (4) l=20,u=100, (5) segmentation results of (1), (6) segmentation results of (2), (7) segmentation results of (3), (8) segmentation results of (4).

convolved synthetic images. Figure 6.9 provides an example of the segmentations produced for the other four stretched images.

Another measure used to evaluate the MTS algorithm in this experiment was the average number of configuration changes performed when segmenting the three synthetic images. This was measured in hopes of providing more insight as to the most effective choice for α and Ω. A configuration change occurs when a block is either split into four smaller blocks or merged with its three neighbors. A large number of configurations changes could suggest an unnecessary number of random splits or merges. A small number of configuration changes could suggest that the solution is stuck in a local minimum.

Figure 6.10 shows the resulting average number of configuration changes for each value of Ω. Note the relatively small effect of Ω until $\alpha = .7$. This suggests that a cooling rate below $\alpha = .7$ is very likely to produce a segmentation which is not optimal (no matter how large Ω gets) because there are not enough random configurations allowed at each temperature interval.

The real ultrasound images were segmented with the same α and Ω values as the synthetic images. The resulting segmentations for the original, stretched, and convolved real images are shown in Figures 6.11, 6.12 and 6.13 respectively. In each figure, α varies from .99 to .3 along each row while Ω varies from 100 to 500 along

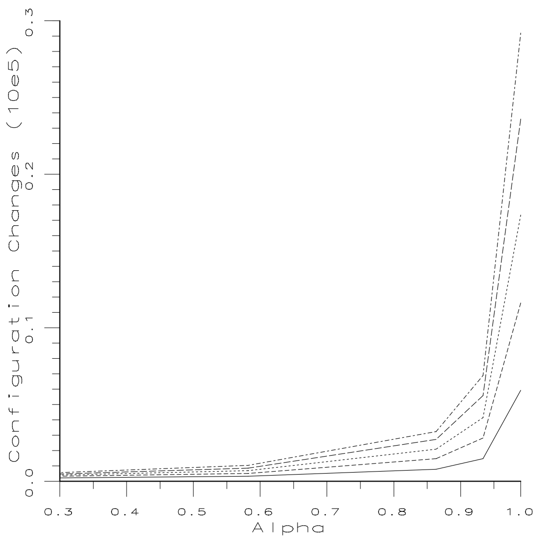


Figure 6.10: Average Number of Configuration Changes with Ω varying from 100 (bottom curve) to 500 (top curve).

each column. For the sake of completeness, the resulting segmentations for the other four stretched real ultrasound images are shown in Figure 6.14. Images (1) -(4) show the stretched ultrasound images while the resulting segmentations with $\alpha = .9$ and $\Omega = 300$ are shown in images (5) -(8) .

The segmentation times for these images are shown in Figure 6.15. As with the synthetic images, the times for the real images are fairly constant until $\alpha > .9$. At this point there is a sharp rise in the amount of time required to produce the segmentations. Also, as Ω gets large, the amount of time increases. This is intuitive because when Ω gets larger, more pixel blocks are examined, thus increasing the amount of computation. The number of configuration changes performed is shown in Figure 6.16. In all three graphs, the number of configuration changes is relatively low until $\alpha > .95$. This suggests that using a schedule with $\alpha > .95$ will inevitably result in unnecessary, random configuration changes during the segmentation process.

The segmentation results for the synthetic images imply that a relatively fast schedule (low values for α and Ω) could be used for segmenting images. The segmented real images in Figures 6.11-6.13 and the associated graphs in Figures 6.15 and 6.16,

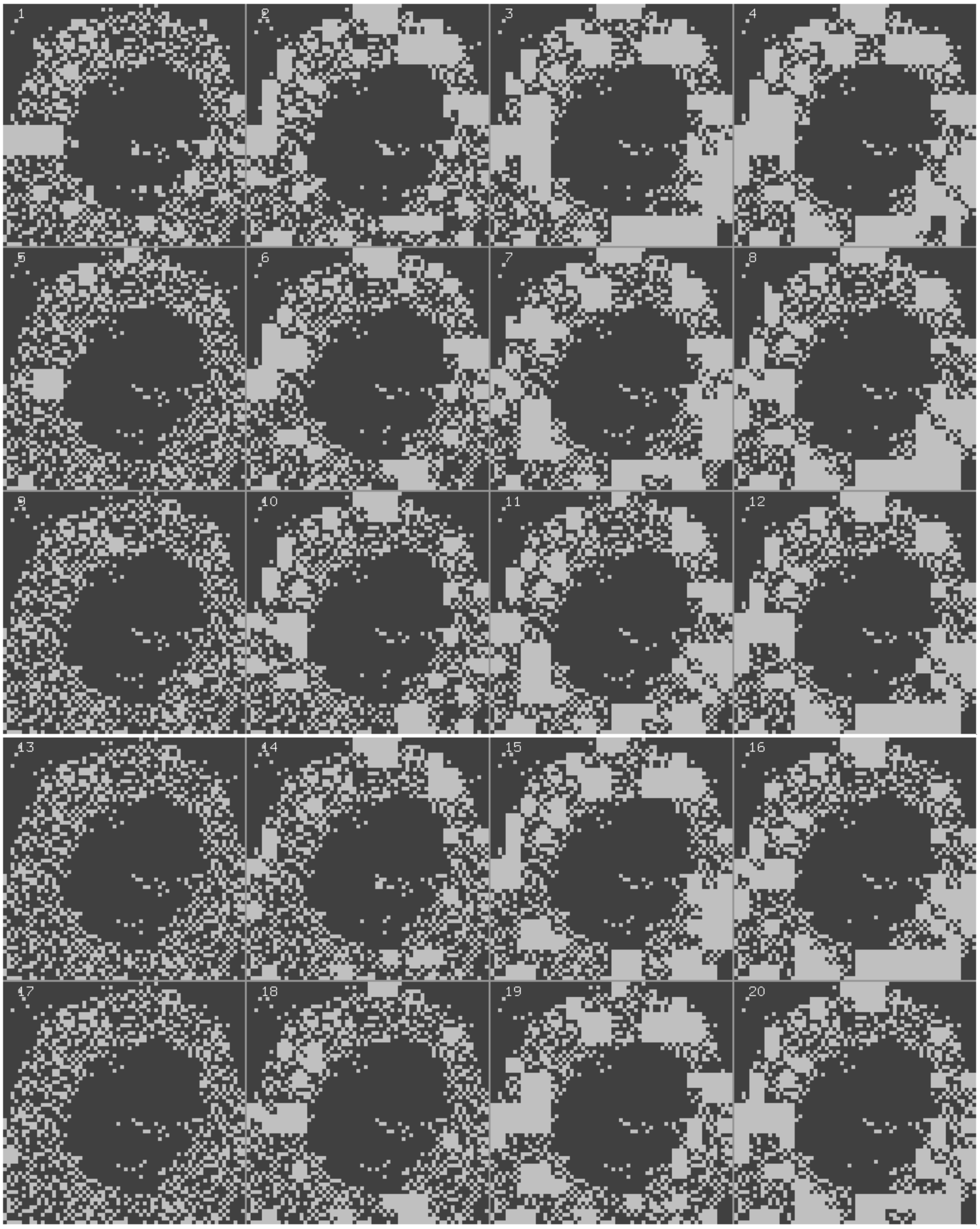


Figure 6.11: Segmentation results for original real image: (1)-(4) Ω=100, α=.99 .9 .7 .3, (5)-(8) Ω=200, α=.99 .9 .7 .3, (9)-(12) Ω=300, α=.99 .9 .7 .3, (13)-(16)Ω=400, α=.99 .9 .7 .3, (17)-(20)Ω=500, α=.99 .9 .7 .3.

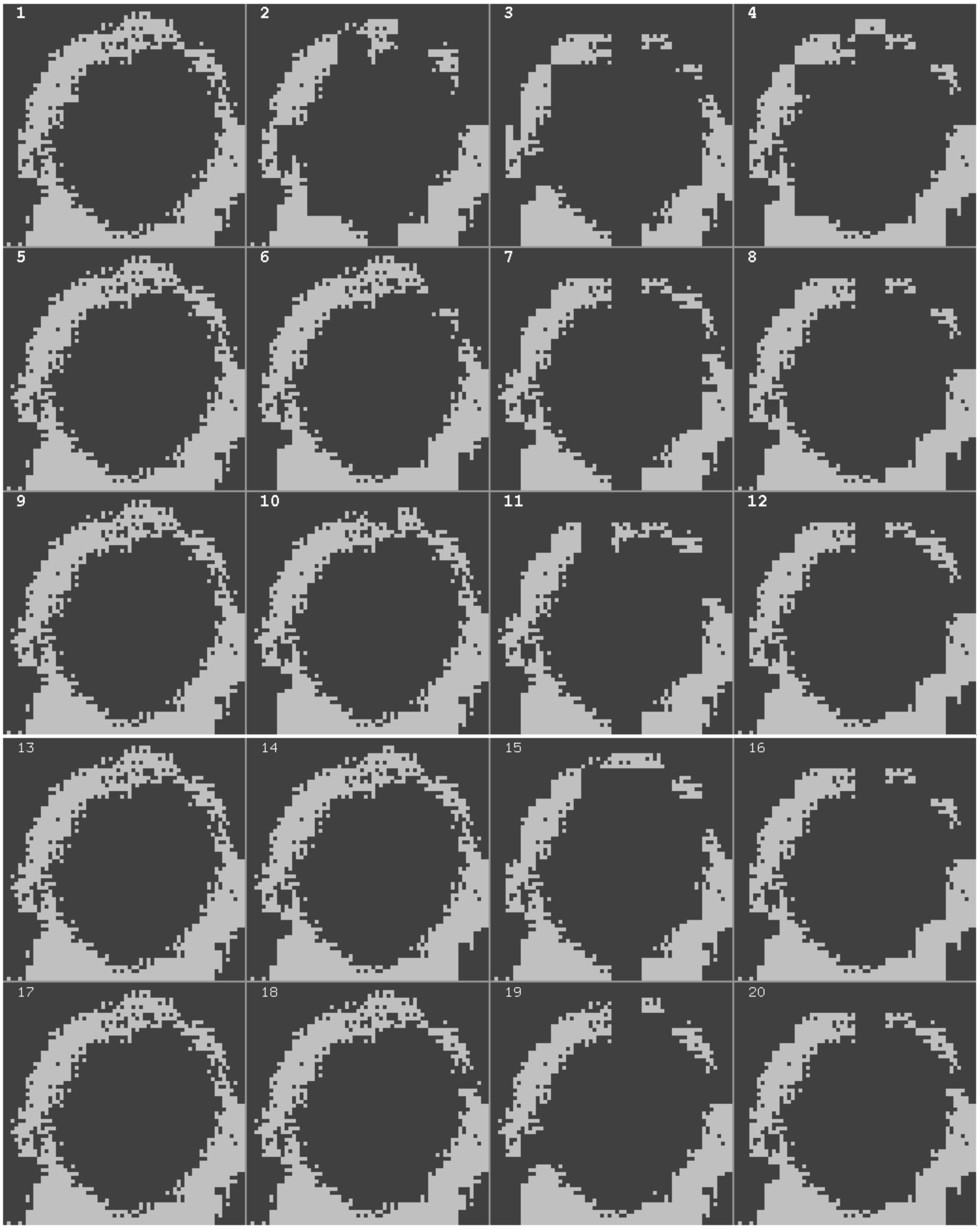


Figure 6.12: Segmentation results for real image, l=20,u=40: (1)-(4) Ω=100, α=.99 .9 .7 .3, (5)-(8) Ω=200, α=.99 .9 .7 .3, (9)-(12) Ω=300, α=.99 .9 .7 .3, (13)-(16)Ω=400, α=.99 .9 .7 .3, (17)-(20)Ω=500, α=.99 .9 .7 .3.

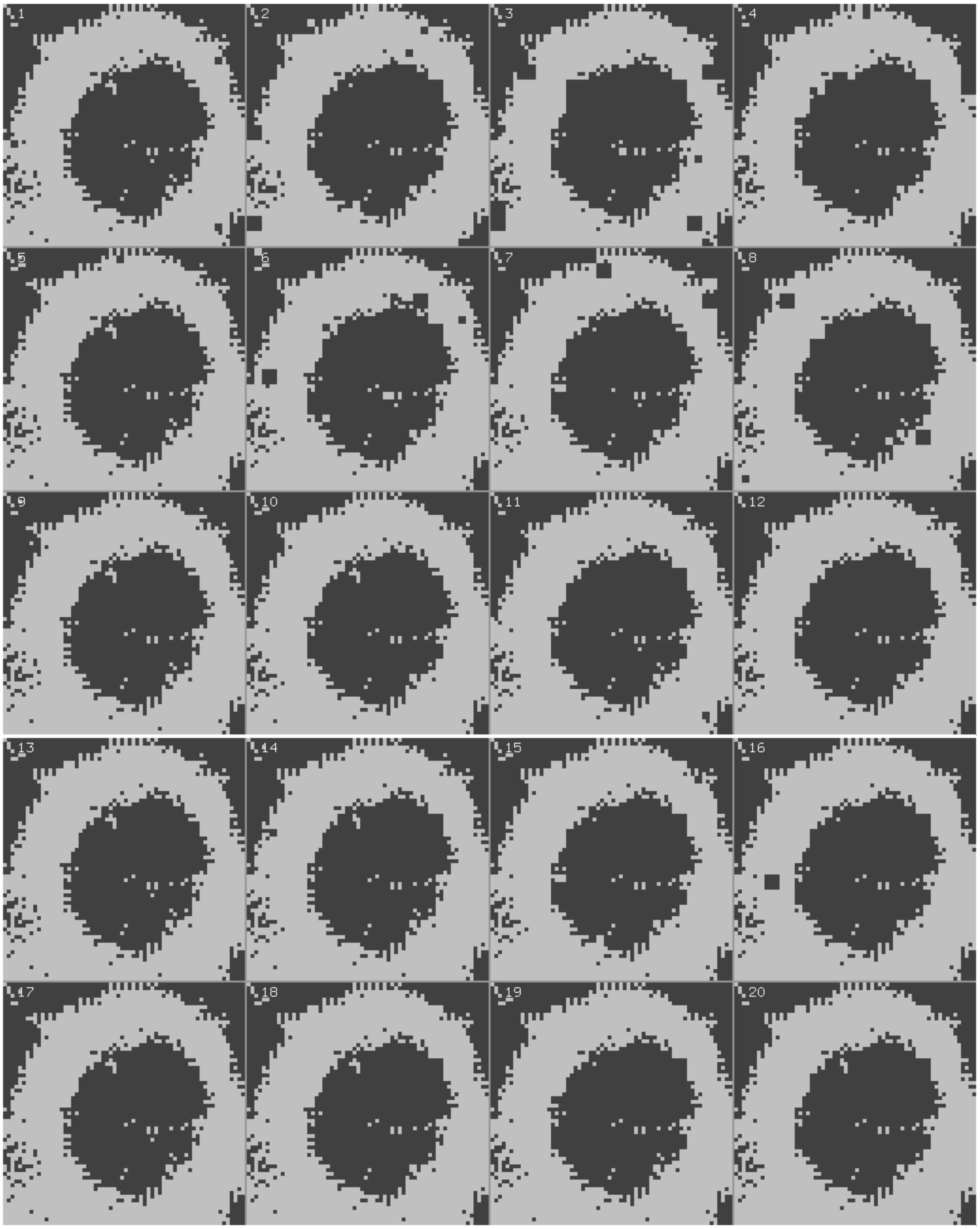


Figure 6.13: Segmentation results for real image convolved with Sobel operator: (1)-(4) Ω=100, α=.99 .9 .7 .3, (5)-(8) Ω=200, α=.99 .9 .7 .3, (9)-(12) Ω=300, α=.99 .9 .7 .3, (13)-(16)Ω=400, α=.99 .9 .7 .3, (17)-(20)Ω=500, α=.99 .9 .7 .3.

