1 INTRODUCTION

Medical imaging is an integral part of diagnostics. Whether a bone is being evaluated for a possible fracture or the size of a tumor is being monitored, some form of medical imaging is involved. There are a number of imaging modalities which may be used, depending on what is being evaluated. These modalities include radiography, Computed Tomography, fluoroscopy, Magnetic Resonance Imaging, Positron Emission Tomography and ultrasound. Each modality measures some physical phenomenon and creates an image based on these measurements.

Some modalities, such as radiography, fluoroscopy and Computed Tomography (CT) involve the use of radiation (x-rays), which may concern many patients. Magnetic Resonance Imaging does not involve the use of x-rays, but like CT, it is an expensive procedure. CT and Magnetic Resonance Imaging (MRI) are two of the most commonly used imaging modalities and provide very high quality data. Another commonly used modality is ultrasound. By comparison to CT or MRI, ultrasound does not require exposing patients to radiation, it is readily available, inexpensive and at worst, only minimally invasive.

Ultrasound has gained widespread use in the evaluation of fetal development [Bartrum-Crow '77] and in the evaluation and treatment of atherosclerosis [Lengyel '95]. Cardiac ultrasound has been used in the veterinary medical field, for some time, for the evaluation of cardiac performance and the detection of breed related anomalies. In the case of fetal ultrasound, the examiners evaluate the length of the body and head, the movement of limbs, and, with the introduction of a contrast material, the possible presence of a cleft palate, among other things. The exam is generally non-invasive, takes little time, and does not expose the patient or the fetus to radiation. Blood vessels could be evaluated for the presence of and degree of plaque accumulation in the evaluation of atherosclerosis. In addition, treatment planning may also benefit.

Unfortunately, ultrasound does not provide the quality of data which CT or MRI provide. Ultrasound images tend to be very noisy, have low dynamic range, high variations in neighboring intensity values in "homogeneous" regions and possess poor contrast [Sakas 95]. This is largely related to the very nature of ultrasound. A more detailed explanation of ultrasound is given in Chapter 2.2 of this paper. Suffice it to say, for now, that the quality of ultrasound images is generally poor. A large amount of real data is missing, while many artifacts are present. It is this poor quality which creates problems while attempting to segment ultrasound images.

Since ultrasound images are so poor, some structures within the image are often difficult to evaluate. Consequently, some means of enhancing the image and segmenting structures present in the image could prove very beneficial to diagnosis and treatment. Segmentation of structures could aid the diagnostician in evaluating fetal development. General growth and development could be monitored as well as specific organ development.

Ultrasound images are initially captured onto video tape during the exam. In order to process these images, a series of images on the tape must be digitized. Since the data set comes from a series of images, it is very important that any package used in processing and segmenting images be able to segment a series of images. The use of this type of package in a clinical setting necessitates the rapid processing and segmentation of those images. These ultrasound images may vary from frame to frame in quality. This may effect the performance of segmentation procedures from frame to frame. This problem indicates the need for an interactive package which would allow the user to alter certain aspects of the procedures (parameters). Thus, an image which is poorly segmented using a given set of parameters may have those parameters altered during segmentation and therefore improve the segmentation of that image immediately. Determining which segmentation is "better", which is "worse" and when a region is optimally segmented depends on a knowledge base which the user must supply. This further indicates the need for an interactive system.

While the previously mentioned variation of quality from frame to frame may effect the performance of segmentation procedures, there is coherence present from frame to frame. This coherence may well aid in speeding up the segmentation of a sequence of images. The coherence from frame to frame may provide short sequences in which parameters from the previous frame perform well on the current frame. In cases where that does not occur, the parameters from the previous frame can act as a starting point in modifying the parameters for the current frame, therefore speeding up the process.

Many types of preprocessing and segmentation methods have been recorded in the literature, and will be further discussed elsewhere in this thesis. The performance of region growing by pixel aggregation and a variation of split and merge will be evaluated here. Region growing is a local algorithm which is effected by the immediate neighbors of a pixel or a region. It will continue to add pixels to a region until it encounters a pixel outside the tolerance range. Split and merge on the other hand, is a more global algorithm, evaluating a larger region of the image before determining if a region is homogeneous.

Variations of small numbers of pixels in a larger region have little effect on the final evaluation of the overall region. It is these differences which suggest that combining these two procedures may provide a method of segmentation which is superior to either algorithm alone. The performance of each of these algorithms, the performance of a combination of the two algorithms and the development of an interactive package which rapidly segments a series of ultrasound images will all be covered in this thesis.

Experimentation shows that the use of frame to frame coherence decreases the time required to segment a series of images by decreasing the number of images which must be altered when segmenting a sequence. In addition, it shows the use of split and merge and region growing procedures, in combination, offer an improved method of segmenting images. This combination of procedures also decreases the number of images altered when segmenting some sequences of images.

2 BACKGROUND

2.1 Introduction

Ultrasound offers a true challenge to image processing and segmentation. Images tend to be of poor quality, fuzzy and noisy. They often have a great deal of speckle and lots of artifacts [Sakas - Schreyer 95][Bartrum-Crow '77] and [Kremkau '89]. These artifacts are generally created by reverberation of sound waves, interference and noise. Further problems associated with ultrasound images include low dynamic range and high variations in neighboring intensity values in "homogeneous" regions. Often, ultrasound images either do not include the full range (0 - 255) of grayscale intensity values, or an overwhelming majority of pixels fall within a certain range of gray level values, 0 - 125, for example. This is poor dynamic range. In addition, a region which, to the human eye, appears "uniformly dark" or "uniformly light" may well consist of a variety of gray level values. These differences in intensity values are too subtle for the human eye to detect, but processing and segmentation algorithms, which examine each intensity value, can be greatly effected. Sakas refers to these and a number of other problems in [Sakas 95]. These include a variation in gray level values within boundaries brought about by the curvature of a surface and its orientation to the transducer which produces the sound wave. Also mentioned are the problems of "fuzzy" boundaries which are several pixels in width and the shadowing of one structure by another. The latter may occur if the hand of the fetus is physically in front of the face. Because of these problems, standard image processing and segmentation techniques, which perform well on Magnetic Resonance Imaging and Computed Tomography images, do not perform as well on ultrasound data. For example, the low dynamic range often seen in ultrasound images prevents the effective use of thresholding to segment regions of interest from noise or speckle. Other procedures that have been successful with CT and MRI data, but which have performed poorly with ultrasound include contour connecting, marching cubes, iso-surfacing, and others. Consequently, alternative methods, or modifications of standard methods are often utilized.

Examples of an MRI image (brain) and an ultrasound image (heart) may be seen in Figure 2.1. One can easily see the vast difference in the quality of data. In the MRI image there are distinct differences in intensity values between most neighboring tissues. The ultrasound however, has lost much of the detail required to readily separate one tissue from another on visual inspection.

Much of the work done on ultrasound data is on three dimensional data, for the purpose of volume rendering. This still requires performing operations (pre-processing and segmentation), on two dimensional "slices" of data. Given the previously mentioned list of techniques, other procedures have been attempted by a variety of researchers.

To prepare the image for segmentation, filtering is performed to improve the image. Most often used are the median or Gaussian filters, usually 3 x 3 or 5 x 5 for two dimensional data, and 3 x 3 x 3 or 5 x 5 x 5 for three dimensional data. In addition to these, [Sakas - Schreyer 95] uses a speckle filter to help remove speckle.

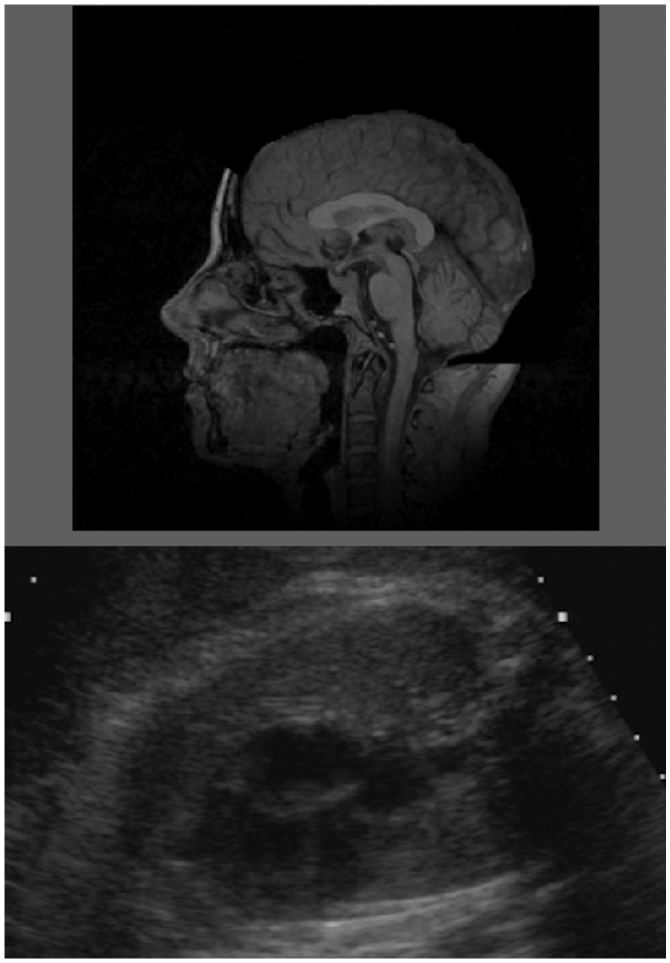


Figure 2.1

An example of MRI data (top) and ultrasound data bottom.