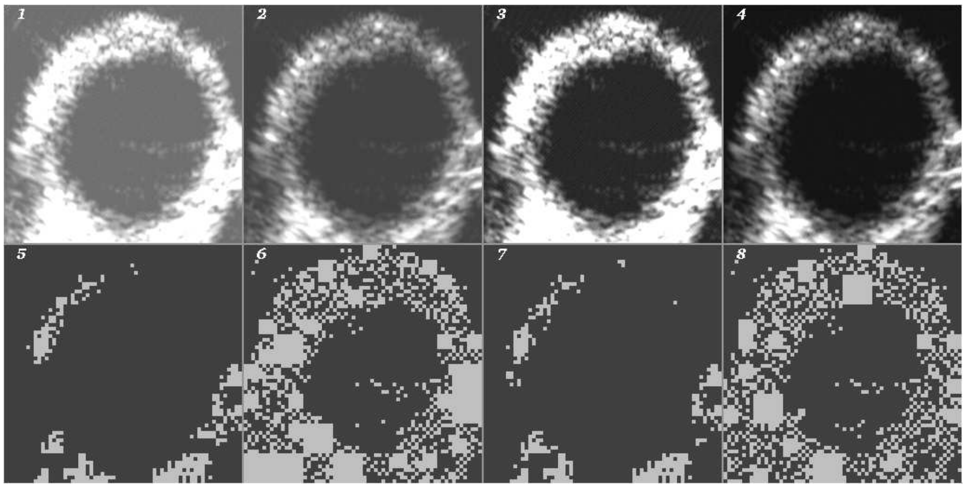


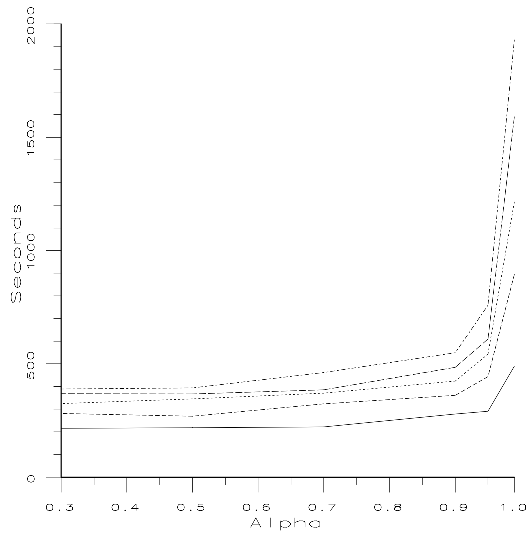
Figure 6.14: Segmentation results for stretched real images,Ω=300,α=.9, (1) l=0,u=60, (2) l=0,u=100, (3) l=20,u=60, (4) l=20,u=100, (5) segmentation results of (1), (6) segmentation results of (2), (7) segmentation results of (3), (8) segmentation results of (4).

however, imply that a slower schedule (higher values for α and Ω) is necessary to achieve a good segmentation. Since the ultimate goal is to produce an accurate 3D model of an ovarian follicle based on a series of **real** (not synthetically contrived) ultrasound images, the following parameters were determined to produce an effective schedule for the MTS algorithm:

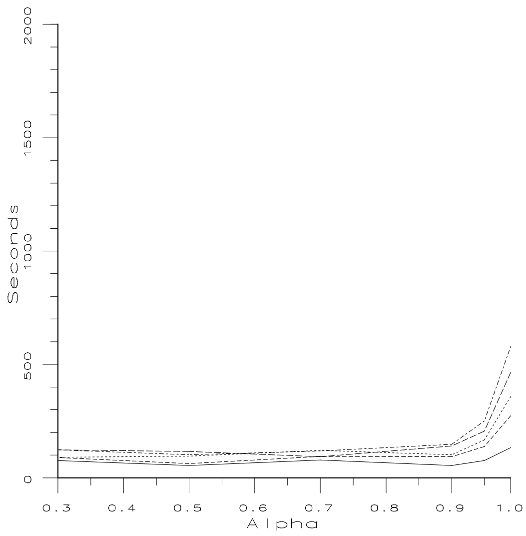
- $\alpha = .9$
- $\Omega = 300$.

These parameter settings were used for the remainder of the experiments. It is interesting to note that these parameter settings were not based solely on the results of the synthetic images. Even though the synthetic images contain textures sampled from the real ultrasound images, their segmentation results were not the same (though similar) as those from the real images. A qualitative analysis of the segmentation results from the real images was also used to determine the parameter settings for α and Ω. This shows that it is of utmost importance for any segmentation algorithm to be tested with both real and synthetic images.

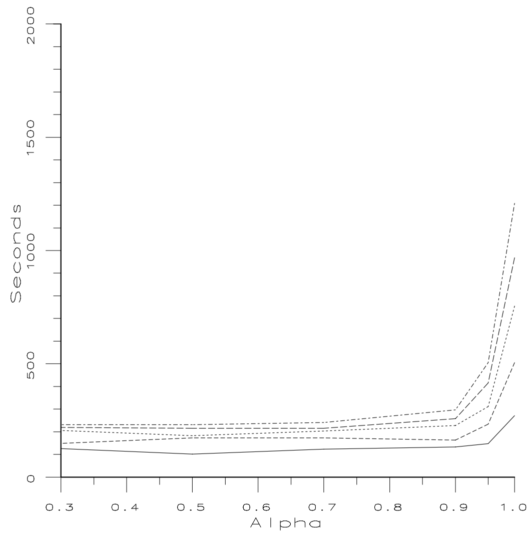
A sample segmentation sequence of a real ultrasound image is illustrated in Figure 6.17. Figure 6.17(2) shows the initial guess at the segmentation. At this point the



1)

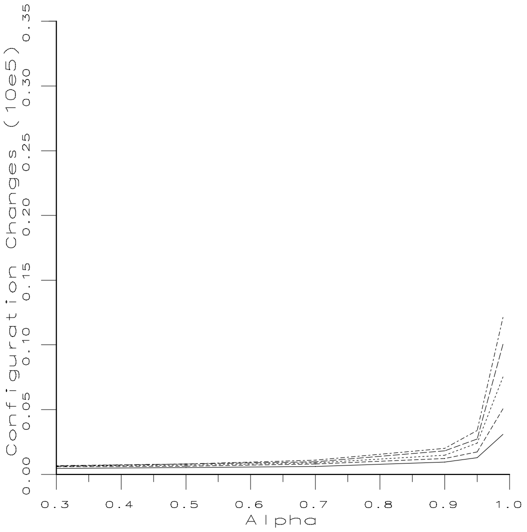


2)

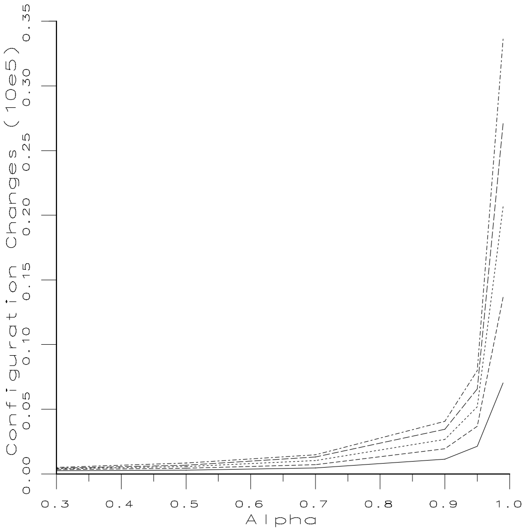


3)

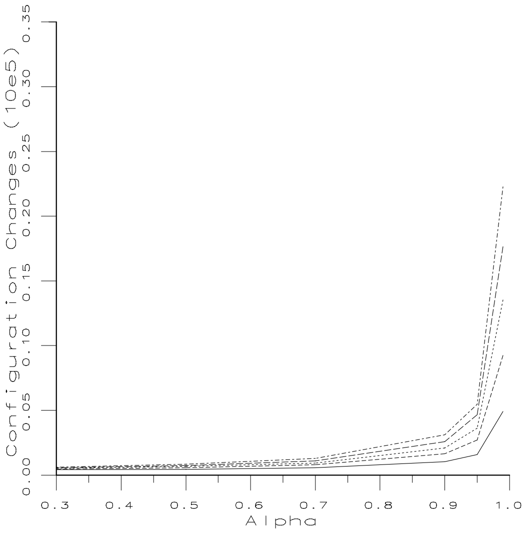
Figure 6.15: Time to segment real image with Ω varying from 100 (bottom curve) to 500 (top curve) for (1) original real image, (2) l=20,u=40, (3) Sobel operator.



1)



2)



3)

Figure 6.16: Number of Configuration Changes with Ω varying from 100 (bottom curve) to 500 (top curve) for (1) original real image, (2) l=20,u=40, (3) Sobel operator.

label for each block is unknown. As the segmentation proceeds, labeled blocks begin to identify the follicular region and the surrounding tissue region (Figures 6.17(3) - 6.17(23)). The final segmentation is shown in Figure 6.17(24)).

6.3.3 Experiment 5: Image Bias

To determine if the synthetic images used in experiment 4 were biased towards the segmenting mechanism of the MTS algorithm (see Section 6.3.1), it was necessary to perform some experiments with different versions of the synthetic images. Since experiment 4 found that the contents of the synthetic image being segmented had little effect on segmentation time and the number of configuration changes, it was only necessary to examine the mislabeling error to see if the synthetic test images were biased towards the MTS algorithm.

Figures 6.18, 6.19, and 6.20 show the resulting segmentations of the original, stretched, and convolved images respectively. Images (1) - (4) in each figure show the synthetic image shifted by 0, 2, 4, and 6 pixels while images (5) - (8) show the resulting segmentations.

Figure 6.21 shows the resulting mislabeling error when the original, stretched, and convolved images are shifted left by 0, 2, 4, and 6 pixels. For the original synthetic image, shifting caused the mislabeling error to fluctuate a few percentages. As expected, the mislabeling error was the highest when the image was shifted by 2 pixels. For the processed synthetic images the mislabeling error was, in general, slightly larger for the shifted versions. However, the increased mislabeling error did not indicate a significant level of bias towards the synthetic images used in experiment 4. As a result, subsequent experimentation used the same set of synthetic images as in experiment 4.

6.3.4 Experiment 6: Level Independence

The purpose of this set of experiments was to determine if the level at which the segmentation is initiated affects the final segmentation results. The MTS algorithm

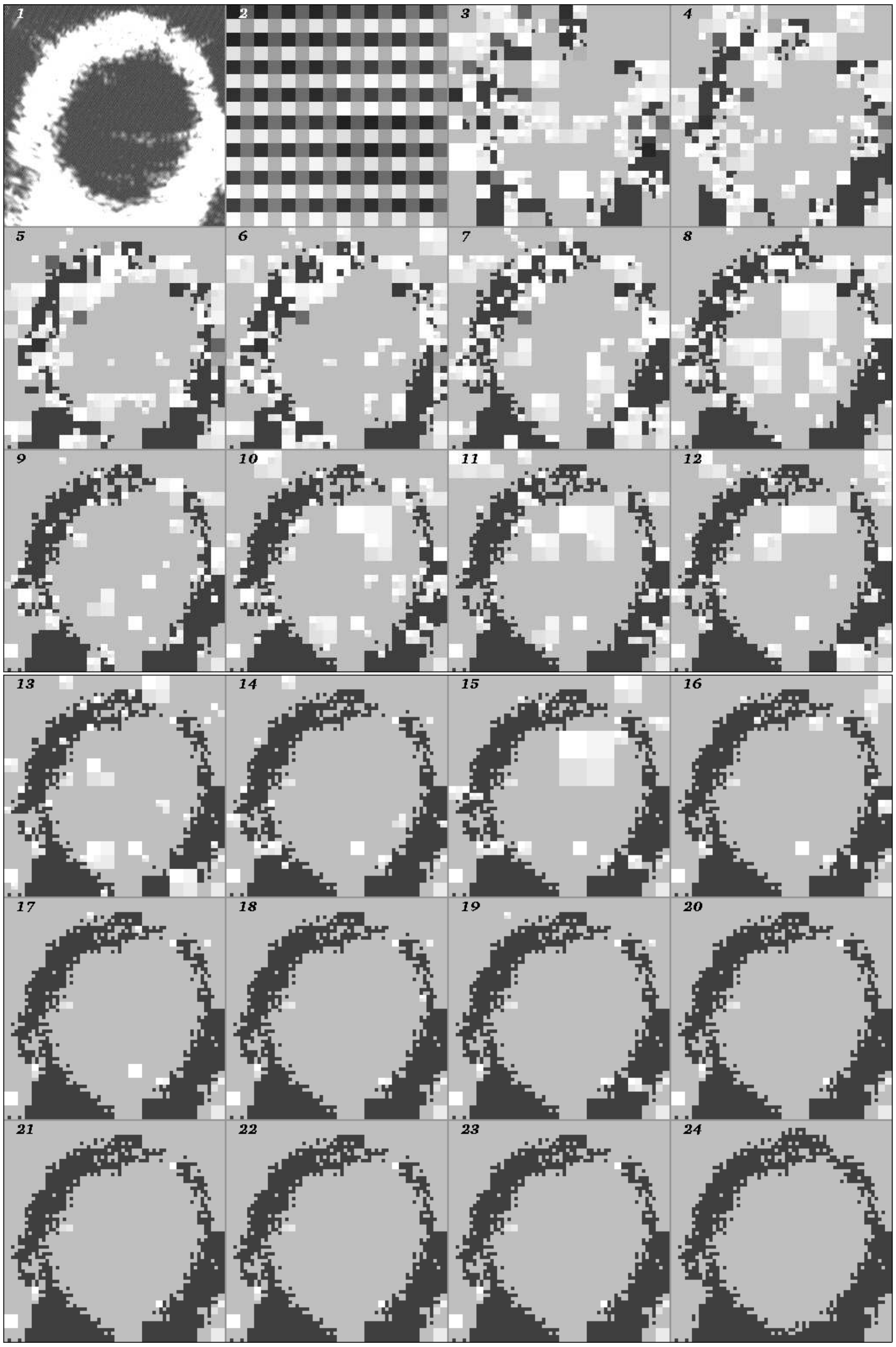


Figure 6.17: Segmentation sequence of a real image for l=20,u=40,Ω=300,α=.9: (1) the stretched image, (2) initial segmentation with no label assignment, 3-23)solution after every second iteration, 24)final segmentation.

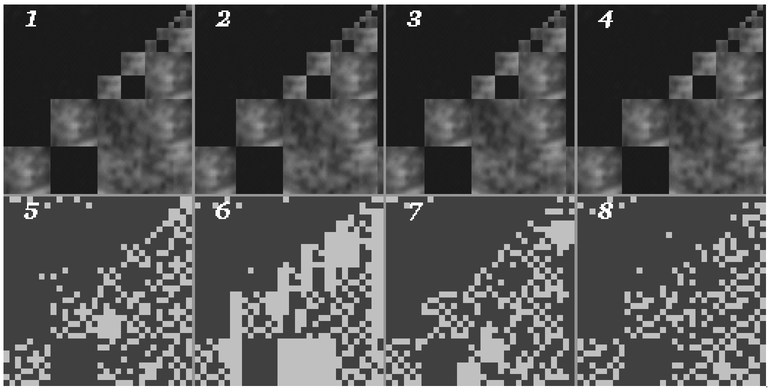


Figure 6.18: Segmentation results for Ω=300, α=.9: (1) original test image, shifted left by (2) 4 pixels, (3) 6 pixels, (4) 2 pixels, (5) segmentation results of (1), (6) segmentation results of (2), (7) segmentation results of (3), (8) segmentation results of (4).