Having prepared the image, the next step is segmentation. Again, a variety of alternative procedures, or procedures modified from standard techniques have been used. [Sakas 95] included the use of multi resolution feature identification, mathematical morphology (erosion, dilation and opening), Binarization - Lowpass - Thresholding -

Propagation filtering and a combination lowpass filter and edge detection. [Meinzer '92] covers, in some detail, thresholding, texture analysis, region growing and edge detection, although this discussion covers their use with Magnetic Resonance Imaging and Computed Tomography data. In [Lengyel '95], energy minimizing splines are used with intravascular ultrasound images to find the center line of an artery, for the purpose of constructing a three - dimensional representation of that artery. The minimizing spline seeks that region of the image which is most opaque. This is the center of the artery. Two offsets are used for edge detection to find the walls of the artery.

2.2 Ultrasound

2.2.1 Introduction

Images are created of the body everyday for the purpose of evaluating the function, structure, or both, of all or part of the body. There are a number of imaging modalities by which these images are created, each measuring a certain phenomenon. These modalities include radiography, fluoroscopy, Computed Tomography, Magnetic Resonance

Imaging and ultrasound. Radiography measures the interaction of x-rays and a phosphorescent cassette with the emulsion covering a piece of radiographic film. [Thrall '86]. Its general uses include evaluating bones for the presence of fractures, or soft tissues for the presence of masses. Fluoroscopy is more like a continuous "x-ray" in which the x-rays are directed toward a phosphorescent screen, instead of a piece of film. It is used for procedures such as angiograms in which the physician can evaluate the flow of blood through vessels and search for blockages. Computed Tomography consists of x-rays projected through the body, from a number of sources. The degree to which the x-rays are attenuated as they pass through the body creates a matrix representing the cross - section or a "slice" of the body. It is also often used in the detection or monitoring of soft tissue masses. Magnetic Resonance Imaging (MRI) uses a strong magnetic field to align the axes of hydrogen nuclei. Since the body is largely composed of water, hydrogen nuclei are abundant. MRI measures the amount of time it takes these nuclei to relax and return to their natural orientation. This also creates a matrix representing a "slice" of the body. Not only can this provide the information which CT provides, it can also provide information on the physiology of the soft tissue, i.e. its activity. Ultrasound images are created by measuring the strength of echoes created when a sound wave, directed into the body encounters a medium of different density. It is also used in evaluating soft tissue. This will be discussed in greater detail in the next section.

2.2.2 Physics of Ultrasound

Humans hear sound at frequencies between 20 and 20000 Hz or cycles per second. Ultrasound is defined to be any sound greater than 20000 Hz. For the purpose of clinical ultrasound imaging, the frequencies used usually fall in the range of 1MHz to 10MHz (1 million Hertz to 10 million Hertz), although even higher frequencies may be used.

Ultrasound is a sound wave, which means it is a mechanical disturbance which propagates through a medium. The medium through which it propagates determines the speed at which it travels. In air the speed of sound is 331 meters per second, while in soft tissue it is 1540 meters per second. This value is actually an average since the speed varies from tissue to tissue. The speed of sound through fat is approximately 1450 meters/second while the speed of sound through bone is approximately 3183 meters/second. This is a good example of the fact that the speed of sound varies with the density and compressibility of a material. The density of bone is much greater than that of fat, and consequently, the speed of sound through bone is greater than that of fat.

It is this difference in density or compressibility which causes echoes to be created. When a sound wave encounters an interface between media of different densities, part of that sound wave is reflected, creating an echo. The greater the difference in densities of the media, the stronger the echo and the greater the amount of signal reflected. This provides a strong echo, but leaves little of the original sound wave to continue through tissue to interact with interfaces beyond that point. An interface of two media whose densities differ slightly creates a weak echo. This creates a weak signal, but leaves more of the signal to continue through the remaining tissue. Regardless of the strength of the echo, or the original sound wave, sound waves deteriorate as they travel through a medium. In soft tissue the rate of deterioration is taken to be 1 dB per cm per MHz. Therefore, due to the natural deterioration of sound waves and the loss of strength due to reflection, the data obtained from ultrasound is often weak or "fuzzy".

2.2.3 Transducer

The sound waves introduced into the body and the echoes created within the body are created and detected by the transducer. The transducer consists of a single piezoelectric crystal, or an array of crystals. These crystals have the ability to change electrical signals into mechanical vibrations and vice versa. The crystals are coated with a conducting veneer such as silver. Wires are attached to this veneer for the purpose of applying and detecting voltages. In figure 2.2 is a diagram of an ultrasound transducer figure [reproduced from Wolbarst '93 - figure 46-1]

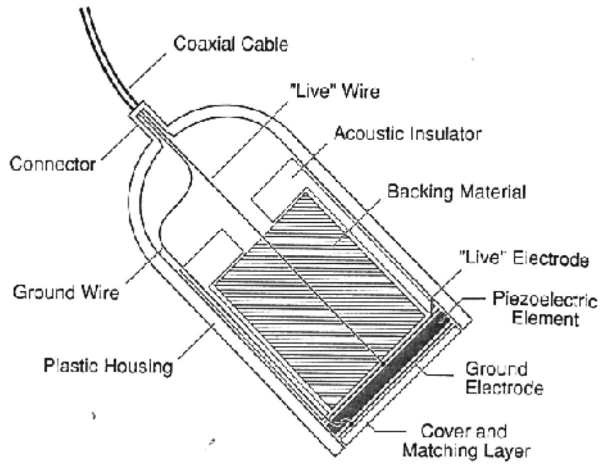


Figure 2.2

Components of a transducer.

Therefore, the basic progression of events is as follows: a voltage is applied to the transducers crystal, causing the crystal to deform and create a mechanical disturbance (sound wave) of a certain frequency. The blast of sound lasts approximately one microsecond. The sound wave enters the body and travels through the soft tissue media until an interface of varying densities is encountered. An echo is created which then travels back to the transducer. Following the microsecond blast of sound, the transducer enters a listening phase which lasts for approximately a millisecond. During this listening phase the crystal detects the echo (mechanical vibration) and converts this to an electrical impulse. This impulse travels along the wires attached to the crystal and is recorded or displayed. Figure 2.3 shows a schematic of this process.

Once the electrical impulse has been detected, there are a number of ways in which this information may be displayed. The oldest and simplest form is called A-mode. In A-mode imaging, the echo (voltage) is displayed as an amplitude spike on the screen of an oscilloscope. As
the sound wave is released into the body, a light point moves across the screen. Upon detecting an echo, a spike is generated on the oscilloscope. The horizontal axis represents time. Therefore, the depth of the interface which created the echo can be found using the equation :

$$\text{depth} = \frac{1}{2} * c * t$$

where t is the time (i.e. the point along the horizontal axis at which the spike occurred) and c is the speed of sound (1540 m/s).