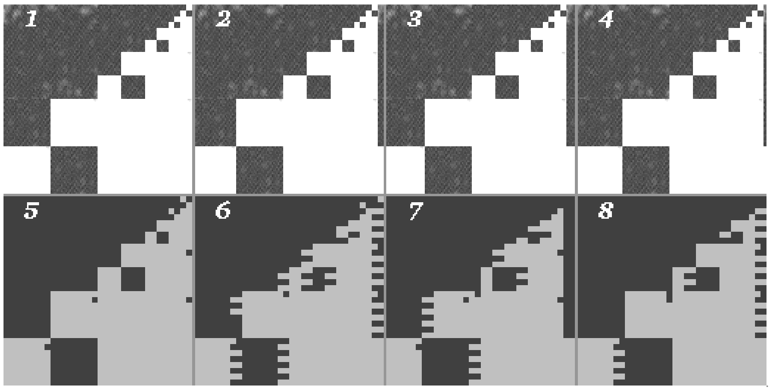


Figure 6.19: Segmentation results for l=20,u=40, Ω=300, α=.9: (1) test image, shifted left by (2) 4 pixels, (3) 6 pixels, (4) 2 pixels, (5) segmentation results of (1), (6) segmentation results of (2), (7) segmentation results of (3), (8) segmentation results of (4).

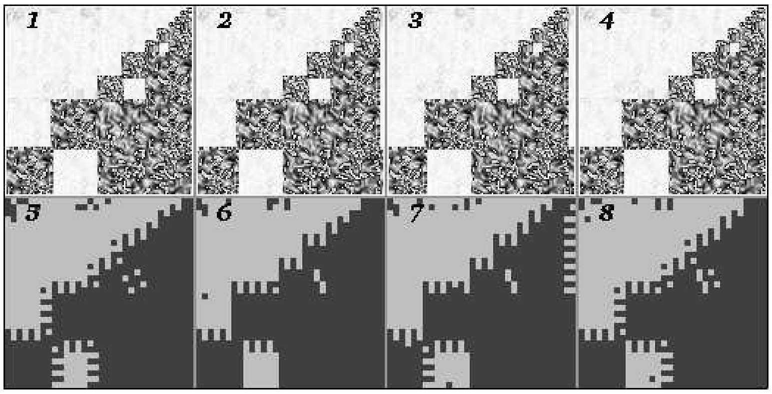


Figure 6.20: Segmentation results for the Sobel operator, Ω=300, α=.9: (1) test image, shifted left by (2) 4 pixels, (3) 6 pixels, (4) 2 pixels, (5) segmentation results of (1), (6) segmentation results of (2), (7) segmentation results of (3), (8) segmentation results of (4).

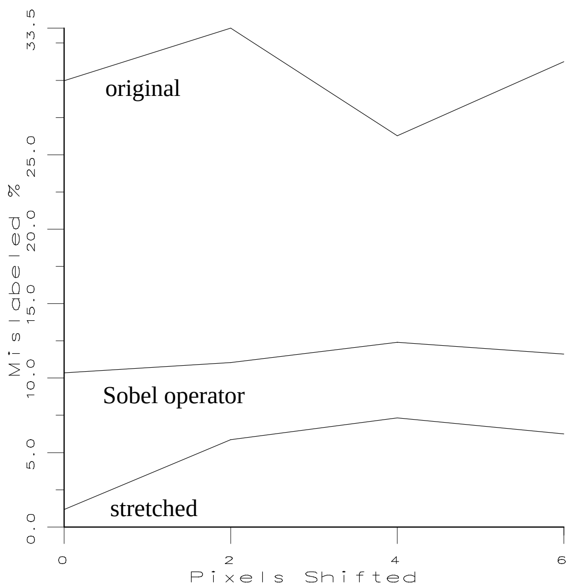


Figure 6.21: Mislabeling error for synthetic images shifted by 0, 2, 4 and 6 pixels: Ω=300 and α=.9.

uses a quad tree structure for keeping track of the different levels of resolution at which the pixel blocks in the image are being examined. The entire image is represented as level 1 in the quad tree. Lower levels in the tree have their nodes representing smaller pixel blocks. Level n is the lowest level in the tree with each of its nodes representing pixel blocks of size $\frac{N}{2^{(n-1)}} = 4$ where N is the size of the image. By initiating the same segmentation at varying levels, it is possible to determine if the results are independent of the initial level chosen.

The set of three synthetic images was segmented first with $\alpha = .9$ and $\Omega = 300$. The initial level was varied from 2 to 6 (n=6). Figure 6.22 shows the resulting segmentations for each of the synthetic images initiated from varying levels. Note that for the processed images, the set of segmentations depicted in Figures 6.22(8) - 6.22(12) and 6.22(14) - 6.22(18) are almost identical, suggesting that the level at which segmentation is initiated does not have an effect on the final segmentation (the differences in the segmentations for the original synthetic image may be a result of randomness and the inadequate characterization of the textures in their original form).

The effect of the initial segmentation level on the configuration changes for the synthetic image segmentations is shown in Figures 6.23 and 6.24. Figure 6.23 compares the type of configuration changes for each image while Figure 6.24 shows the distribution of the configuration changes for each image. Note that as the level increases, the amount of merging increases while the amount of splitting decreases. This is an intuitive result because when the initial level is small, larger blocks are examined, requiring more splitting than merging to achieve the appropriate detail in the segmentation. Conversely, as the level gets larger, the amount of splitting decreases as more merging is required to achieve the same detail in the segmentation. In both cases, the number of random configuration changes should remain relatively constant (slight fluctuations due to the randomness involved). The graphs verify this. The effect of the initial level on the mislabeling error is shown in Figure 6.25. For the processed images, the error is constant (except for the slight decrease with the convolved image when level = 6). The variability in mislabeling error is predominantly due to

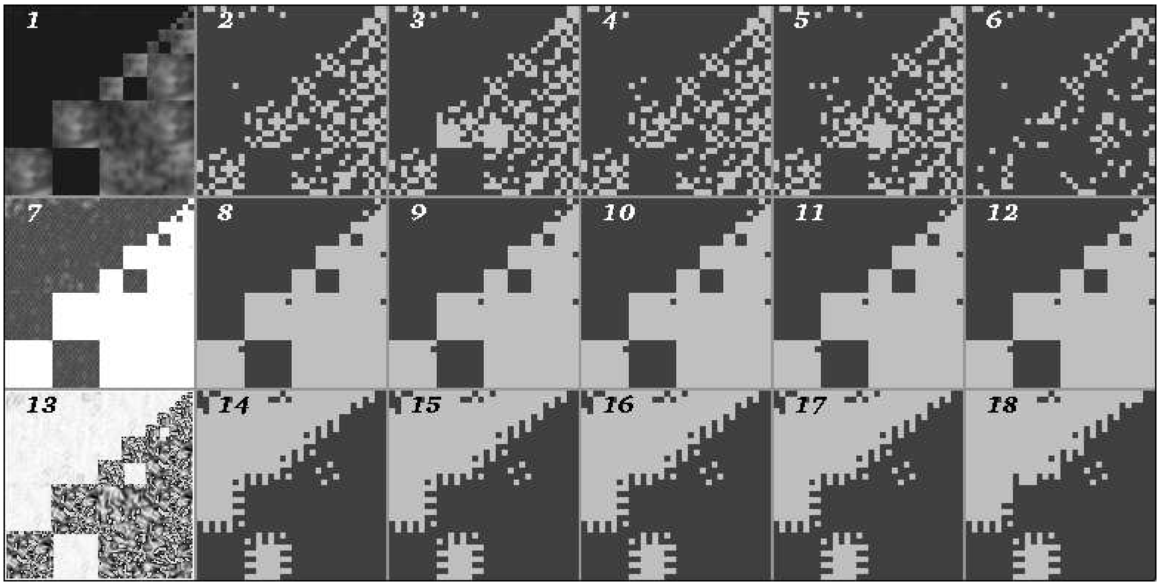


Figure 6.22: Segmentation of synthetic images at varying initial resolutions, Ω=300, α=.9, (1) original synthetic image, (2)-(6) resulting segmentations for level=2, 3, 4, 5, 6, 7) enhanced image with l=20,u=40, (8)-(12) resulting segmentations for level=2, 3, 4, 5, 6, 13) convolution with the Sobel operator, (14)-(18) resulting segmentations for level=2, 3, 4, 5, 6.

the accuracy in which each texture is characterized at different resolutions. When the segmentation is initiated at a high level the majority of the size of blocks examined is larger than when the segmentation is initiated at a lower level. For example, with the original synthetic image it is evident that texture was characterized more accurately at larger resolutions since there is a significant increase in the mislabeling error when the initial level = 6.

The real ultrasound images were segmented in the same manner to determine if the results would be consistent with those achieved for the synthetic images. Figure 6.26 shows the resulting segmentations for the original, enhanced, and convolved real ultrasound images with $\alpha = .9$ and $\Omega = 300$. As with the processed synthetic images, the segmentations initiated at varying levels for the real processed images are quite similar. In Figure 6.26(5), smaller details were lost in the large blocks resulting in a more coarse solution. The variability in the segmentations for the original real image can be attributed to the randomness and inadequate texture characterization (as with the original synthetic image). The graphs in figure 6.27 compare the types

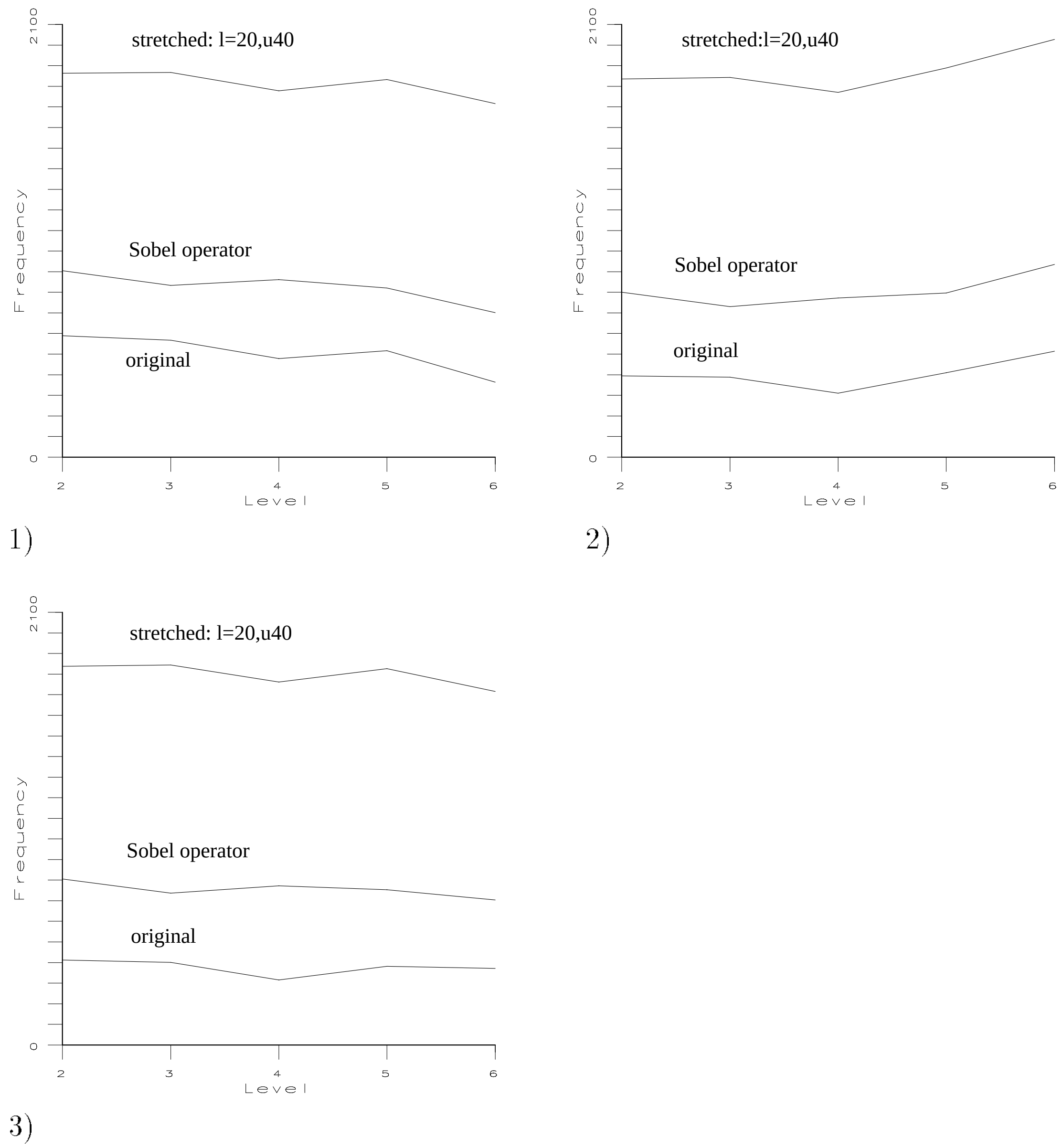
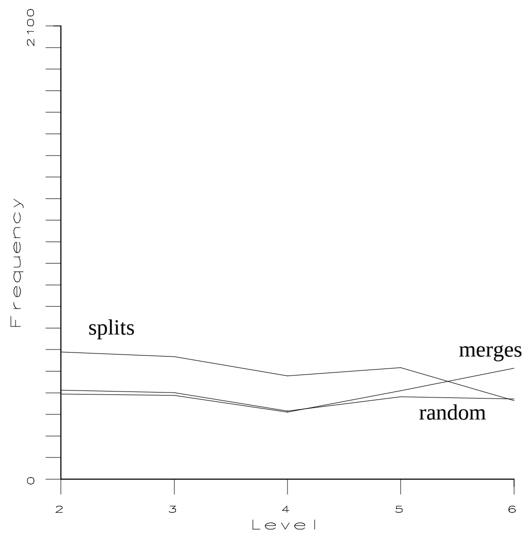
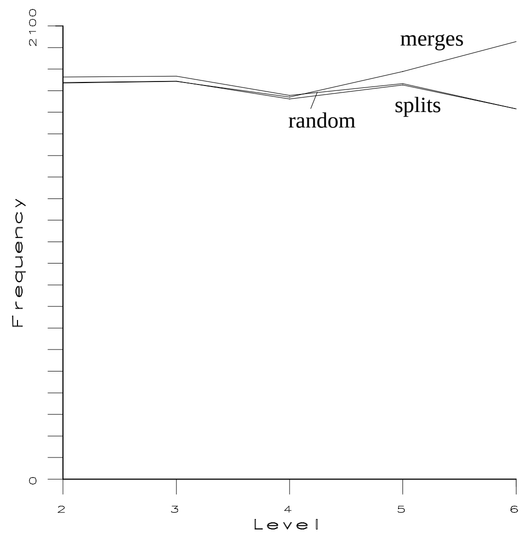


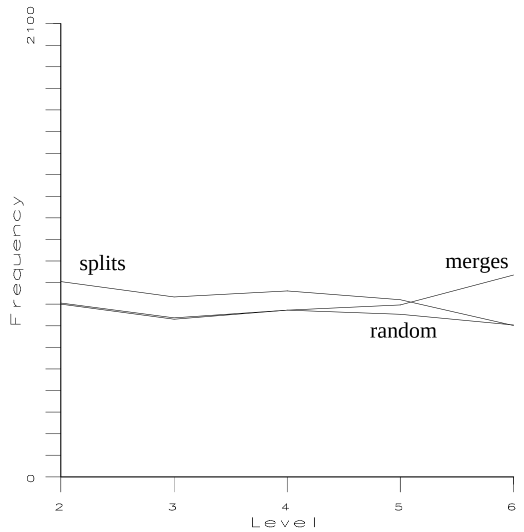
Figure 6.23: Comparison of configuration changes during segmentation of synthetic images with Ω=300, α=.9; (1) blocks split, (2) neighborhoods merged, (3) total random splits and merges.



1)



2)



3)

Figure 6.24: Distribution of configuration changes during segmentation of synthetic images with Ω=300, α=.9; (1) original synthetic image, (2) enhanced synthetic image with l=20,u=40, (3) Convolution of the Sobel operator with the original synthetic image.

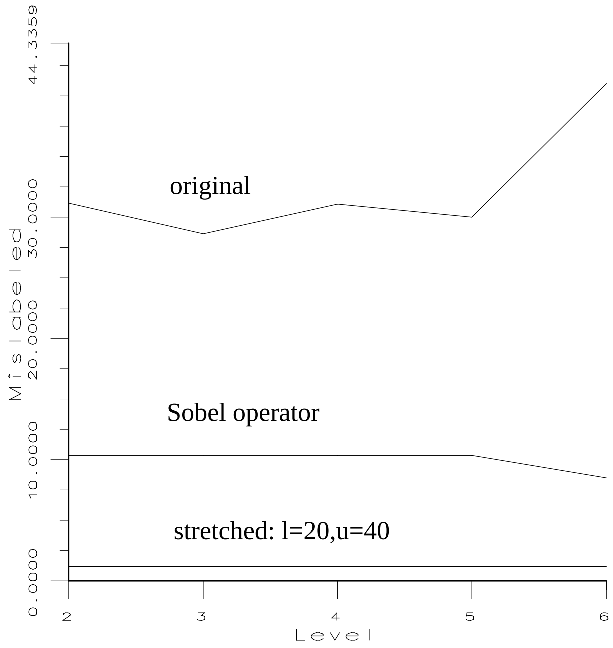


Figure 6.25: Mislabeling error for segmented synthetic images, Ω=300, α=.9.

of configuration changes for the real images. The graphs in Figure 6.28 show the distribution of the configuration changes for each image. In both figures, the results are consistent with those obtained for the synthetic images. As the level increases, the amount of merging increases, the amount of splitting decreases and the number of random configuration changes remains relatively constant.

6.3.5 Experiment 7: Fixed Feature

The purpose of this set of experiments was to determine if the use of different features at varying resolutions produces better results than using a fixed feature detector (see Chapter 5 for a description of the features). In these experiments, images were segmented three different times. The first segmentation was performed using the feature, f_i, which most accurately characterized the texture blocks at the largest resolution (64×64 for the synthetic images and 128×128 for the real ultrasound images). The second segmentation used the feature which most accurately characterized the texture blocks at the smallest resolution (4×4). The third segmentation was performed using

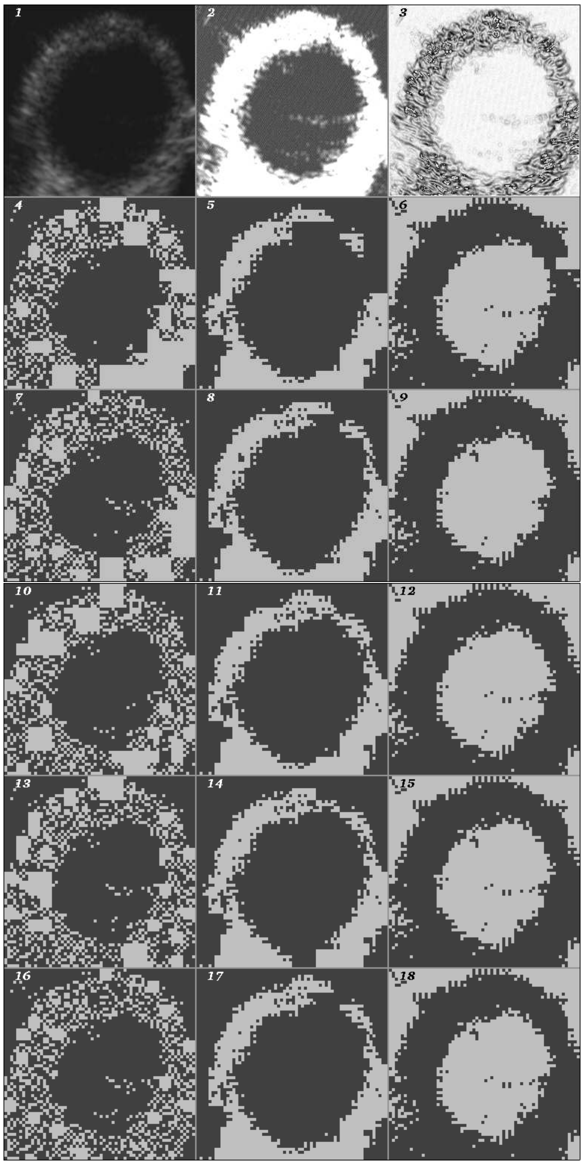


Figure 6.26: Segmentation of a real image at varying starting resolutions, Ω=300, α=.9, (1) original real image, (2) enhanced image with l=20,u=40, (3) convolution of the Sobel operator with the original real image, (4)-(18) resulting segmentations for level=2, 3, 4, 5, 6.

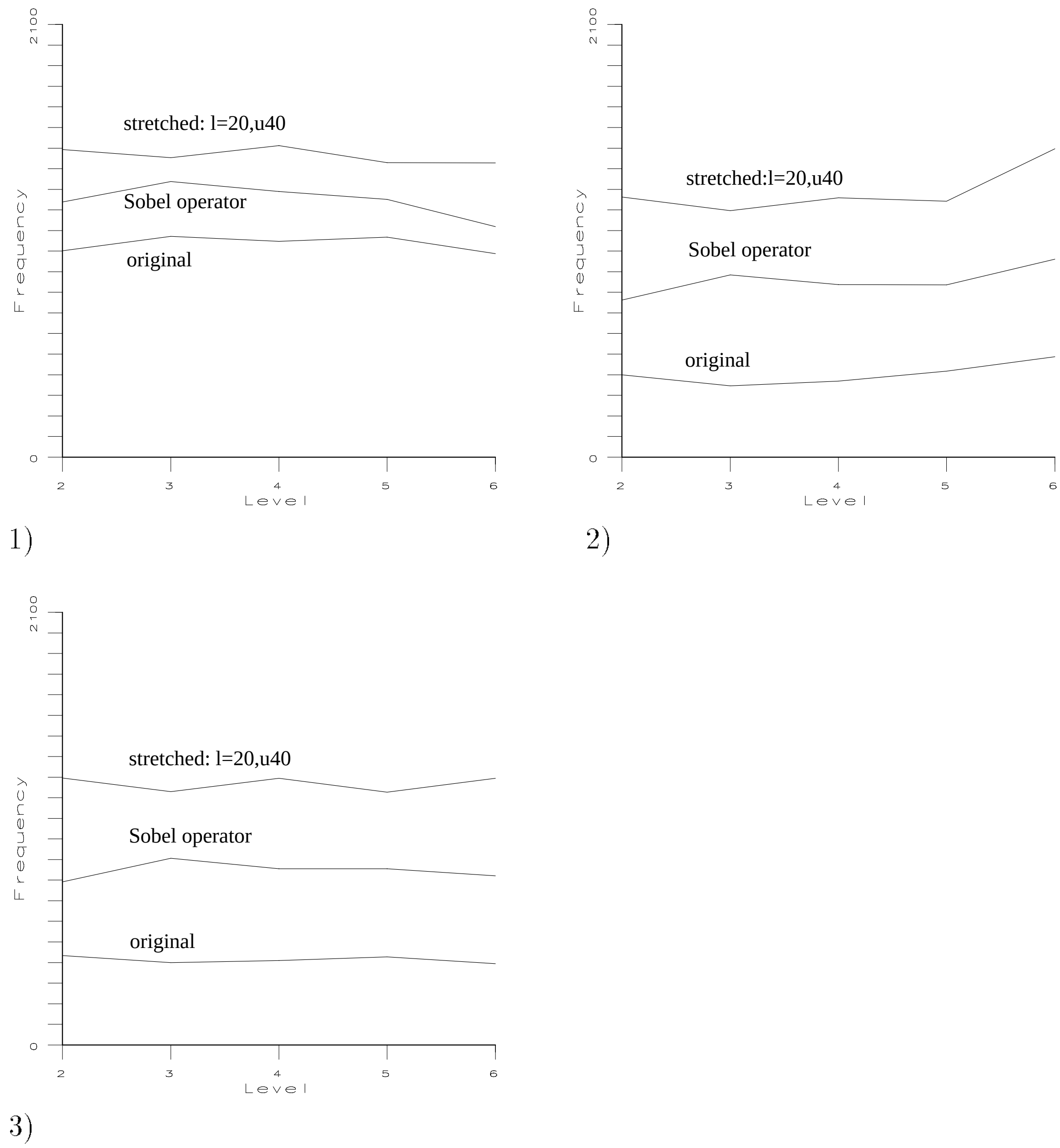
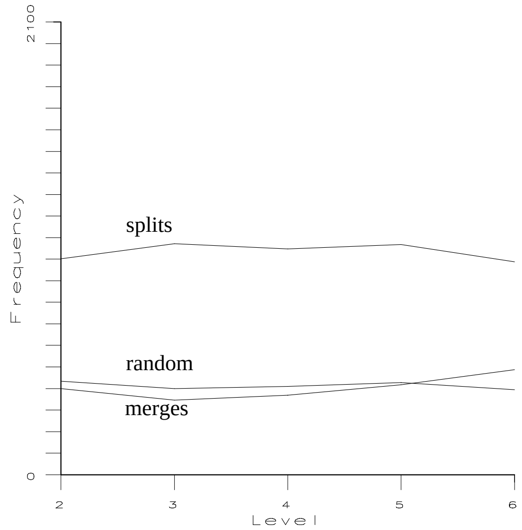
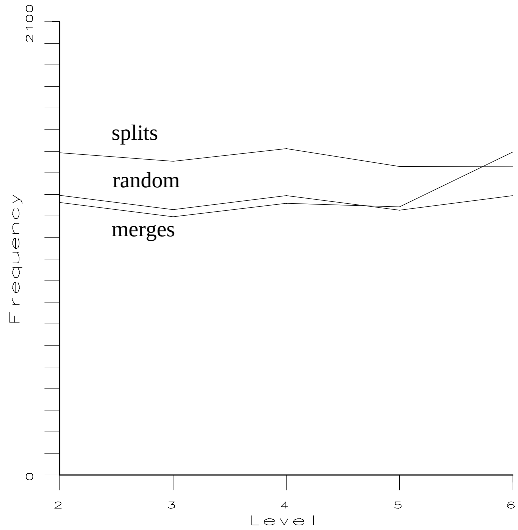


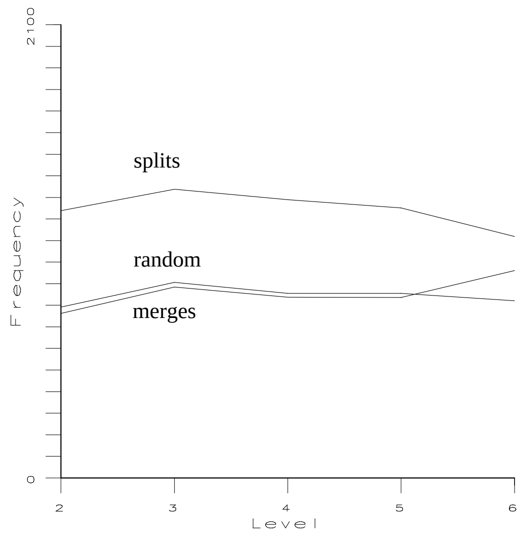
Figure 6.27: Comparison of configuration changes during segmentation of real images with Ω=300, α=.9; (1) blocks split, (2) neighborhoods merged, (3) total random splits and merges.



1)



2)



3)

Figure 6.28: Distribution of configuration changes during segmentation of a real images with Ω=300, α=.9; (1) original real ultrasound image, (2) enhanced real image with l=20,u=40, (3) convolution of the Sobel operator with the original real image.

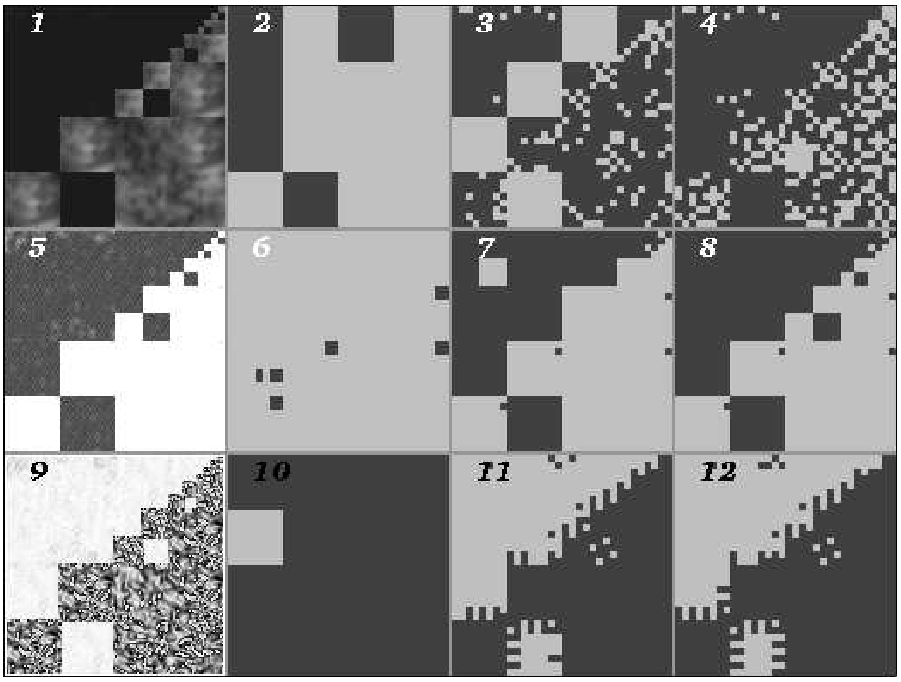


Figure 6.29: Segmentation of synthetic images, Ω=300, α=.9, (1) the original synthetic image, (2) f_i=10, 3) f_i=5 , 4) varying f_i, 5) enhanced image with l=20,u=40, 6) f_i=6, 7) f_i=23, 8) varying f, 9) convolution of Sobel operator with original image, 10) f_i=10, 11) f_i=9, 12) varying f_i.

a varying feature. Figure 6.29 shows the resulting segmentations produced for the original, enhanced, and convolved synthetic images. For the original and enhanced synthetic images, it is obvious that varying the feature produced the best results. Using the original synthetic image as input, mislabeling errors of 66.02% (f_i = 10), 66.6% (f_i = 5), and 29.98% (varying f_i) were produced. When the enhanced synthetic image was used, mislabeling errors of 50.98% (f_i = 6), 6.64% (f_i = 23) and 1.17% (varying f_i) were produced. Finally, with the synthetic image, which was convolved with the Sobel operator, mislabeling errors were 43.75% (f_i=10), 7.62% (f_i=9) and 8.5% (varying f_i). In Figure 6.30, the segmentations for the real ultrasound images all have qualitatively the most accurate segmentations when the feature is varied.

6.3.6 Discussion of Results

The purpose of Experiment 4 was to determine the most effective settings for the scheduling parameters α and Ω. By performing numerous experiments on both syn-

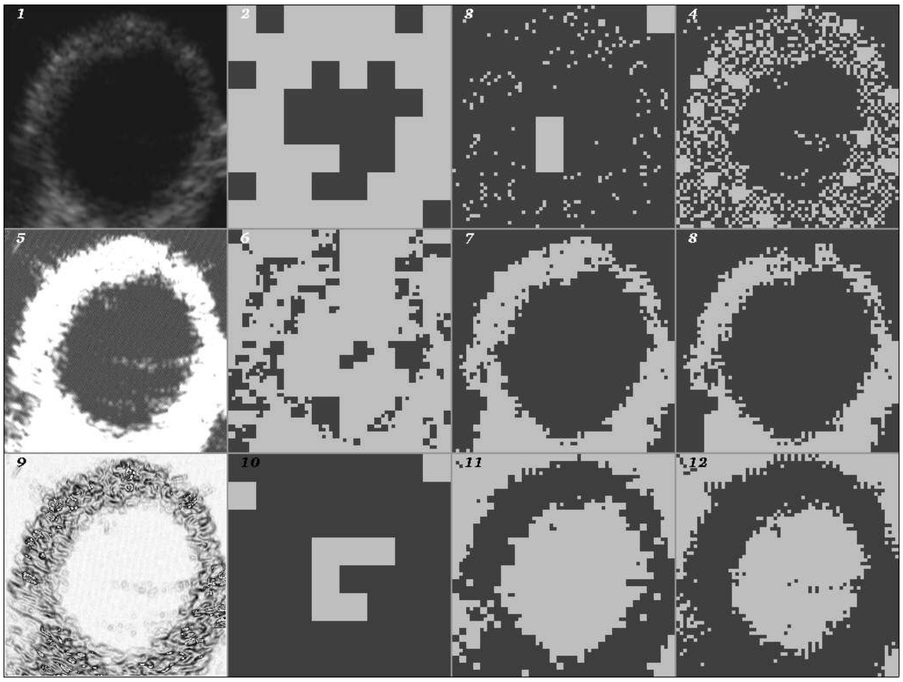


Figure 6.30: Segmentation of real ultrasound images, Ω=300,α=.9, 1)original image, 2)f_i=10, 3)f_i=5 , 4)varying f, 5)enhanced image with l=20,u=40, 6)f_i=6, 7)f_i=23, 8)varying f, 9)convolution of Sobel operator with original image, 10)f_i=10, 11)f_i=9, 12)varying f_i.

thetic and real images it was observed that a relatively slow rate of cooling ($\alpha =$ .99) consistently produced the most accurate segmentation results. The value chosen for Ω had little effect on the segmentation results as compared to the effect of α. It should be noted, however, that the largest images segmented were 256x256 pixels. For larger images it is probable that the effect of Ω will become more apparent since the number of blocks to examine will increase significantly. -1z Examination of the segmentation results for the synthetic and real images suggested that a schedule with a faster rate of cooling could be used without a significant loss in the accuracy of the segmentations. This small loss was traded for a substantial decrease in segmentation time. As a result $\alpha = .9$ and $\Omega = 300$ were used for the remaining experiments.