Another means of imaging the echo information is called M-mode and allows the display of motion of tissue. The display

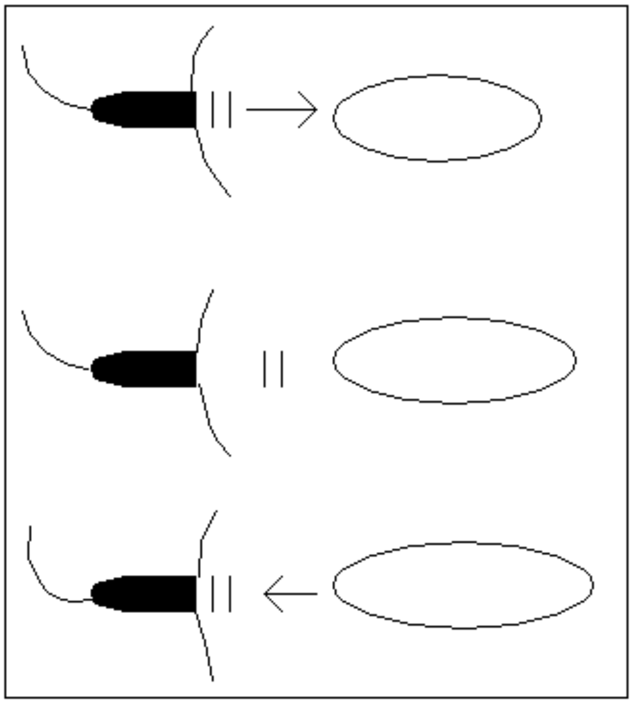


Figure 2.3

Path of sound wave into body and returning echo.

is oriented 90 degrees from that of A-mode, and a dot appears on the screen when a spike occurs on the oscilloscope. The brightness of the dot corresponds to the strength of the echo. A stronger echo creates a brighter dot while a weaker echo produces a darker dot. This method has been replaced, for the most part by real time B- mode. Real time B-mode imaging is the most commonly used form of ultrasound imaging. It produces an image of a two-dimensional "slice" of anatomy. The sound beams sweep back and forth a number of times per second. The system tracks each beam. As an echo is detected, for any given beam, a dot on a corresponding display line is produced. The timing between the detection of echoes for a given beam allows the system to calculate the distance between the interfaces creating the corresponding echoes. This is translated onto the display line as dots separated by distance *d.* As always, the brightness of these dots is proportional to the strength of the echoes. All of the dots therefore create an image of a slice of the body. Figure 2.5 shows an example of a B-mode display, while figure 2.4 gives a schematic layout of a B-mode system. Both of these images were also taken from [Wolbarst '93 - figures 47-8 and 47-6 respectively.

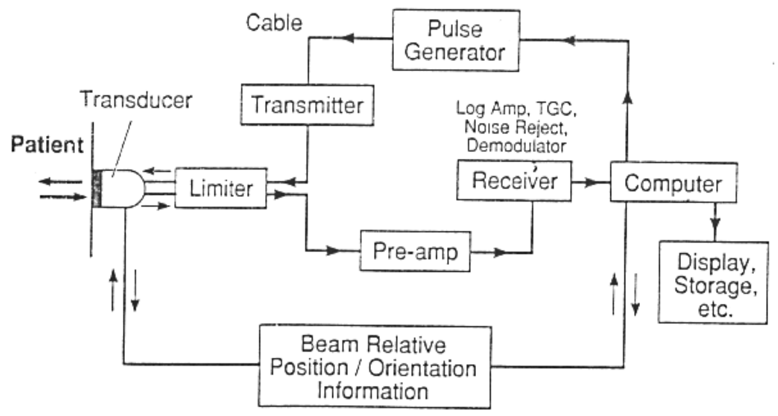


Figure 2.4

Schematic of a basic B-mode system

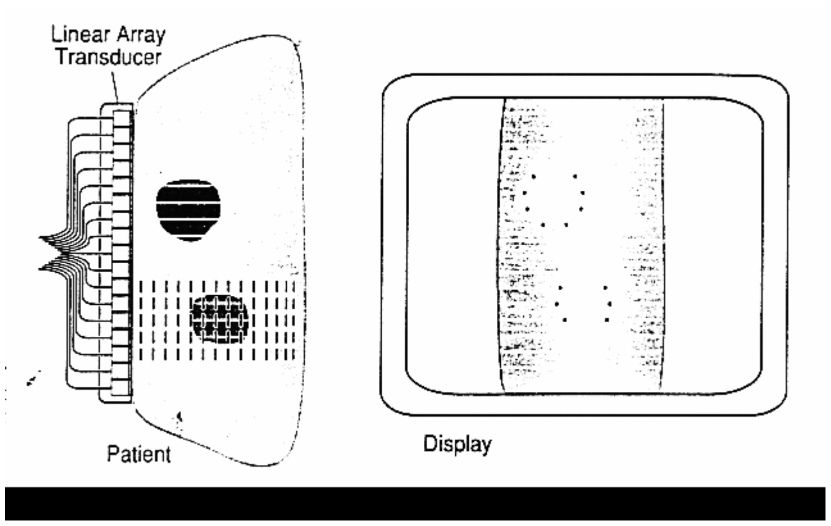


Figure 2.5

Echo detection and corresponding B-mode display.

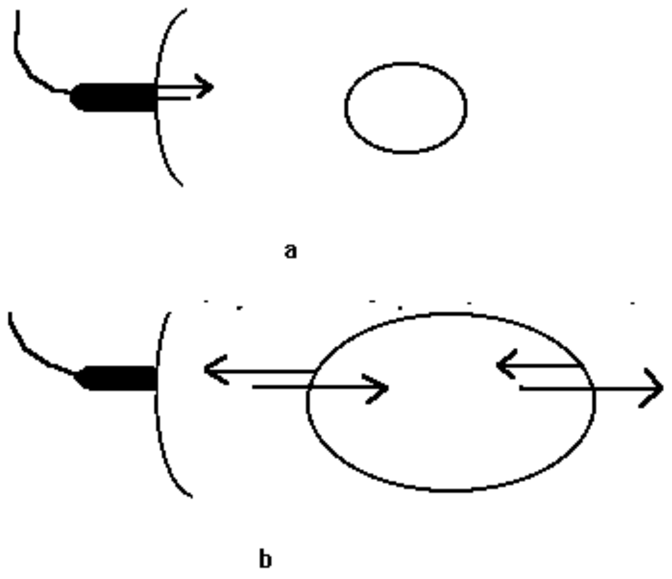
2.2.4 Artifacts

Unfortunately, not all the dots present in an image represent true interfaces. Those dots which are not true interfaces are known as artifacts. The cause of these artifacts generally fall into three categories : electronic noise or interference, signal processing or reverberations. Interference or "electronic noise" may come from inside or outside the device. Inside an electronic device electrons move in a random pattern until a signal is applied. At that time, electrons move in a specific pattern. Despite this, some electrons still move about randomly. This can create gray dots in the image which do not represent a true structure. Likewise, high frequency signals created outside the ultrasound device, i.e. by other devices, produce similar results.

Once a signal is received by the unit, the way in which it is processed may create artifacts. In certain types of scanners a "loud - echo artifact" is common. A very strong echo signal hides other echo signals received after it, and creates omission artifacts. This may result when the gain is set too high. Grayscale scanners however, do not have this problem, but may encounter artifacts created by other signal processing techniques. Many modifications are made to the signal in an attempt to improve the resulting image. Some scanners smooth the image while others highlight edges. Those that smooth the image may create loss of detail artifacts while edge enhancing scanners may create inclusion artifacts by promoting noise to echoes. Many of these artifacts are not detrimental from a diagnostic standpoint, but remain artifacts.

Reverberation artifacts generally result from 1) the reflection of a returning echo by the transducer or 2) the reflection of a sound wave off internal structures. An example of these may be seen in figure 2.6 a and b respectively. In figure 2.6 a it can be seen that a sound wave emitted by the transducer is reflected by the internal surface. This echo returns to the transducer where it is detected. The echo, encountering another interface, in the form of the transducer, is reflected into the body. Therefore they travel through the body again, encountering the same interfaces already recorded. Since the depth of the interface is determined by the timing of the echo detection, the second encounter with an interface ultimately creates a dot on the display twice as deep as the true interface. This reverberation continues until the sound wave deteriorates to the point below the detection capabilities of the transducer.

Figure 2.6 b shows that as a sound wave encounters an interface, part of that sound wave is reflected as an echo, to be detected by the transducer. Another part of that original sound wave continues beyond that interface, to encounter deeper interfaces. Here the sound wave is again reflected. On the path back to the transducer, it encounters the first interface, where part of the sound wave continues on to the transducer and part is reflected



[redacted]

Figure 2.6

Reverberation of sound waves.

back into the body to again encounter interfaces already encountered. Again, this continues until the sound wave deteriorates to a point below the detection capabilities of the device. Reverberation artifacts may also stem from reflection of sound waves from interfaces at less than 180 degrees. These waves are not returned to the transducer, but travel into the body in other directions. They may eventually be detected by the transducer after having been reflected several more times. These include scatter and multipath artifacts. When a sound wave hits the surface of a structure, reflections are created in a number of directions, not just 180 degrees. This is scatter. These scatter echoes may eventually be detected by the transducer. They may reinforce each other, or cancel each other, creating a dot pattern on the image. This pattern is speckle. An example of scatter may be seen in figure 2.7. Multipath artifacts occur when sound waves encounter an interface. The sound wave is reflected away from the transducer and encounters another surface. The echo created by this surface eventually reaches the transducer. Since the transducer interprets echoes as occurring in the direct path, the dot is placed farther along the axis of the transducer than the surface really occurs, creating an artifact. An example of this may be seen in figure 2.8.

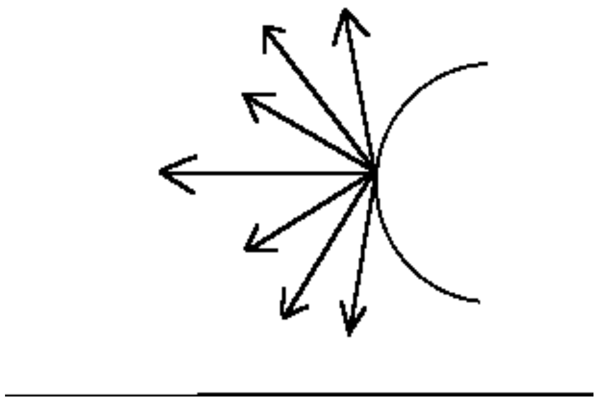


Figure 2.7

Scatter

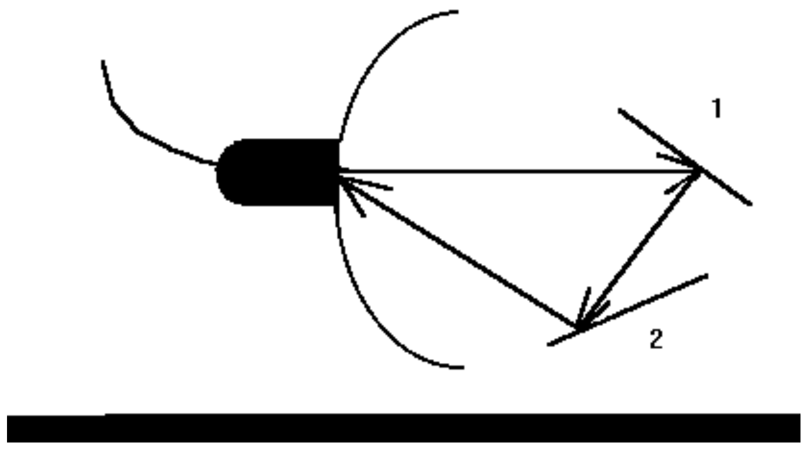


Figure 2.8

Multipath Reverberation

All of these artifacts help create fuzzy, noisy images. This, coupled with less than optimal data created by true interfaces, can make the evaluation of these images difficult. Further information concerning the physics of ultrasound, the transducer and artifacts may be found in [Wolbarst '93],[Bartrum-Crow '77] and [Kremkau '89].

2.3 ENHANCEMENT

There are a large number of procedures which fall into the category enhancement. Within this category are pre - processing procedures and segmentation procedures. Pre - processing procedures are generally used to improve the quality of an image prior to further processing, such as segmentation. Preprocessing procedures include Gaussian, median and lowpass filtering, gray - level slicing, contrast stretching, histogram equalization and image averaging. Segmentation procedures divide an image into its constituent parts. These include, but are not limited to, region growing by pixel aggregation, split and merge, thresholding and edge detection.

2.3.1 Pre - processing

Preprocessing methods may be based on the intensity value of an individual pixel (point processing), or the intensity values of a neighborhood surrounding a pixel. Point processing techniques include contrast stretching, gray level slicing and histogram equalization. Neighborhood or local techniques include image averaging and spatial filtering. Spatial filters may be sharpening (highpass) filters, which highlight detail, or smoothing filters, which blur and reduce noise. Smoothing filtering includes lowpass and median filtering.

Lowpass filtering blurs edges by highlighting or "allowing to pass through" the darker gray level values. It is the darker values which give smoothness to an image and decrease detail. A lowpass filter also works by replacing the pixel centered under the filter by a weighted sum of those pixels within the filter. Box filters and Gaussian filters are examples of lowpass filters. Gaussian filtering is used for smoothing or blurring an image. It works by replacing the value of a pixel with the weighted sum of the intensity values of those pixels within the filter. The weights used depend on the standard deviation. The box filter calculates an equally weighted average for the neighborhood.

Median filtering works to reduce noise, especially noise of a spike-like nature. While it reduces noise, it does so while preserving edges. The filter works by replacing the intensity value of the pixel centered under the filter with the median value of those pixels which fall under the filter. The intensity values of the pixels falling under the filter are read into an array and sorted. The median value replaces the value of the pixel at the center of the filter.

Image averaging is another method by which noise is removed. The intensity values of corresponding pixels from several images are averaged to create one image. The underlying principle is that noise present in one image may not be present at the same place in another image. Consequently, by averaging several images , the noise will be decreased. Examples of Gaussian and Box filters may be seen in figure 2.9. The Gaussian filter in figure 2.9 was created using the following equation, from [Haralick '92]. A standard deviation of 1.0 was used.

$$w(r, c) = ke^{-\frac{1}{2}\frac{(r^2 + c^2)}{\sigma^2}}$$

where

$$(k) = \frac{1}{\sum e^{-\frac{1}{2}\frac{(r^2 + c^2)}{\sigma^2}}}$$

for all (r,c) contained in neighborhood W. Where r indicates the row number of the filter (from -filter width/2 to +filter width/2), c indicates the column number (also from -filter width/2 to +filter width/2) and s is the standard deviation.[Haralick '92]

1	1	1
1	1	1
1	1	1

BOX

.0556	.0917	.0556
.0917	.4109	.0917
.0556	.0917	.0556

GAUSSIAN

Figure 2.9

Smoothing filters

Contrast stretching, gray level slicing, histogram equalization and highpass filtering all work to brighten all or part of an image. Contrast stretching enhances the contrast between pixels by increasing the differences in the original intensity values. It works by multiplying the intensity value for each pixel by a stretch factor. For example, if original values 10 and 30 were located in adjacent pixels, following a stretch of 3.0, the values would be 30 and 90 respectively. Therefore instead of a gray level difference of twenty, which is hard for the eye to discern, the difference is now sixty. Gray level slicing allows pixels with intensity values falling within a range to be set to some set value. This allows the highlighting of certain structures consisting of pixels whose intensity values fall within that range. For example, all pixels with intensity values between 50 and 100 may be set to a value of 255. Histogram equalization lightens dark images by redistributing the intensity values of the pixels more evenly through the range of gray level values. This is accomplished by determining the histogram of each image. The percentages determined by the histogram are added and, based on the number of gray levels across which the mapping occurs, the old intensity values are mapped to new values. These new values are then written back to the image. Highpass filtering allows the higher gray level values or frequencies to pass through the filter. Since the higher gray level values provide detail to an image, all these procedures work to brighten the image and add detail. Further information concerning all these procedures may be found in [Gonzalez '92].

2.3.2 Segmentation

Segmentation is defined as the subdivision of an image into its constituent parts or objects. This may be as simple as separating a dark object from a light background, or as complex as separating many regions within a noisy, fuzzy ultrasound image. The segmentation of grayscale images is based on the discontinuity of neighboring pixel intensity values, or the similarity of neighboring intensity values. Those techniques which look for similarity include region growing by pixel aggregation, split and merge and thresholding. Edge detection is a technique which looks for the discontinuity of neighboring intensity values.

Thresholding

Thresholding segments an image based on one or more threshold values. These values are usually obtained by evaluating the histogram of the image in question. In simple thresholding, one threshold value is chosen. Those pixels having intensity values less than the threshold are set to one value, while the remaining pixels are set to another value. If two threshold values are used, then pixels with intensity values less than the first threshold are set to one value. Pixels with intensity values greater than the higher threshold are set to a second value, and those with intensity values between the two thresholds are set to a third value. This allows the simple separation of objects from background, or objects of varying intensity values from each other and the background.

Region Growing

Region growing segments by taking a seed pixel or a group of seed pixels and determining the intensity value ,or an average in the case of multiple seed pixels. The intensity value of the four neighbors of the seed pixel are then compared to the intensity value of the seed pixel. If the difference in intensity values falls within an acceptable range, the neighbor is added to the region. In turn, as pixels are added to the region, their neighbors are evaluated. This continues until no further pixels are added to the region. Since this procedure is based on evaluating the immediate neighbors of the seed pixel, it is considered a local or neighborhood procedure. It is therefore greatly affected by the intensity values of the four neighbors of each pixel within the region. Figure 2.10 gives the general algorithm for region growing, while figure 2.11 shows the progression of the growth of a region using this method. In 2.11a an X indicates the seed pixel, and the dots indicate the neighbors which will be evaluated for membership. 2.11b indicates those neighbors which were found to be in the region with an X. Those which were not found to be in the region are ignored. Again, the new neighbors to be evaluated are indicated with a dot. This continues through 2.11c and 2.11d

```
Get Seedpixel
                    Determine Intensity value of seed pixel
             While (pixels added to region)
             {
                  Get pixel
                   Evaluate Left Neighbor
                         If (Left neighbor in region)
                              Add pixel
                   Evaluate Right Neighbor
                         If (Right neighbor in region)
                              Add pixel
                   Evaluate Top Neighbor
                         If (Top neighbor in region)
                              Add pixel
                   Evaluate Bottom Neighbor
                         If (Bottom neighbor in region)
                              Add pixel
             }
```