In Experiment 5, bias in the synthetic images towards the MTS algorithm was examined. The texture boundaries in synthetic images seemed to lie in the exact places which would produce the most favorable segmentations for the MTS algorithm. It was determined that the small amount of bias present in the synthetic images was not large enough to cause discrepancies between the results obtained for the synthetic images and the results obtained for the real ultrasound images. Thus, the original versions of the synthetic images were used in subsequent experiments without special consideration to bias.

The choice for the initial level at which to begin segmentation was examined in Experiment 6. Since the MTS algorithm examines pixel blocks at varying resolutions, the algorithm should produce the same segmentations independent of the initial level chosen. By segmenting the same image at different initial levels, it was found that the choice for the initial level did not affect the final segmentation results for almost all cases,.

The multiresolution mechanism of the MTS algorithm was further examined in Experiment 7. By fixing the feature detector for some of the segmentations, comparisons were made to determine if the use of a varying feature detector produced better results than those obtained with the fixed feature detector. Examination of the segmentation results for the real ultrasound images showed that by varying the feature detector, more significantly better segmentation results could be obtained.

6.4 Comparison of the MTS and CO Algorithms

The MTS and CO algorithms were evaluated in a common computing environment. Section 6.2 described this environment. The experiments used in comparing the MTS and CO algorithms, along with their results, are described in the following sections.

6.4.1 Implementation of the CO Algorithm

The CO algorithm was implemented using the partition model. As in [12], the neighborhood size η was set to 20, the penalty schedule λ was fixed with $\lambda_k = 0$ for the first ten iterations and increased by 1 for each subsequent iteration, and the same penalties for forbidden label configurations were used (see Section 4.3). The number of labels ρ was set to 2 (follicle or non follicle texture). The site visitation schedule μ was determined at each iteration by randomly ordering the sites in the label lattice with one site visited at a time. This schedule varies slightly from the one in [12] where the sites were visited in a raster scan fashion. The most important parameters for the CO algorithm were the thresholds $\{c_i\}$ $1 <= i <= m$, which control the sensitivity of the KS measure to differentiate among blocks (see Section 4.3). Since the CO algorithm uses a fixed block size (in [12] all experiments had β=21), it was necessary to determine the block size which produced the best segmentation results according to the types of images being segmented. This required a different set of thresholds for each block size. Note that this evaluation is not present at all in [12].

For the experiments, the thresholds were chosen as follows:

1. Fix the block size β.
2. Sample homogeneous texture blocks of size β from the images.
3. Compute the KS distance for all block pairs.
4. Update the histogram H with each KS distance.
5. Compute the cumulative distribution function, S, from histogram S and set the threshold to s_i, $s_i \epsilon S$ such that $\sum_{j=0}^{s_i} j = .97$.

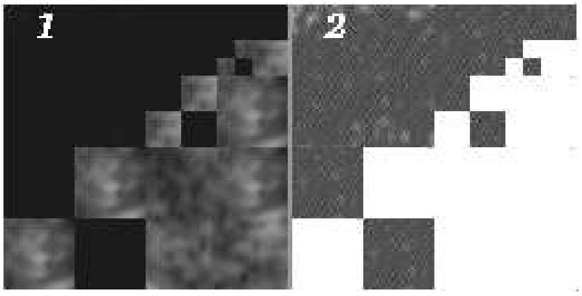


Figure 6.31: The test images: (1) original, (2) l=20,u=40.

Thus, no more than 3% of the KS distances for blocks from the same texture would exceed the threshold. As well as raw gray scale values, thresholds for the four transforms applied to the images in [12] were used in the experiments (m=5) for each fixed block size.

6.4.2 Experiments

Before a comparison of the MTS and CO algorithms could be made, it was necessary to perform a number of experiments with the CO algorithm. The first set of experiments was performed to determine how the parameters σ and β affect the performance of the CO algorithm as well as to determine which combination of these parameters produced the best segmentation results. The second set of experiments followed exactly the same procedure as in the first set except that real images were segmented.

6.4.3 Experiment 8: Synthetic Images

Figure 6.31 shows the two 128×128 synthetic images segmented for this set of experiments. These images were obtained in precisely the same manner as described in Section 6.3.1.

The images were segmented 20 times each with a different combination of (σ, β) where $\sigma \epsilon$ {3, 5, 7, 9,11} and $\beta \epsilon$ {5, 7, 13,15}. The resulting segmentations for each synthetic image are illustrated in Figures 6.32 and 6.33 in which σ is fixed along each row and β is fixed along each column. Figure 6.34 shows the mislabeling error produced for these segmentations. Figure 6.34(1) depicts the mislabeling error for varying block sizes when synthetic image shown in Figure 6.31 was segmented. The error varies as σ increases (resolution decreases). In Figure 6.34(2) the mislabeling error is shown for varying block sizes when the synthetic shown in Figure 6.31(2) was segmented. The amount of time required to segment the synthetic images is graphed in Figure 6.35. In Figure 6.35(1) the block size has a relatively small effect on the segmentation time for the synthetic image shown in Figure 6.31(1). However, for the synthetic image shown in Figure 6.31(2), Figure 6.35(2) shows that when σ is low there is quite a difference in the times for $\beta = 5$ and $\beta = 13$.

Examination of the resulting segmentations, mislabeling error, and segmentation time led to the choice of $\sigma = 3$ and $\beta = 13$ as the best parameter setting for the set of synthetic images tested. The greater amount of time needed (7358 seconds) to segment the synthetic image shown in Figure 6.31(2) had to be accepted in order to achieve a reasonable mislabeling error (9.73%) in the segmentation.

6.4.4 Experiment 9: Real Images

The purpose of Experiment 9 was to determine if real images could be segmented with the same quality and in the same amount of time as the synthetic images.

Using the same procedure as in Experiment 8, the real images in Figures 6.4(1) and 6.4(6) were segmented. The resulting segmentations for these images are shown in Figures 6.36 and 6.37. Note that $\sigma = 6$ in the first row. This is different from Experiment 8 in which σ took on the values 3 and 5 in the first two rows (see Figures 6.32 and 6.33). The larger value of σ (6) was selected because the larger real images (256×256) took significantly longer to segment. Figure 6.38 shows the segmentation times for the real images at varying block sizes. Figure 6.38(1) depicts the segmentation times for the real image in Figure 6.4(1). Here, as was the case in Experiment 8, the block size has a relatively small effect on the segmentation times. In Figure 6.38(2), the segmentation times for real image 6.4(2) are shown. The block size has a more profound effect on the segmentation times with larger blocks ($\beta = 13$, $\beta = 15$)

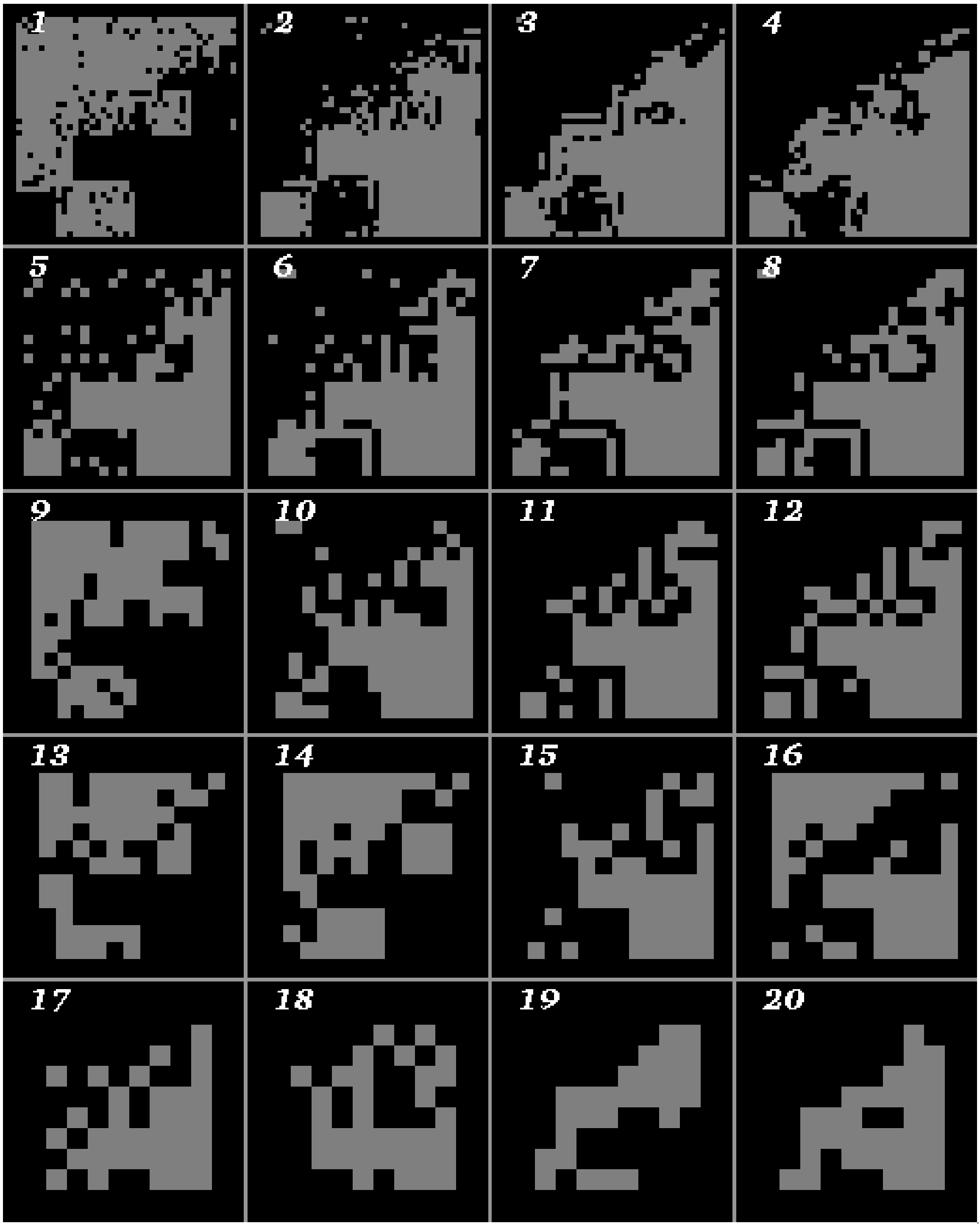


Figure 6.32: Segmentation of the original test image with σ=3 and (1) β=5, (2) β=7, (3) β=13, (4) β=15, σ=5 and (5) β=5, (6) β=7, (7) β=13, (8) β=15, σ=7 and (9) β=5, (10) β=7, (11) β=13, (12) β=15, σ=9 and (13) β=5, (14) β=7, (15) β=13, (16) β=15, σ=11 and (17) β=5, (18) β=7, (19) β=13, (20) β=15.

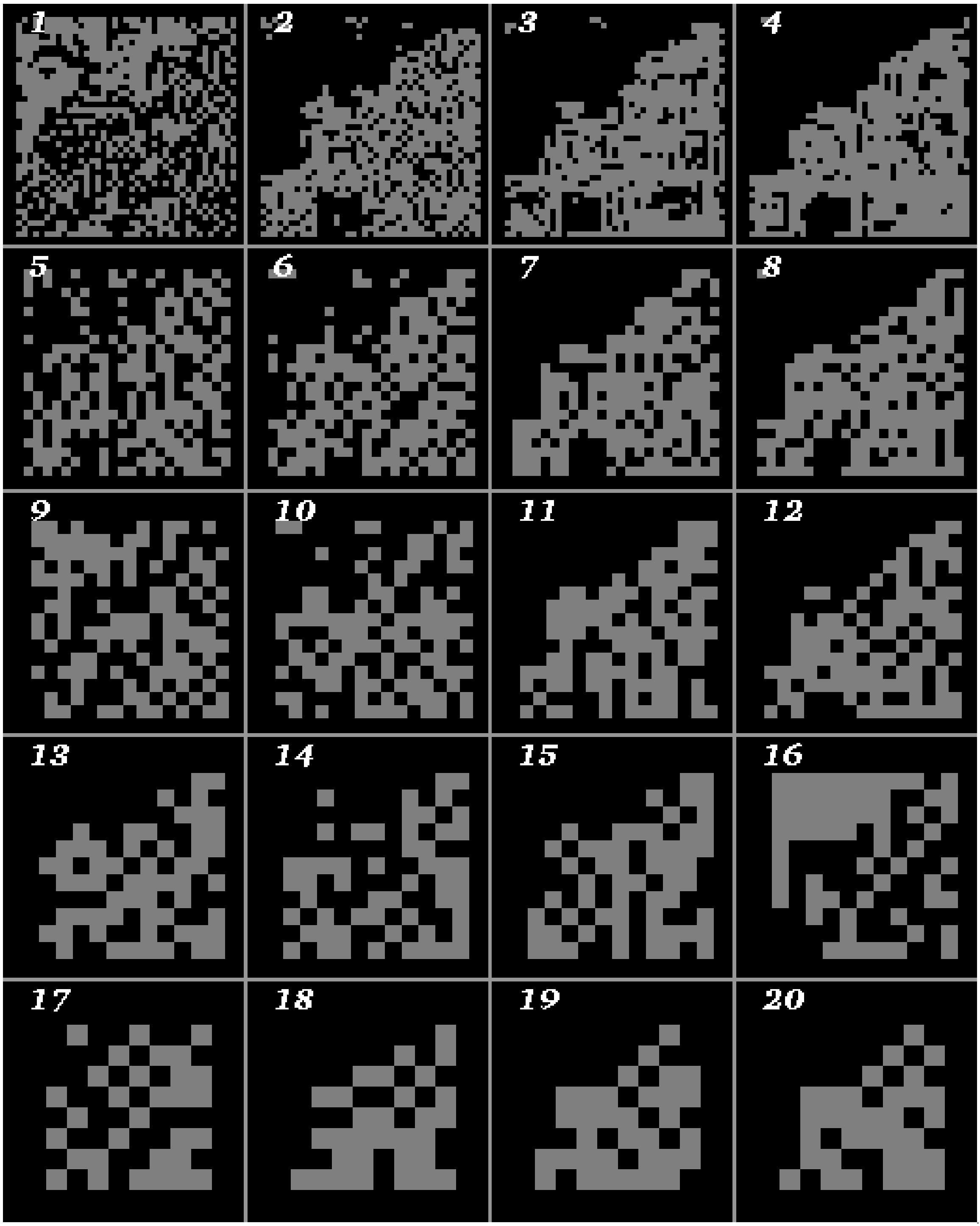


Figure 6.33: Segmentation of the test image with l=20, u=40, σ=3 and (1) β=5, (2) β=7, (3) β=13, (4) β=15, σ=5 and (5) β=5, (6) β=7, (7) β=13, (8) β=15, σ=7 and (9) β=5, (10) β=7, (11) β=13, (12) β=15, σ=9 and (13) β=5, (14) β=7, (15) β=13, (16) β=15, σ=11 and (17) β=5, (18) β=7, (19) β=13, (20) β=15.