Figure 2.10

Region growing algorithm

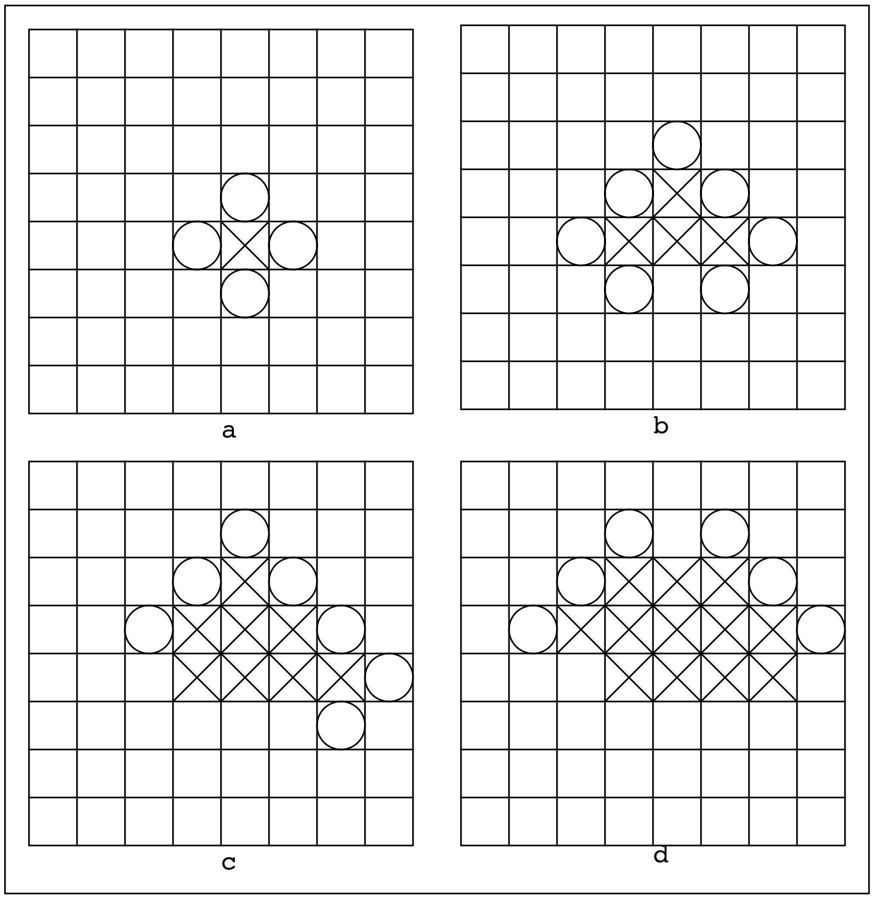


Figure 2.11

Graphical representation of Region growing algorithm

Split and Merge

Split and merge works by evaluating an image for homogeneity. If the image is not homogeneous, the image is subdivided into four parts. These parts are in turn evaluated. Any subregion found to be homogeneous has all pixels within the region set to the average intensity value for that region. If there are adjacent regions which are homogeneous, these regions may be merged. This process continues until no further splitting or merging can occur. Figure 2.12 shows the general algorithm for this procedure, while figure 2.13 shows the general algorithm for this procedure.gives a graphic display of the algorithm's progression through segmentation.

```
Do

     Evaluate region
     If (homogeneous)
     {
            label with region average
            merge with other homogeneous regions
     }
     else
     {
            Split into 4 subregions
     }
Until no more splitting or merging can be done
```

Figure 2.12

Split – Merge Algorithm

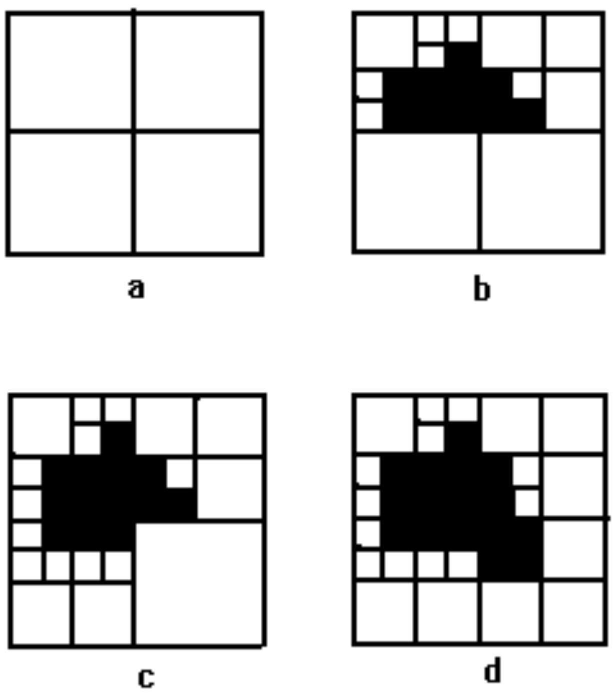


Figure 2.13

Graphical representation of Split/Merge

Edge Detection

Edge detection looks for discontinuity between neighboring pixels. This generally represents a change from one object to another. Any of a number of filters may be used to search for these discontinuities. Some actually look for lines in certain directions. Filters falling into this category include Sobel, Prewitt and Laplacian filters. Examples of such kernels may be seen in figure 2.14.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Prewitt

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Sobel

0	-1	0
-1	4	-1
0	-1	0

Laplacian

Figure 2.14

Edge detection filters.

Other Methods

Still other segmentation procedures are utilized by Bezdek in [Bezdek '93]. These include both supervised and unsupervised methods. The supervised methods include Bayesian classifiers, k - nearest neighbors and Feed

Forward neural networks. These require that the data set consists of labeled members which provide information about each data member. Using this information, these processes can segment the image into classes so as to minimize error in classification. The Bayesian classifier uses a training set to determine a maximum likelihood estimation function for each density. Using this information, each pixel can then be classified and the image segmented. The k - nearest neighbor procedure assigns a classification to unlabeled pixels by considering the k closest neighbors. Closest generally refers to some Euclidean distance. The classes of the neighbors are found and the unlabeled pixel is assigned to the same class to which the majority of its neighbors belong. The number of classes, the number of neighbors to consider, the distance and the method of vote counting may be altered. Feed forward neural networks take information stored on each pixel, as a feature vector, process these vectors through a number of hidden layers, and produce an output layer indicating to which class the pixel belongs. The number of hidden layers, nodes per layer, and any weights placed on node connections between layers may all be varied.

Unsupervised methods are automatic, not utilizing prior knowledge of the data. Bezdek utilizes an unsupervised version of the Bayesian classifier in which the only major difference is that the data is not labeled. Fuzzy and hard - c means algorithms segment an image based on c partitions or classes. These class definitions may be hard or fuzzy. Pixels are assigned colors based on assignment to a partition. Assignment to a partition is based on minimizing the sum of squared errors function. It is still required however, that a human operator assign these segments to specific tissue classes. These procedures have been used with varying degrees of success on MRI data.

3 INTERACTIVE SEGMENTATION

3.1 Background:

This project was developed in association with the Women's Institute at the Carolina's Medical Center in Charlotte, North Carolina. The initial idea was to provide a means of enhancing and segmenting images obtained from The Women's Institute clinical ultrasound exams. Since the captured ultrasound image quality is so poor, it was hoped that enhancement and segmentation of these images would aid in the clinical evaluation of the fetus. Specific problems of interest were the evaluation of the chambers of the heart for detection of masses and the possible identification of cleft palates, without injecting a substance to aid in its detection, which is presently required. In addition, the general enhancement and segmentation might also assist in the general evaluation of the development of the fetus.

Since ultrasound consists of a sequence of images taken over time, it was very important to be able to process a series of images. Since the ultimate goal was to use this package in a clinical situation, the package would have to process these images rapidly. Given that ultrasound images may vary from frame to frame, it was also important that the user be able to interact with the system, in order to guide the segmentation of individual frames. Therefore, what was needed was an interactive system which would allow a user to process and segment a series of images in a short period of time.

3.2 User Session :

The present version of the software presents the user with a graphical user interface to begin the session. A menu bar provides options for file manipulation (read image, write image, display, quit), image operations (smoothing, median filtering, contrast stretching, gray level slicing, image averaging) and segmentation (region growing, split / merge, split/merge region growing overlay).

A session begins by reading in a single image or a sequence of image files. Figure 3.1 shows the interface the user is given. A single image file is given an ID (which the user may set) which is used internally to identify the image. When a sequence of image files is read in, the user supplies an identification prefix. This prefix is suffixed with the number of the image files read. For example, if the files 0.rgb through 9.rgb are read in and the image prefix ID is given as "image", then the images will be identified internally as image0, image1, ... image9. This identification label is used as for the duration of the session. At this point, the ID "image0" can not be used more than once in a session, an alternative ID would have to be chosen. Likewise, in a sequence, a particular prefix could not be used more than once. If "image" is used as the identification prefix for images 0.rgb through 9.rgb, then another identification prefix would have to be used if another set of images 0 through 9 needed to be read in.

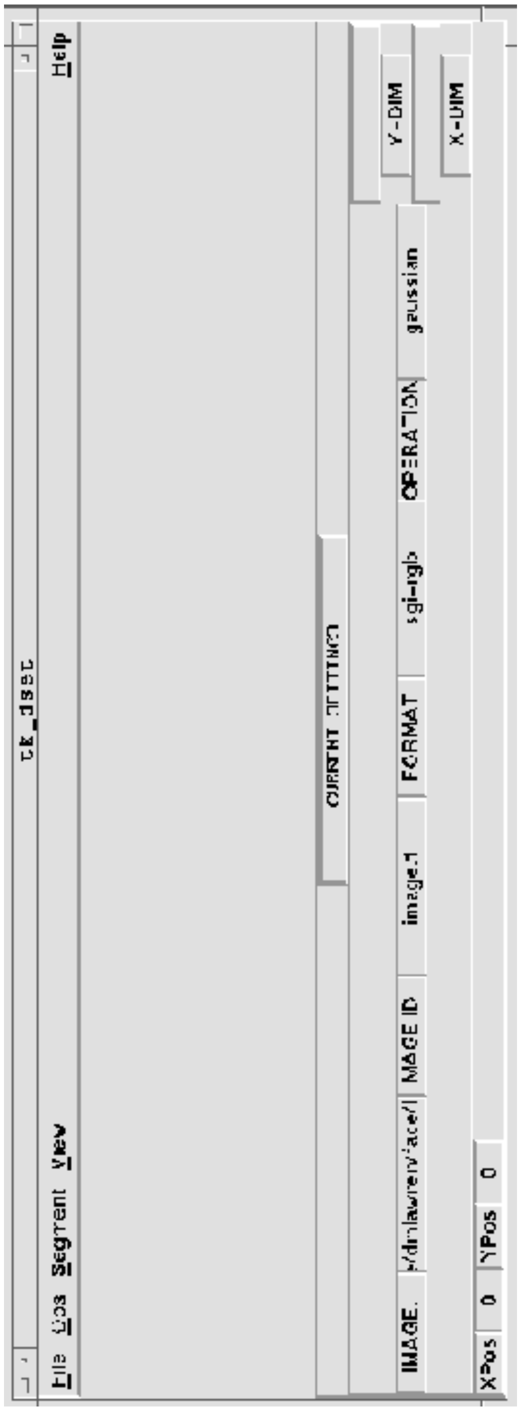


IMAGE: .../dmlawren/face/i
IMAGE ID image1
FORMAT sgi-rgb
OPERATION gaussian
XPos 0 YPos 0

Figure 3.1

tk_dset user interface

As an image is read in, it becomes part of a data structure referred to as a discrete set. The intensity value for each position within the image is stored in a one-dimensional array.In addition, information concerning the size of the image (x , y and z dimensions, size of storage for each intensity value, etc.) are also stored within the data structure. This data structure is accessed by a pointer which is stored in a hash table. The hash table uses the image identification label to access the correct image, thus the need for unique identification labels during a session.

Once the image(s) is/are read in, any of a number of procedures may be used to manipulate the images. To prepare the image for segmentation, usually some type of pre-processing procedure, available under the OPS menu, is selected. The appropriate parameters are entered by the user, in the dialog boxes. If a sequence of images are being pre-processed, the entire sequence will be processed at once. Upon finishing all images, the first image in the sequence will be displayed.

As an image is put through the pre-processing technique(s), the identification label is used to access the appropriate pointer in the hash table. This gives access to the intensity values for that particular image. The technique is carried out and the modified intensity values are written to the array in the discrete set. Neither the hash table nor the pointer is altered, only the discrete set.

Once pre-processed, the user selects a segmentation method from the SEG menu. Immediately, the user selects two opposite points of a bounding rectangle, outlining the general area of interest. Next, in the case of region growing, a seed pixel is selected. In the case of split and merge, the user clicks on a point representing the region of interest. A widget, seen in figure 3.2, then comes forward indicating parameters for the corresponding segmentation procedure. After setting the parameters, OK is selected to perform the segmentation. Immediately the software is prepared for the user to select another region for segmentation (i.e. select another bounding rectangle and seed pixel / region of interest.) A maximum of ten regions may be grown at this time.

Once all regions desired are grown, the user may select MODIFY in order to alter parameters on any region. Upon selecting MODIFY, the scales within the widget are automatically set to the region number one and its corresponding parameter values. The user selects which region to alter by setting the region id scale to the appropriate number. Parameters may then be changed. These new parameters take effect only when OK is selected. This process can continue until the user selects END, QUIT, or, in the case of a sequence, NEXT. END takes the software out of the current segmentation mode (region grow, split and merge or overlay) while QUIT ends the entire session and returns the user to the unix prompt. If the user is processing a sequence of images, the selection of NEXT advances the system to the next frame in the sequence and

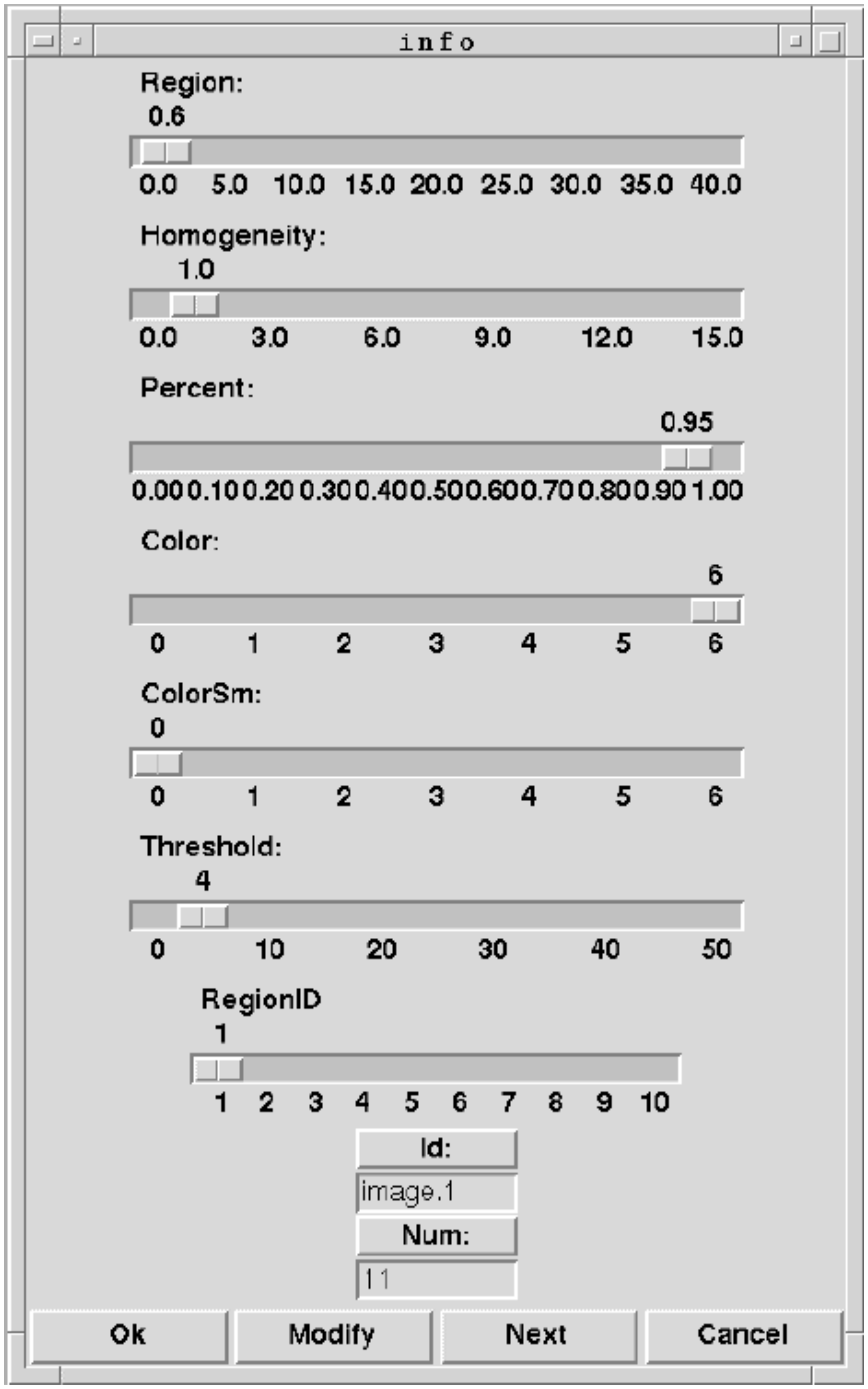


Figure 3.2

Tk widget for receiving parameters - overlay automatically uses the parameters for the regions grown in the preceding frame, stored in a data structure, to grow the regions in the new frame. These regions may be acceptable to the user. In this case, selecting CONFIRM accepts these regions and advances the system to the next frame in the sequence. If one or more regions are not acceptable to the user, selecting TEST will put the software into a state allowing the user to modify the parameters. As explained before, modification occurs by selecting the number of the region to be modified, altering the parameters and selecting OK . This can continue until the user selects END, QUIT or NEXT. This process continues on throughout the entire sequence of frames. When the user has finished with the frame(s), the frame(s) may be written to separate files for later viewing by selecting WRITE_IMAGE under the FILE menu and inputting the proper information.