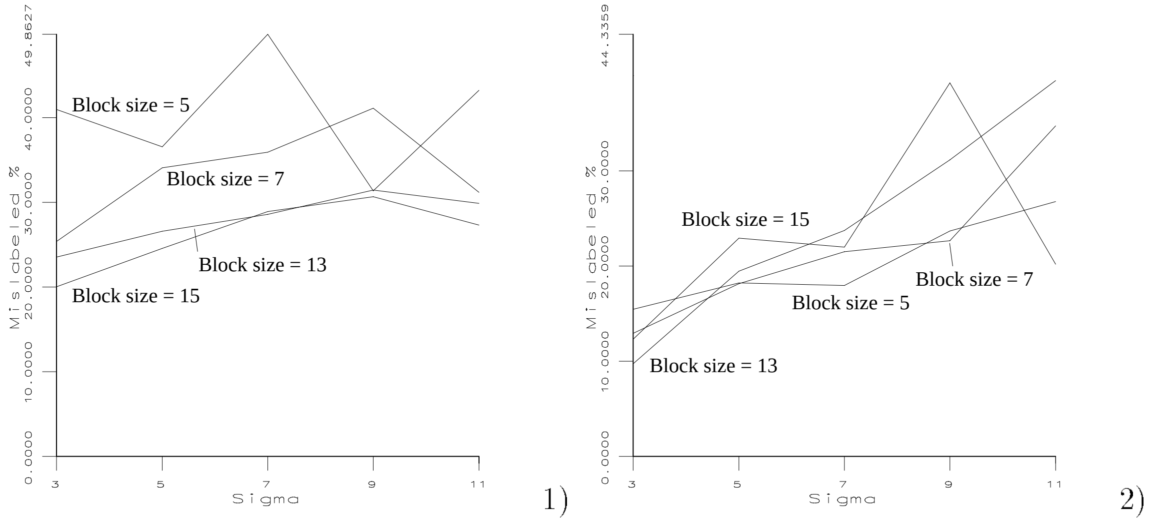


Figure 6.34: Mislabeling error for test image: (1) original test image, (2) l=20,u=40.

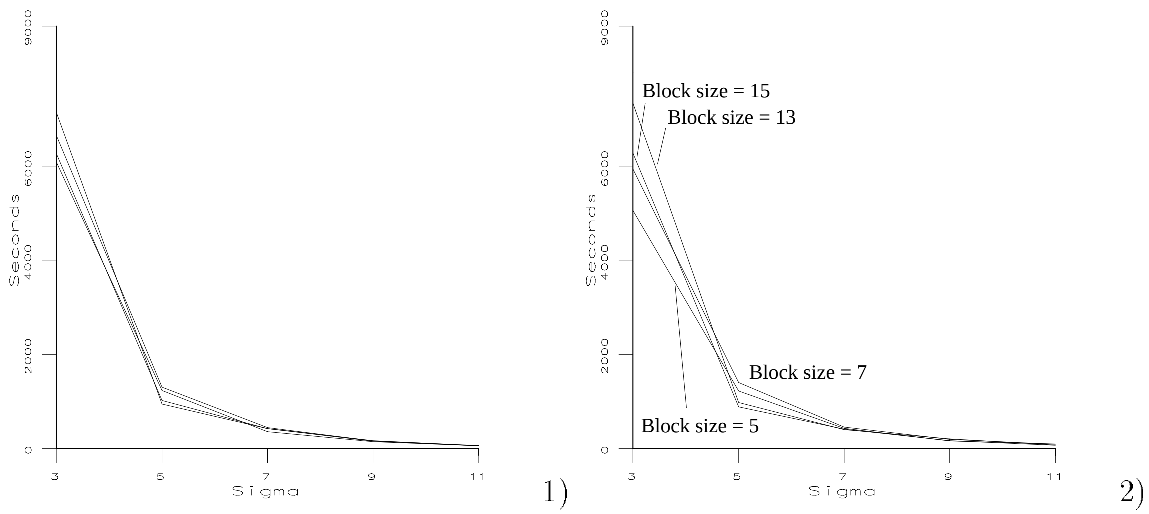


Figure 6.35: Time to segment test image: (1) original test image, (2) l=20,u=40.

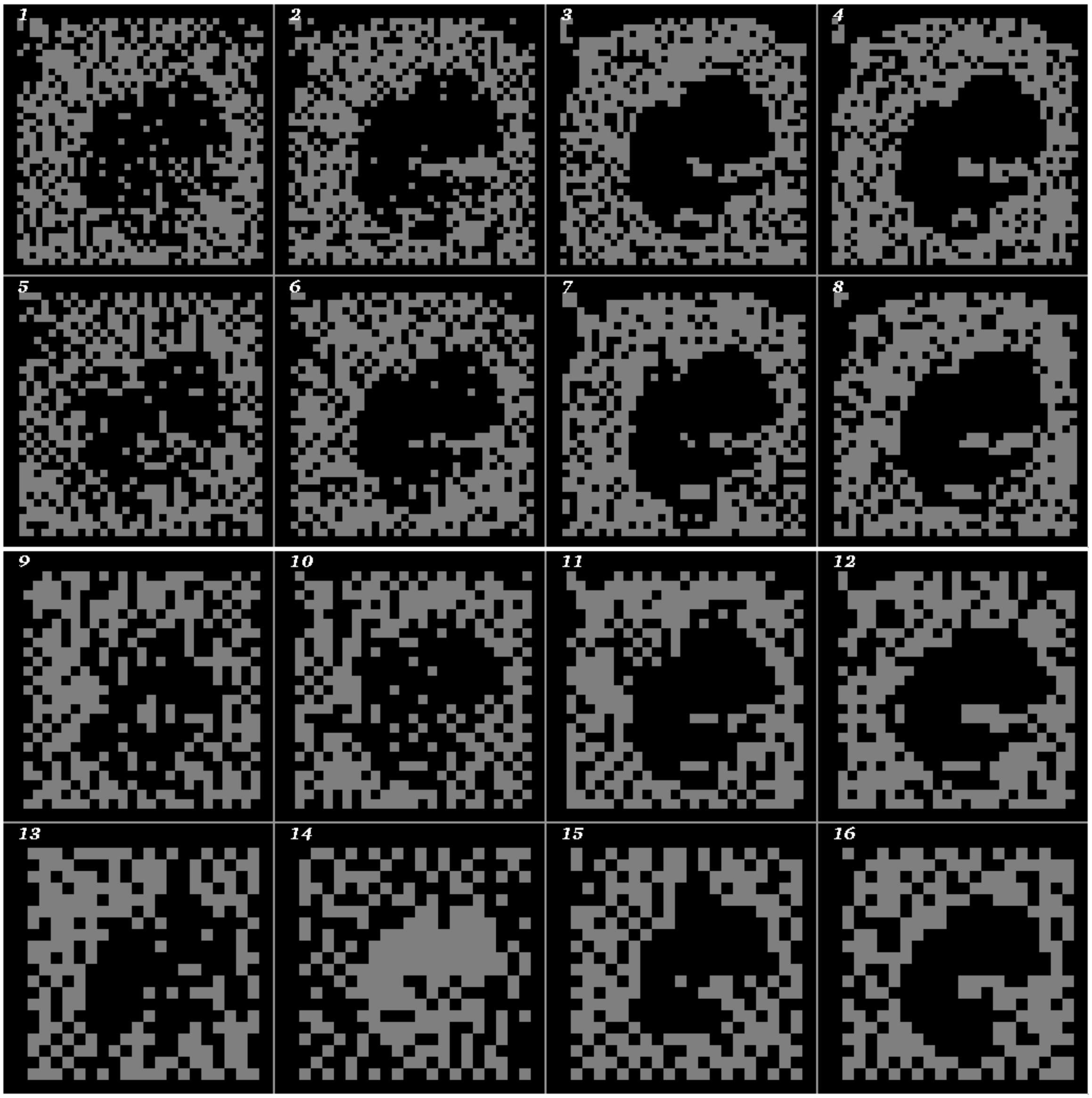


Figure 6.36: Segmentation of the original real image with σ=6 and (1) β= 5, (2) β= 7, (3) β= 13, (4) β= 15, σ=7 and (5) β= 5, (6) β= 7, (7) β= 13, (8) β= 15, σ=9 and (9) β= 5, (10) β= 7, (11) β= 13, (12) β= 15, σ=11 and(13) β= 5, (14) β= 7, (15) β= 13, (16) β= 15,

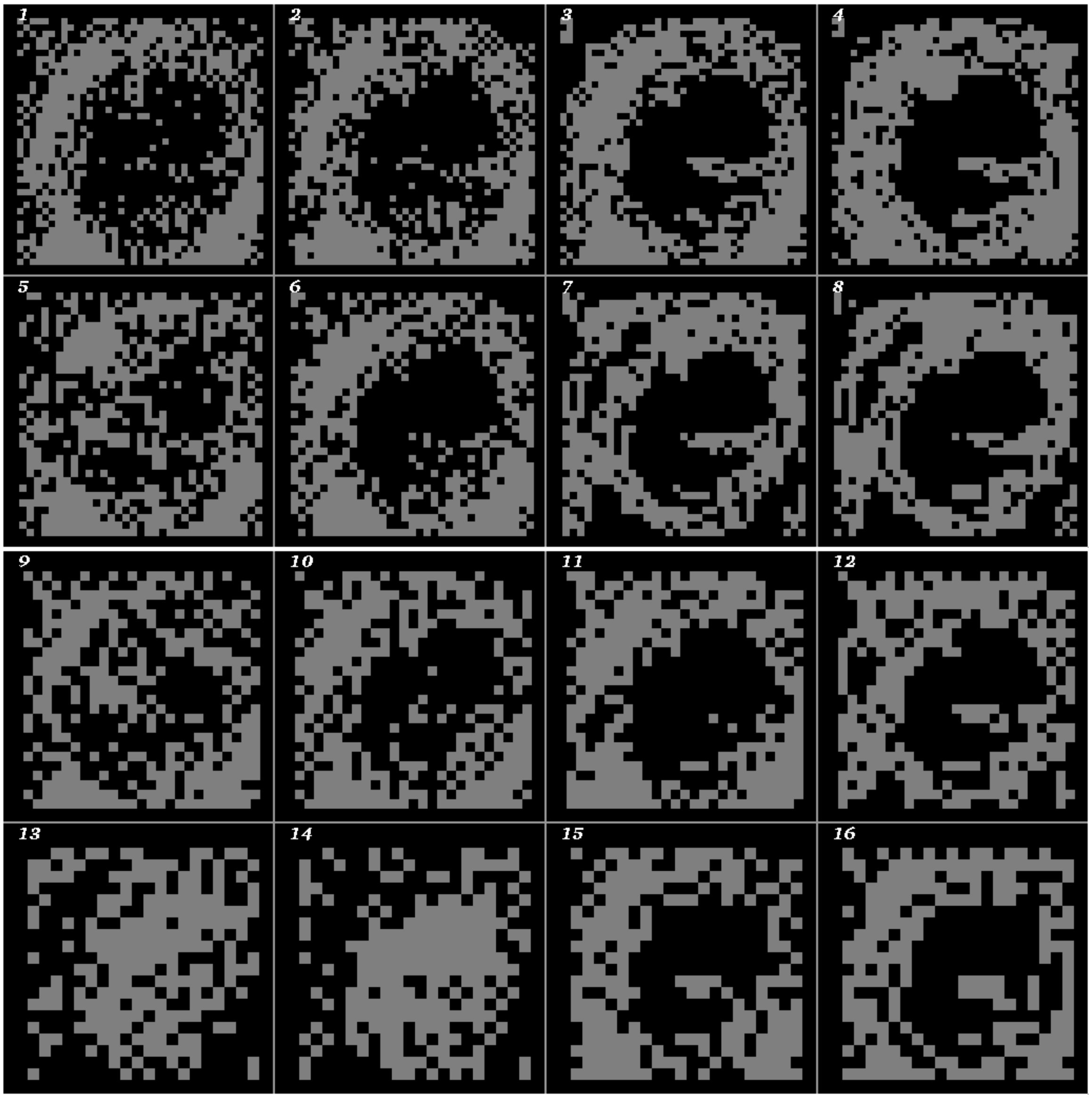


Figure 6.37: Segmentation of the real image with l=20, u=40, σ=6 and (1) β= 5, (2) β= 7, (3) β= 13, (4) β= 15, σ=7 and (5) β= 5, (6) β= 7, (7) β= 13, (8) β= 15, σ=9 and (9) β= 5, (10) β= 7, (11) β= 13, (12) β= 15, σ=11 and(13) β= 5, (14) β= 7, (15) β= 13, (16) β= 15,

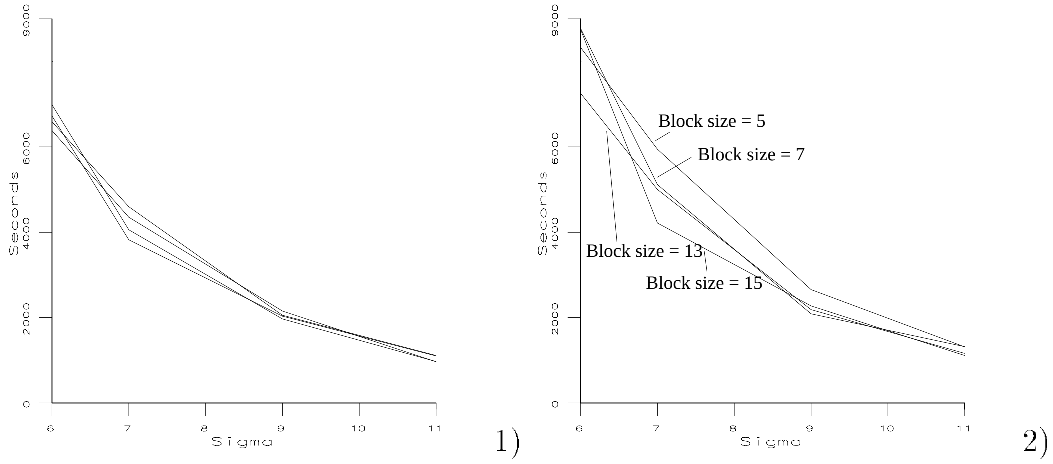


Figure 6.38: Time to segment real image: (1) original real image, 2)l=20,u=40.

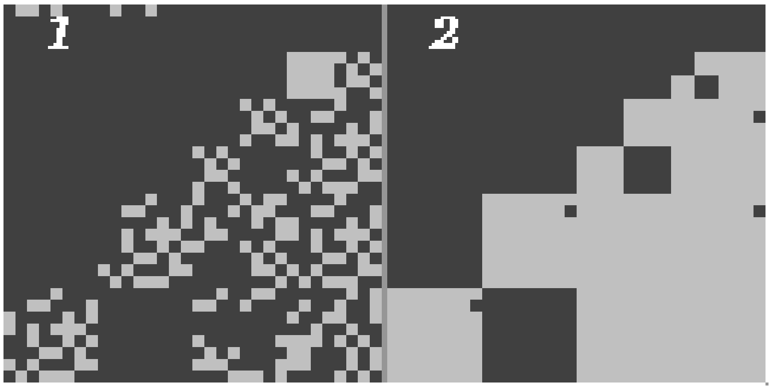


Figure 6.39: Resulting segmentations from the MTS algorithm with $\Omega = 300$, $\alpha = .9$ for (1) the original synthetic image, (2) the synthetic image with l=20,u=40.

requiring less segmentation time.

6.4.5 Discussion of Results

For the MTS algorithm, both synthetic images in Figure 6.31 were segmented with $\Omega = 300$ and $\alpha = .9$. The resulting segmentations are illustrated in Figure 6.39. A mislabeling error of 31.20% was produced for the synthetic image in Figure 6.31(1) while a mislabeling error of 1.16% was produced for the synthetic image in 6.31(2). With $\sigma = 3$ and $\beta = 13$, the mislabeling error produced by the CO algorithm was 23.54% for Figure 6.31(1) and 9.73% for Figure 6.31(2). The CO algorithm had less error for Figure 6.31(1) but more error than the MTS algorithm for Figure 6.31(2). However, since both synthetic images differ only by a linear difference in pixel intensities (thus having the same ground truth), the MTS algorithm can be judged more accurate because it located all texture regions with only 1.16% error (98.84% accuracy).

Examination of the segmentation times for the MTS and CO algorithms with the above parameter settings shows the superiority of the MTS algorithm. The MTS algorithm required 199 seconds to segment the synthetic image in Figure 6.31(1) and 118 seconds to segment synthetic image in Figure 6.31(2). The CO algorithm required 6291 seconds to segment Figure 6.31(1) and 7358 seconds to segment Figure 6.31(2). In both cases, The MTS algorithm produced segmentations in a fraction of the time needed by the CO algorithm. For the real images segmented in Experiment 9, The MTS algorithm segmented the images in Figures 6.4(1) and 6.4(6) in 438 and 103 seconds while the CO algorithm (with $\beta = 6$) required 6723 and 7257 seconds respectively. It is obvious that the multiresolution approach used in the MTS algorithm significantly reduces the number of blocks that have to be examined during the segmentation process, thus requiring less time to segment an image.

6.5 Summary

Numerous experiments were performed with the MTS algorithm to determine its ability in segmenting both synthetic and real images. In Experiment 4 it was found that the scheduling parameters α and Ω could be chosen such that the segmentations produced were accurate and the time required to produce the segmentations was small. In Experiment 5 it was found that the synthetic images used in the experiments were not significantly biased towards the MTS algorithm while in Experiment 6 it was found that the initial level at which the segmentation was started did not affect the final segmentation. In Experiment 7 it was determined that the multiresolution mechanism performed better when varying the feature detector rather than fixing the feature detector for different resolutions.

Experimentation with the CO algorithm and its parameter settings suggested that for the set of images segmented, there was an optimal block size (13x13) which produced a good segmentation in a consistent amount of time. As well, the accuracy of the segmentations increased as the resolution at which blocks were examined increased (i.e. low σ values). However, as the resolution increased the segmentation time also increased. Thus, producing a segmentation with a low mislabeling error required a significant amount of time.

Examining the segmentation results from the MTS and CO algorithms implies a number of conclusions. First, the lower mislabeling error achieved by the MTS algorithm can be attributed to the fact that it makes better use of *a priori* knowledge than the CO algorithm. The feature table (see Section 5.4) characterizes each unique texture at varying block sizes while the CO algorithm uses a threshold at a fixed block size to signal non-uniqueness among different textures. Second, the amount of time required to segment images is related to the number of blocks that have to be examined during the segmentation process. Because of the multiresolution approach the MTS algorithm characteristically examines far fewer blocks than the fixed number of blocks examined by the CO algorithm. As the desired level of accuracy for segmentation is increased, the difference in the number of blocks examined by each algorithm is quite significant. Thus, the CO algorithm will require much more time than the MTS algorithm to segment an image. Conversely, the CO algorithm produces a much higher mislabeling error when segmenting an image in the same amount of time as the MTS algorithm. This is because blocks have to be examined at a lower resolution which results in the higher mislabeling error.

Chapter 7

3D Reconstruction

This chapter describes how the segmentations produced by the MTS algorithm are used in creating a 3D model of an ovarian follicle. Since 3D reconstruction is not the emphasis of this thesis, a relatively simple but effective method is used to reconstruct the segmented images in 3D. The method is effective in the sense that it allows different 3D perspectives of the follicle to be viewed. This provides much more information about the shape of the follicle than the 2D cross sections can provide. The 3D model is based on a sequence of 18 256×256 ultrasound images as illustrated in figure 7.1 (Section 6.1 explains how a sequence of ultrasound images are obtained). The images were preprocessed by convolving each image with the Sobel operator as in Section 6.3.1. The resulting segmentations are shown in Figure 7.2.

7.1 Obtaining the Contour

Once the ultrasound images are segmented, the next step is to determine which region represents the follicle. Since the follicle appears as a contiguous region in the central portion of each image, a contour around the follicle can be obtained by scanning along a middle row from a point not in the follicular region until a change in pixel intensity is reached. At this point, tracing around the contiguous region is initiated. The contour is complete when the initial point is again reached. From the seventh segmented image in Figure 7.2, Figure 7.3 shows the smoothed contour

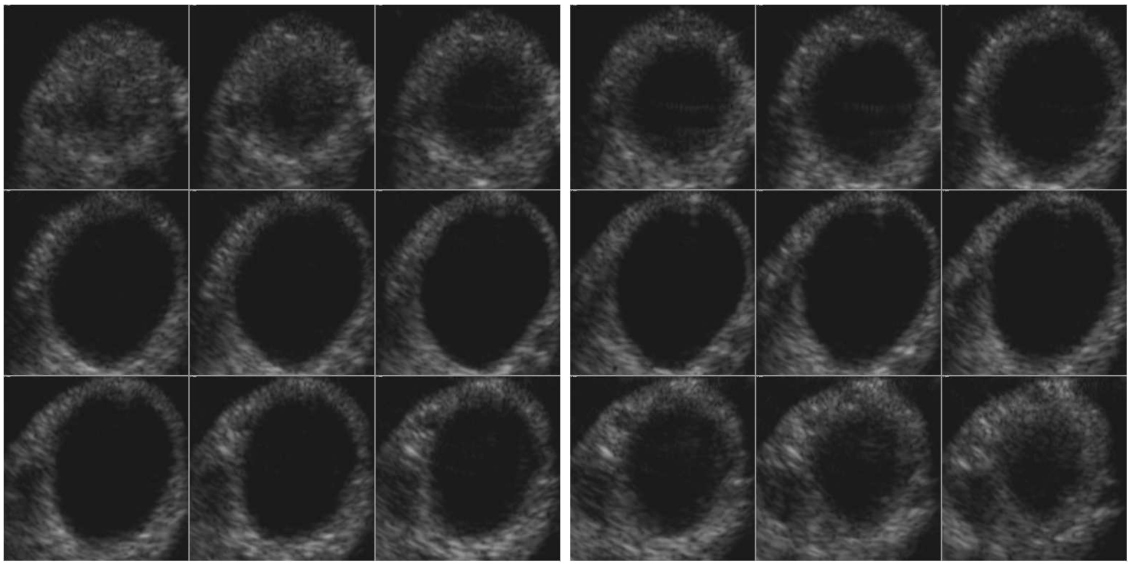


Figure 7.1: The sequence of 18 ultrasound images scanned at 1mm intervals.

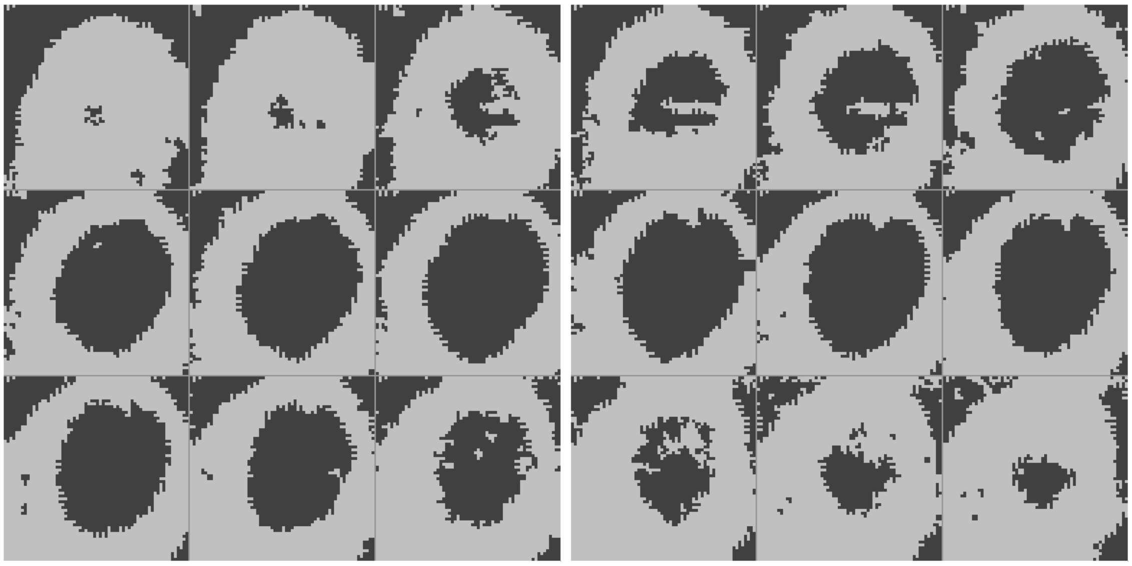
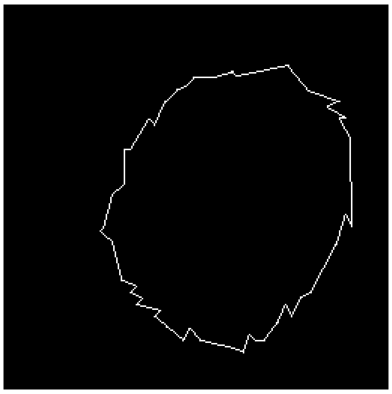
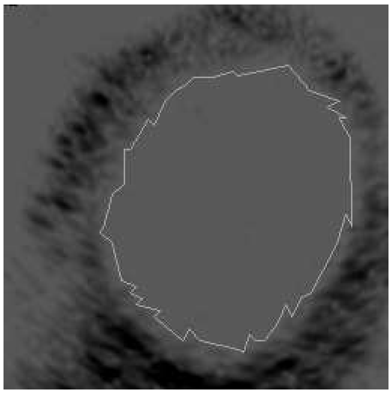


Figure 7.2: The resulting segmentations for a sequence of ultrasound images.



1)



2)

Figure 7.3: (1) Contour obtained by tracing the follicular region followed by smoothing, (2) the contour superimposed on the original ultrasound image. The image is inverted and darkened to emphasize the tissue surrounding the follicle.

using the dynamic strip algorithm developed by Leung and Yang [27]. Each of the 18 segmented ultrasound images was processed in this way to obtain a series of 2D contours which could then be stacked beside each other to produce the 3D model.

7.2 Stacking Method

The purpose of stacking is to combine a series of 2D contours in such a way that information about a third dimension can be inferred. A simple stacking method is used to produce the 3D model described in this section. First, the center of gravity is obtained from one of the 2D contours. This point is used as the center of gravity for all contours (based on the assumption that the all the images are scanned along the same axis). Each contour is then translated in the XY plane so that the center of gravity is at the origin.

A ray is projected along the X axis until it intercepts a point on the contour. By performing this on each contour, the common starting point (x_{z0},y_{z0}) is used to connect the corresponding point on each contour along the Z axis. Corresponding points are then sampled from each contour at frequency N. A contour is divided into N equal segments and the points on the contour at the end of each segment are used as the sample points. For the 3D follicle in Figures 7.4-7.6, N = 75 resulting in 1350 points describing the shape of the ovarian follicle in the XYZ planes.

The distance at which contours were stacked along the Z axis was arbitrarily set to z equal units. Although the ultrasound images were sampled at 1 mm intervals, the dimensions of the 2D contours were not available. Thus, it was impossible to determine the correct scaling along the Z axis with respect to the XY plane. The value for z was chosen so that the resulting 3D model would have roughly an oval shape.

7.3 Surface Reconstruction

Surface reconstruction of the follicle was accomplished in two ways. One method involved the use of a 3D plotting package to plot the points directly. Figures 7.4-7.6 show the resulting 3D perspectives of the follicle based on the 3D model described in the previous section. In these figures, the surface points between contours are interpolated by the plotting package.

The second way in which surface reconstruction of the follicle was accomplished was with a ray tracing program. With ray tracing, a complete description of all object surfaces is required. Therefore, it was necessary to interpolate the surface between contours. Two triangles can be defined between two corresponding pairs of points on two adjacent contours. By representing the surface between adjacent pairs of contours in this manner, the entire surface of the follicle was obtained. The coordinates of these triangles were then used by the ray tracing program to produce a shaded perspective projection of the follicle. Figure 7.7 illustrates four different views of the follicle. Because of the low sampling rate of the contours, $N = 50$, the surface of the follicle appears quite rough and the triangular interpolation between the contours is quite apparent.

Figure 7.4: 3D plots of follicle with N=75, *Top*: contour profile along Z axis, *Bottom*: same contours at different viewing angle.

Figure 7.5: 3D plots of follicle with N=75, *Top*: shaded surface, *Bottom*: wire mesh.

Figure 7.6: 3D plots of follicle with N=75, *Top*: bottom half of follicle, *Bottom*: exploded view of follicle top, middle and bottom.

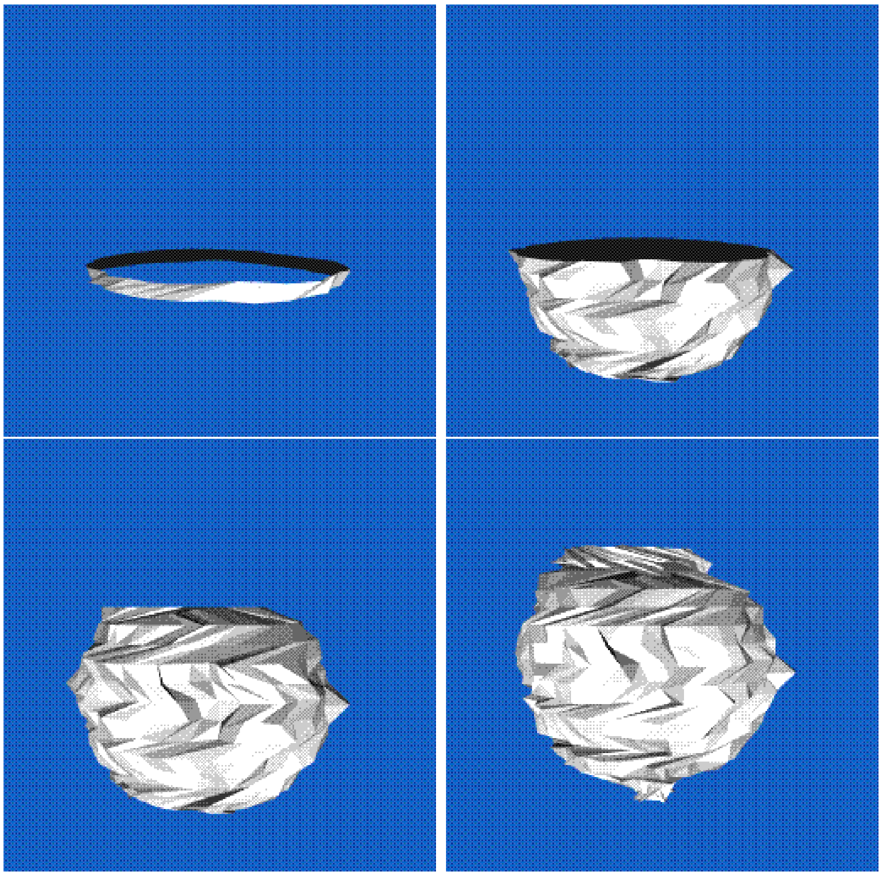


Figure 7.7: Perspective views of the follicle produced by a ray tracing program for N=50: (1) contours 8-9, (2) contours 3-12 , (3) contours 3-17, (4) all 18 contours.

Chapter 8

Contributions and Future Work

This chapter discusses the contributions made by this thesis as well as the future work resulting from this thesis. A summary of the thesis is also presented.

8.1 Contributions

A number of contributions are made by the work in this thesis. These contributions are discussed in the following paragraphs.

The first contribution of this thesis is the development of a framework for the evaluation of features in characterizing texture. The method of texture characterization used in the MTS algorithm allows the evaluation of any set of features in characterizing texture at varying resolutions. The relationship between the performance of the choice of features and the size of the texture blocks on which the features are used is determined. By ranking the features according to their ability to characterize texture at each resolution, the framework shows the importance of the relationship between feature and texture resolution. No previous work has addressed this relationship.