3.3 Data Structures

Regions grown in a frame using this software are stored in an array of data structures called paramnodes. The structure of the paramnode and its associated structures, may be seen in figure 3.1. Information pertaining to each region is stored within the structure. This information storage drives the software. Information concerning the seed pixel used is stored in xval and yval, while the bounding rectangle used is stored in minx, miny,maxx, and maxy. This allows the software to use the same seed pixel and bounding rectangle used to grow a region in one frame on the next frame in the sequence. There is also space for the storage of parameters used by the various segmentation procedures. Region growing requires the storage of the tolerance range value, while split/merge requires percent, region and homogen. Overlay uses all these parameters. In addition, information concerning the respective color, both the index and an rgb triple are stored. All this works to grow regions in sequential frames based on parameters established in the previous frame. Also stored within this structure is an id number and two points lists. When a region is grown using the region growing procedure, the points identified as part of the region of interest are added to the ptslist_rg points list. Likewise in split/merge, the points identified are added to the ptslist_sm points list. In overlay, both lists are used since points are added by each procedure. These points are the key to "undoing" a region in modify mode, where the parameters are altered. In addition, the id number stored allows the user to select a specific region, when multiple regions are present, for modification. Using this id, the software can identify the proper data structure within the array and use the correct information to undo and regrow when modifying.

```
struct paramnode
{
  int xval, yval;       /* seed pixel coords */
  char name[30];    /* image handle */
  int num;      /* neighborhood size for ROI */
  int id;       /* region id */
  int threshold;    /* threshold for regiongrow */
  float percent;   /*percent of pixels accepted for homogeneity */
  float region;    /* # of std dev from ROI mean for acceptance */
  float homogen; /*#of std dev from region mean for homogeneity */
  int color_sm_index;
  int color_reg_index;
  RegColor color_reg, color_sm;
  struct properties prop;
  struct node *ptslist_rg;
  struct node *ptslist_sm;
}
struct {int r, g, b;} RegColor;

struct node
{
  int addr;
  int intensity;
  int x;
  int y;
  struct node *next;

};
struct properties
{
  int minx, maxx, miny, maxy;  /* bounding coordinates */
  int frame;
  int image;
  float ave;
  float std_dev;
  int pixels;
};
```

Figure 3.3
Data structure for paramnode and associated structures

3.4 System Design

Tcl, a scripting language, was used to create the package. The early Tcl version handled all image manipulation through the command line. Only after the image being manipulated was written to a file, could the effects of the image manipulation be viewed. In order to make the package more interactive and allow the user to view the changes to the image as they occur, the Tk toolkit was added.[Ousterhout '94] The use of the Tcl/Tk toolkit allows for the rapid development of the graphical user interface, with all frames, dialog boxes and other necessary widgets created and managed with minimal code. Once the basic framework of the package is established, it is relatively easy to modify, in terms of adding other commands or menu items. Another benefit to using Tcl/Tk was that it utilizes OpenGL graphics software. Both the Tcl/Tk software and the OpenGL software are readily available through public domain, and are easily ported to PCs. This is very important in ultimately providing this package to hospitals. Examples of the user interface may be seen in figure 3.1 and 3.2.

3.4.1 Image Enhancement

Given the poor quality of ultrasound images, a means of preprocessing these images would have to be incorporated. Research has indicated the widespread use of smoothing prior to segmentation, usually by Gaussian or median filtering. Therefore, these procedures would be incorporated in the software. To add versatility, other pre-processing techniques were added as well. The following operators were implemented:

1. Gaussian filtering
2. Median filtering
3. Contrast stretching
4. Gray level slicing
5. Image averaging
6. Histogram equalization

These procedures follow the standard algorithms outlined in Chapter 2.3 There are however, certain parameters which the user must enter, depending on the procedure. For median filtering, this software allows the user to select a median filter of size 3 x 3, 5 x 5, 7 x 7, 9 x 9, or 11 x 11. The size is entered via a dialog box. Gaussian filtering requires that both the size of the filter and the standard deviation are set by the user. Image Averaging requires the user to designate how many images are to be averaged (N). Once averaged, the resulting image is written into the discrete set for image N/2.

Contrast stretching offers a little more versatility than the standard algorithm in that the user may chose to stretch only values in a certain range. Of course, the option remains to stretch the entire image by selecting 0 as the lower bound and 255 as the upper bound. The desired stretch factor is also entered by the user. Figure 3.4 shows, on the top, an original dark image. On the bottom is the same image following contrast stretching using a stretch factor of 2.0. Similarly, gray level slicing has the user select a lower and upper bound, but instead of a stretch factor, the desired gray level value to which the values within the bounding range are set, must be given. Histogram equalization requires no input by the user at this time.

3.4.2 Image Segmentation

Segmentation techniques used in this software are based on the similarity of intensity values. These are region growing and split and merge. In addition, this software includes an overlay of the region growing procedure over the split and merge procedure, as a possible improved method of segmenting ultrasound images.

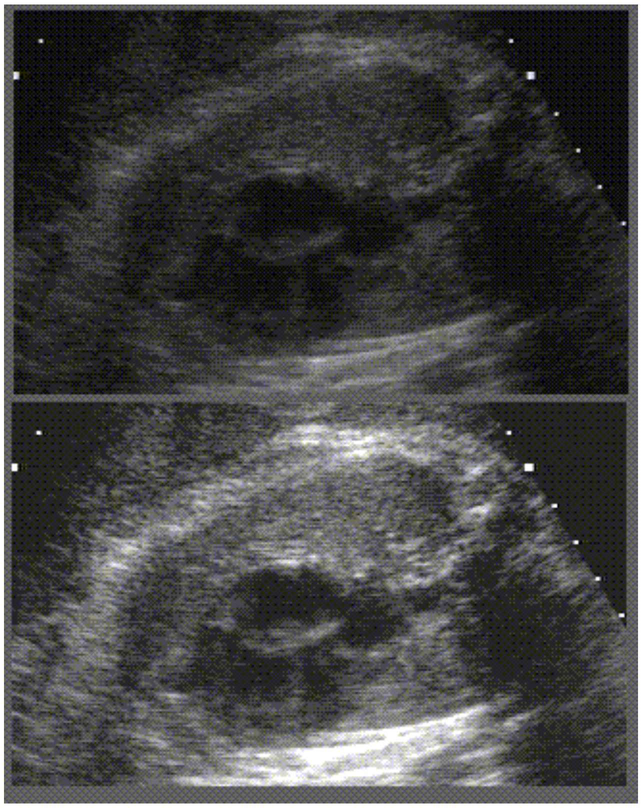


Figure 3.4

Top-original image Bottom-after contrast stretching by a factor of 2.0

Region growing begins with the user selecting two opposite points of a bounding rectangle for the section to be processed. This is followed by indicating a seed pixel.

Only one seed pixel is used in this procedure. The intensity value for this pixel is found and information about the pixel added to a queue. At this point, a loop is entered and the first item removed from the queue. The four - neighbors of this item are evaluated. If a neighbor's intensity value falls within a tolerance range, set by the user, of the seed pixel's intensity value, the neighbor is added to the queue. The loop repeats, taking an item from the queue as long as the queue is not empty. An empty queue signifies that no further pixels are added to the region and therefore serves as the stopping criteria. As a pixel is taken from the queue, it is added to a points list created for that specific region. These points are used for both "undoing" in Modify mode and for the determination of region statistics. Figure 3.5 shows the algorithm used in this software.

```
Get Seed Pixel
Find seed pixel
While (queue_not_empty)
{
    get pixel from queue
    if (left neighbor intensity - seed intensity <= Tolerance)
                put left neighbor on queue
    if (right neighbor intensity - seed intensity <= Tolerance)
                put right neighbor on queue
    if (top neighbor intensity - seed intensity <= Tolerance)
                put top neighbor on queue
    if (bottom neighbor intensity - seed intensity <= Tolerance)
                put bottom neighbor on queue
}
```

Figure 3.5

Region growing algorithm

Split and merge also begins with the user identifying two opposite points of a rectangle bounding the section of the image to be processed. Next, the user clicks on a position within the region of interest. This position is used as the center of a neighborhood, which is used to represent the region of interest. The size of the neighborhood is set by the user. Pixels within the neighborhood are used to determine the average intensity value of the region of interest and the standard deviation. These are the characteristics of the region of interest to which the homogeneous regions will be compared.