The second contribution of this thesis is a new texture segmentation algorithm, the Multiresolution Texture Segmentation (MTS) algorithm, which segments images at varied resolutions. The algorithm can also segment images corrupted by noise. By using *a priori* knowledge about the textures that exist in the images being segmented, as well as using only the best feature at each resolution, the MTS algorithm has been shown to produce more accurate results in less time than a current texture segmentation algorithm, the Constrained Optimization (CO) algorithm.

The third contribution is the evaluation of the CO algorithm. By performing numerous experiments on the CO algorithm and by comparing its performance to the MTS algorithm, the thesis has validated the work of other researchers in the field.

The fourth contribution of this thesis is the characterization of the textures in ultrasound images. The thesis identified two different classes of textures; the texture of the follicular region and the texture of the surrounding tissue.

Finally, the fifth contribution is that this thesis represents a first attempt at applying texture segmentation to ultrasound images. More significantly, the 3D model of the ovarian follicle, based on a series of cross sectional ultrasound images, provides a non-destructive means of examining the follicle in much more detail than that available in a single 2D cross section.

8.2 Thesis Summary

In Chapter 2, background on ultrasound imaging was provided. An overview of the ultrasound technique was described as well as some of the difficulties encountered when trying to construct an accurate 3D model from a series of diagnostic, cross sectional ultrasound images.

Chapter 3 provided a survey of image segmentation techniques. Characteristics of edge detection were identified in the examination of approaches using edge detection. Since the edge detection approaches were shown to be susceptible to noise in images texture discrimination approaches were also examined. The advantages of using a multiresolution approach in segmenting an image based on texture, were discussed. Because of these advantages, it was determined that a segmentation algorithm using the multiresolution approach would be effective in segmenting a series of characteristically noisy ultrasound images.

Theoretical background for two texture segmentation algorithms, the Constrained Optimization (CO) algorithm [12] and the Multiresolution Texture Segmentation

(MTS) algorithm, was provided in Chapter 4. Both simulated annealing and constrained optimization were presented. The CO algorithm was also described and its major weaknesses pointed out; first, the CO algorithm can not adapt to varying size texture regions, second, the CO algorithm performs unnecessary label computations requiring more time to segment an image. The MTS algorithm was developed in this thesis to overcome the weakness in the CO algorithm. Based on the multiresolution approach used in the MTS algorithm, hypotheses about the algorithm's ability to produce a faster, more accurate segmentation than the CO algorithm were made and tested.

The use of texture characterization in the MTS algorithm was discussed in Chapter 5. 32 spatial and frequency domain features were defined. These features were based on both the raw pixel intensities of the textures as well as transformations of the raw intensities. The relationship between features and the size (resolution) of the texture blocks that they are applied was also examined. Because of the inherent variability within the same types of texture in most real images the MTS algorithm uses only the features which can best characterize texture classes at each different resolution, rather than trying to find an optimal feature independent of the resolution. As well, the use of outlier detection, rather than texture classification, was discussed as a method of determining the effectiveness of features in characterizing different texture classes.

Experiments were performed to evaluate the ability of the 32 features in characterizing real textures sampled from a series of ultrasound images. The experiments produced a number of results. The first result found that there were two consistently distinguishable texture classes present in the ultrasound images. Second, it was found that the 32 features produced results independent of the 0, 90, 180, and 270 degree orientations of the textures. It was necessary to consider the case where the orientation of the textures was varied among the four orientations within the representative sample. However, since the texture orientation among the ultrasound images is constant, it was not necessary to consider the effects of orientation on the resulting feature rankings for the different texture classes.

Finally, it was found that by preprocessing the representative samples it was possible to reduce the number of detected outliers, as well as the number of overlapping regions for the 32 features at small sample resolutions. It was also found that when the representative samples were preprocessed by stretching a range of pixel intensities, the frequency domain features dominated the top rankings for each resolution. When the representative samples were convolved with the Sobel edge detector the spatial domain features dominated the top rankings for each resolution.

In Chapter 6, numerous experiments were performed with the MTS algorithm to determine its ability in segmenting both synthetic and real images. Experiment 4 found that the scheduling parameters α and Ω could be chosen such that the segmentations produced were accurate and the time required to produce the segmentations was small. Experiment 5 found that the synthetic images used in the experiments were not significantly biased towards the MTS algorithm while Experiment 6 found that the initial level at which the segmentation was started did not affect the final segmentation. Experiment 7 determined that the multiresolution mechanism performed better when varying the feature rather than fixing the feature for different resolutions.

Experimentation with the CO algorithm and its parameter settings suggested that for the set of images segmented, there was an optimal block size (13x13) which produced a good segmentation in a consistent amount of time. As well, the accuracy of the segmentations increased as the resolution at which blocks were examined increased (i.e. low σ values). However, as the resolution increased the segmentation time also increased. Thus, producing a segmentation with a low mislabeling error required a significant amount of time.

Examination of the segmentation results from the MTS and CO algorithms implied a number of conclusions. First, the lower mislabeling error achieved by the MTS algorithm was attributed to the fact that it made better use of *a priori* knowledge than the CO algorithm. Second, because of the multiresolution approach the MTS algorithm characteristically examined far fewer blocks than the fixed number of blocks examined by the CO algorithm. This resulted in faster segmentation times for the MTS algorithm. Conversely, the CO algorithm produced a much higher mislabeling error when segmenting an image in the same amount of time as the MTS algorithm.

Chapter 7 described the steps required to produce a 3D model of an ovarian follicle based on a series of 2D cross sectional ultrasound images. First, the series of images is segmented to determine the follicular region in each of the images. Next, the follicular region in each segmented image is traced to produce a contour. The set of contours obtained can then be smoothed slightly to remove some of the jaggedness resulting from the segmentation process. The smoothed contours are then stacked on top of each other, along the Z axis at a given interval, to obtain the third dimension for the follicle. By sampling each contour at a common number of points, a 3D set of points is used to describe the 3D follicular model. This set of points is used by a plotting program to produce shaded and meshed surface perspectives of the follicle. The set of points is also transformed into a triangular surface description of the follicle which a ray tracing program uses to produce different 3D perspectives of the follicle.

8.3 Future Work

The advent of new approaches instills the need for future work. The research accomplished in this thesis presents a number of points which require closer examination.

The feature characterization process discussed in Chapter 5 could be researched in more detail. First, the selection of features could be improved so that preprocessing of the representative sample would not be necessary. Also, it may be possible to find a set of features that are able to determine more than two texture classes in the ultrasound images. Second, the mapping of the feature measures into the feature space could be improved. Currently, a simple one dimensional range is used to indicate the portion of the feature space which is occupied by the measures for each feature at each resolution. Third, different ranking schemes used to rank the features at each resolution could be investigated. The ranking scheme most certainly would have to be modified if a different feature space mapping is used.

For the segmentation component of the MTS algorithm, future work could involve the application of a penalty function in the relaxation process. By limiting the formation of certain regions (such as very narrow regions or small isolated regions) during the relaxation process, it may be possible to increase the accuracy of the final segmentation. Also, better treatment of unknown blocks could be incorporated into the segmentation component. Currently, the identity of each unknown block is determined by assigning it to the texture class having the closest measure to its own. This is performed only after all splitting and merging has been completed.

Future work could also involve evaluation of the texture characterization and segmentation frameworks with different types of real textures and images. This would serve as a point of verification for the general application of the texture characterization and segmentation frameworks.

References

[1] H.J. Antonisse. Image segmentation in pyramids. *Computer Graphics, and Image Processing*, 19:367–383, 1982.

[2] S. Batnitzky, H.I. Price, P.N. Cook, L.T. Cook, and S.J. Dwyer III. Three-dimensional computer reconstruction from surface contours for head ct examinations. *Journal of Computer Assisted Tomography*, 5(1):60–67, 1981.

[3] C. Bouman and B. Liu. Multiple resolution segmentation of textured images. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 13(2):99–113, 1991.

[4] R.N. Bracewell. *The Fourier Transform and Its Applications.* McGraw-Hill, 1986.

[5] P. Brodatz. *Textures: A Photographic Album for Artists and Designers.* Reinhold, New York, 1968.

[6] V. Cantoni and S. Levialdi. *Pyramidal Systems for Computer Vision*, pages 215–288. Springer-Verlag, 1986.

[7] G.R. Dattatreya and L.N. Kanal. Detection and smoothing of edge contours in images by one-dimensional kalman techniques. *IEEE Transactions On Systems, Man, and Cybernetics*, 20(1):159–165, January 1990.

[8] J.M.H. du Buf, M. Kardan, and M. Spann. texture feature performance for image segmentation. *Pattern Recognition*, 23(3/4):291–309, 1990.

[9] P.H. Eichel and E.J. Delp. Quantitative analysis of a moment-based edge operator. *IEEE Transactions On Systems, Man, and Cybernetics*, 20(1):59–66, January 1990.

[10] K.B. Eom and R.L. Kashyap. Composite edge detection with random field models. *IEEE Transactions On Systems, Man, and Cybernetics*, 20(1):81–93, January 1990.

[11] K.R. Erikson, F.J. Fry, and J.P. Jones. Ultrasound in medicine-a review. *IEEE Transactions On Sonics and Ultrasonics*, 21(3):144–170, July 1974.

[12] D. Geman, S. Geman, C. Graffigne, and P. Dong. Boundary detection by constrained optimization. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 12(7):609–628, July 1990.

[13] R.C. Gonzalez and P. Wintz. *Digital Image Processing*, chapter 7. Addison-Wesley Publishing Company Inc., 1977.

[14] R.C. Gonzalez and P. Wintz. *Digital Image Processing*, pages 36–88. Addison-Wesley Publishing Company Inc., 1977.

[15] D. Gordon and R.A. Reynolds. Image space shading of 3-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 29:361–376, 1985.

[16] C.C. Gotlieb and H.E. Kreyszig. Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics, and Image Processing*, 51(1):70–86, July 1990.

[17] A.D. Gross and A. Rosenfeld. Multiresolution object detection and delineation. *Computer Vision, Graphics, and Image Processing*, 39(1):102–115, 1987.

[18] E.S.E. Hafez. *Human Reproduction: Conception and Contraception*. Harper and Row, 1980.

[19] G.T. Herman and H.K. Liu. Dynamic boundary surface detection. *Computer Graphics, and Image Processing*, 7:130–138, 1978.

[20] E.C. Hildreth. The detection of intensity changes by computer and biological vision systems. *Computer Vision, Graphics, and Image Processing*, 22:1–27, 1983.

[21] T.H. Hong and A. Rosenfeld. Compact region extraction using weighted pixel linking in a pyramid. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 6(2):222–229, March 1984.

[22] J.Y. Hsio and A.A. Sawchuk. Unsupervised textured image segmentation using feature smoothing and probabilistic relaxation techniques. *Computer Vision, Graphics, and Image Processing*, 48(1):1–21, October 1989.

[23] A. Huertas and G. Medioni. Detection of intensity changes with subpixel accuracy using laplacian-guassian masks. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 8(5):651–664, September 1986.

[24] D.L. Hykes, W.R. Hedrick, and D.E. Starchman. *Ultrasound Physics and Instrumentation*, pages 87–129. Churchill-Livingstone, 1985.

[25] M.E. Jernigan and F. D'Astous. Entropy-based texture analysis in the spatial frequency domain. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 6(2):237–243, March 1984.

[26] R.P. Johnson. Contrast based edge detection. *Pattern Recognition*, 23(3/4):311–318, 1990.

[27] M.K. Leung and Y.H. Yang. Dynamic strip algorithm in curve fitting. *Computer Vision, Graphics, and Image Processing*, 51(3):146–165, 1990.

[28] H.K. Liu. Two- and three-dimensional boundary detection. *Computer Graphics, and Image Processing*, 6:123–134, 1977.

[29] S. Liu and M.E. Jernigan. Texture analysis and discrimination in additive noise. *Computer Vision, Graphics, and Image Processing*, 49(1):52–67, January 1990.

[30] P. Meer. Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, 45:269–294, 1989.

[31] D.R. Ney, E.K. Fishman, D. Magid, and R.A. Drebin. Volumetric rendering of computed tomography data: Principles and techniques. *IEEE Transactions On Computer Graphics and Applications*, 17:24–32, March 1990.

[32] J. Ophir and N.F. Macklad. Digital scan converters in diagnostic ultrasound imaging. *Proc IEEE 1979*, 67:654–664, 1979.

[33] T. Peli. A study of edge detection algorithms. *Computer Graphics, and Image Processing*, 20:1–21, 1982.

[34] M. Pietikainen and A. Rosenfeld. Image segmentation by texture using pyramid node linking. *IEEE Transactions On Systems, Man, and Cybernetics*, 11(12):822–825, December 1981.

[35] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*, pages 343–352. Cambridge University Press, 1988.

[36] I.L. Pykett, J.H. Newhouse, F.S. Buonanno, T.J. Brady, M.R. Goldman, J.P. Kistler, and G.M. Pohost. Principles of nuclear magnetic resonance imaging. *Radiology*, 143:157–168, April 1982.

[37] H.M. Raafat. A texture information-directed region growing algorithm for image segmentation and region classification. *Computer Vision, Graphics, and Image Processing*, 43(1):1–21, July 1988.

[38] A.R. Rao. *A Taxonomy for Texture Description and Identification*. Springer-Verlag, 1990.

[39] T.R. Reed and H. Wechsler. Segmentation of textured images and gestalt organization using spatial/spatial-frequency representations. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 12(1):1–12, January 1990.

[40] A.A. Rodriguez and O.R. Mitchell. Image segmentation using background estimation. *Intelligent Robots and Computer Vision*, 1002:170–181, 1988.

[41] A. Rosenfeld and A.C. Kak. *Digital Picture Processing*, chapter 8. Academic Press, Inc., 1976.

[42] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection.* John Wiley and Sons, 1987.

[43] H. Samet. Neighbor finding techniques for images represented by quadtrees. *Computer Graphics, and Image Processing*, 18:37–57, 1982.

[44] H. Samet and R.E. Webber. Hierarchical data structures and algorithms for computer graphics (part 1). *IEEE Transactions On Computer Graphics and Applications*, pages 48–68, May 1988.

[45] H. Samet and R.E. Webber. Hierarchical data structures and algorithms for computer graphics (part 2). *IEEE Transactions On Computer Graphics and Applications*, pages 59–75, July 1988.

[46] H.C. Shen and A.K.C. Wong. Generalized texture representation and metric. *Computer Vision, Graphics, and Image Processing*, 23(1):187–206, 1983.

[47] J.C. Taenzer and J.F. Havlice. Medical ultrasonic imaging: An overview of principles and instrumentation. *Proc IEEE 1979*, 67:620–641, 1979.

[48] H.L. Tan, S.B. Gelfand, and E.J. Delp. A comparative cost function approach to edge detection. *IEEE Transactions On Systems, Man, and Cybernetics*, 19(6):1337–1349, November 1989.

[49] U. Tiede, K.H. Hoehne, M. Bomans, A. Pommert, M. Riemer, and G. Wiebecke. Investigation of medical 3d-rendering algorithms. *IEEE Transactions On Computer Graphics and Applications*, 17:41–53, March 1990.

[50] J.K. Udupa, S.N. Srihari, and G.T. Herman. Boundary detection in multidimensions. *IEEE Transactions On Pattern Recognition and Machine Intelligence*, 4(1):41–50, January 1982.

[51] M. Unser and M. Eden. Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering. *IEEE Transactions On Systems, Man, and Cybernetics*, 20(4):804–815, July 1990.

[52] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, 1988.

[53] M.W. Vannier, J.L. Marsh, and J.O. Warren. Three dimensional computer graphics for craniofacial surgical planning and evaluation. *Computer Graphics*, 17(3):263–273, July 1983.

[54] Y. Xiaohan and Y. Baozong. A new texture feature: Direction measure. *SPIE*, 1395:291–298, 1990.

[55] Y.H. Yang and M.D. Levine. The background primal sketch: An approach for tracking moving objects. *Machine Vision and Applications*, (in press).