The next step consists of dividing the section outlined by the bounding rectangle into four subregions, creating a quadtree structure. Each of these four sub-regions is evaluated for homogeneity. This is accomplished by determining the average intensity value of the region and the standard deviation. If a certain percentage of the pixels fall within a certain number of standard deviations of the average, the region is considered homogeneous. Presently, the user sets the percentage and the number of standard deviations used to determine homogeneity. For example, the user may set the percentage to ninety-five and the number of standard deviations (listed as homogeneity in the dialog box) to one. Therefore, if ninety-five percent of the pixels within the region fall within one standard deviation of the average, the region is considered homogeneous. If the region is not homogeneous, it is sub-divided into four parts and the process of evaluation starts all over. This method of subdivision creates a quadtree structure internally.

A region which is found to be homogeneous is then compared to the region of interest. Since the region of interest's average and standard deviation represent the nature of the area to be segmented, the homogeneous region's average is compared to these values. If the average value of the homogeneous region falls within a certain number of the region of interest's standard deviation of the region of interest's average, the homogeneous region is considered part of the region of interest. For example:

σR = standara deviation of the region of interest""

μR = average intensity value of the region of interest

μH = average intensity value of the homogeneous region

```
$X$ = float value
    $if(\mu H - \mu R) <= X * \sigma R$
            accept
    else
            reject
```

where X is set by the user (labeled Region in the dialog box).

Once a region is found to be homogeneous, its sibling regions, based on the quadtree structure, if also homogeneous and adequately similar to the region of interest, may be merged.Again, as points within the region of interest are identified, they are added to a points list maintained for each region. As with Region growing, the points lists are used for "undoing" regions in Modify mode and for the determination of statistics of each region.

This split and merge procedure is different from the procedure described in [Gonzalez '92]. The procedure in [Gonzalez '92] was used to separate a single dark object from a uniform light background. Consequently, the acceptance criteria for homogeneity can be fairly broad and the object of interest can be segmented based solely on region homogeneity. Since ultrasound images are noisy, speckled, of poor quality and contain multiple regions of interest on a non-homogeneous background, it was necessary to incorporate a means of designating a region of interest, and establishing a set of acceptance criteria for determining that a homogeneous region is adequately similar to the region of interest. This was accomplished by first identifying a bounding rectangle beyond which the region would not grow. The region of interest is identified by means of a "seed pixel" of sorts which serves only to identify the general area of the region of interest. An eleven by eleven neighborhood centered on this pixel is used to determine the characteristics of the region of interest, specifically, the average intensity value and the standard deviation. These characteristics are then compared to the characteristics of the homogeneous region. Those homogeneous regions whose average intensity values are within a certain number of standard deviations of the average intensity value of the region of interest are marked as part of the region of interest. This algorithm may be seen in Figure 3.6.

```
Get Point in Region of Interest
Determine characteristics of Region of Interest
Split Bounding Region into 4 subregions
Evaluate Subregion 1
If (homogeneous)
{
    If (homogeneous region similar to  ROI)
                 Label as ROI
    else
                 Do nothing
}
else
{
                 Split/Merge (Subregion 1)
}
Evaluate Subregion 2
If (homogeneous)
{
    If (homogeneous region similar to  ROI)
                 Label as ROI
    else
                 Do nothing
}
else
{
                 Split/Merge (Subregion 2)
}

Evaluate Subregion 3
If (homogeneous)
{
    If (homogeneous region similar to  ROI)
                 Label as ROI
    else
                 Do nothing
}
else
{
                 Split/Merge (Subregion 3)
}
Evaluate Subregion 4
If (homogeneous)
{
    If (homogeneous region similar to  ROI)
                 Label as ROI
    else
                 Do nothing
}
else
{
                 Split/Merge (Subregion 4)
}
```

Figure 3.6

Split/Merge algorithm

Region grow is a "local" algorithm which is greatly effected by a pixel's immediate neighbors. Varying the tolerance range by a value of one can greatly alter the performance of the algorithm, and can mean the difference between growing a region contained within the region of interest, and incorporating sections of the image beyond the region of interest. Also, small pockets of pixels within the region of interest may be excluded by the region grow procedure if the intensities of these pixels fall outside the tolerance range. This may artificially create holes within the region of interest.

Split and merge on the other hand, is more global in nature, being effected by a much larger area of an image. Consequently, a few pixels which would be excluded by the tolerance range of region grow, may be incorporated into a region based on the values of the majority of the pixels in the subregion under consideration. In addition, since split and merge is not tied to connectivity, outlying regions lost by region grow may be brought into the region of interest. While the use of characteristics of a region of interest, as determined by a small neighborhood within the region of interest, increases the sensitivity of split and merge to local factors, it still provides a more global approach to segmentation.

Since region growing provides connectivity and split and merge incorporates many outlying regions, as well as "pockets" within the region of interest, the two procedures, used together, may offer a performance superior to either algorithm used alone. This overlay procedure is used to explore that possibility.

Each procedure is performed independently on the original (filtered) image, and a points list created for each procedure. The region growing points list is considered first and those pixels set to the appropriate color. When the split and merge points list is considered, if a point is encountered which was grown by region grow, the color remains set to the region grow color. Only pixels not included in region grow, but included by the split and merge procedure are set to the split and merge color. This allows the user to see what has been contributed by split and merge which was not identified by region growing. Altering the order in which the points lists are considered would allow the user to see what region grow contributes which split and merge does not. This presently must be altered within the C code itself.

4 EXPERIMENTATION

Three data sets were used to generate six sequences of images used for testing this software. All data sets were gathered at the Women's Institute at Carolina's Medical Center in Charlotte, North Carolina. The first data set was collected in 1995 from an ultrasound exam of an eight month old fetus. It was digitized using the ACCOM digital recorder. Data set two was generated during an exam of a seventeen week old fetus in January 1997 and was digitized using the Diskus digital recorder. The third data set was obtained from an exam in November 1996 of a five month old fetus. It was also digitized using the Diskus digital recorder. Data set one generated Sequence 1 and Sequence 2. Data set two generated Sequence 3, Sequence 4 and Sequence 5. Data set three generated Sequence 6. Sequence 1 consists of thirty frames showing the eye sockets, while Sequence 2, also thirty frames, consists of images showing the chambers of the heart. Images used in each of these two sequences were resampled to twice the original height and width prior to testing. Sequence 3 is a series of fifty frames showing the face. As above, the regions of interest are the eye sockets. Another series of forty-nine frames showing the face is used in Sequence 5. Sequence 4 is a series of fifty frames showing the fetal heart. Because of the orientation, only two regions were used for segmentation. Sequence 6 is a series of fifty frames showing the fetal heart. Again, orientation allows for the segmentation of only three regions of interest. Sequences 3, 4, 5 and 6 were not subjected to resampling prior to testing.

Preliminary testing was done on images from Sequence 1 to evaluate the performance of region growing based on the type and size of filter used. Evaluation was based on the number of pixels grown in the region. All tests used the same seed pixel to provide consistency. The large numbers (five or six digit numbers) indicate what came to be referred to as a "bleed - out". A bleed - out was growth of the region outside the area which should be grown. This is generally attributed to the lack of data (i.e. boundary data) in the images, and the sensitivity to local factors, such as seed pixel intensity value and neighbor intensity ,by the region growing procedure. Sometimes, as can be seen in Table 1, a bleed - out could be prevented by altering the size of the filter used. However, what helped one frame did not necessarily help another frame. It was also determined that altering the tolerance range of the region growing procedure greatly effected the performance. Changing the tolerances one could cause a region to bleed-out too. Again, indicating the procedure's sensitivity to local factors.

The next step was to segment a sequence of images. At this stage, the package was not interactive. Therefore, all parameters and the seed pixel were fixed. Figure 4.1 shows a sequence of three frames (10, 11and 12 clockwise from the left), processed in this manner. These images were processed through a 5 x 5 median filter and regions segmented using a tolerance level of 7 for seed pixel 108, 116 and 5 for seed pixel 278, 104. The region grown on the left (seed pixel 108, 116) is very consistent from frame to frame, thus indicating the benefit of coherence. The region on the right however, shows how performance can change from frame to frame. Frames 10 and 12 in figure 4.1

Table 1

Frame #	Filter	Filter size	Region #	Count
15	Median	3 x 3	1	1296
			2	51
	Median	5 x 5	1	1309
			2	157111
	Median	7 x 7	1	1431
			2	1933
	Median	9 x 9	1	1436
			2	1595
	Gaussian	3 x 3	1	1222
			2	62
	Gaussian	5 x 5	1	1230
			2	28788
	Gaussian	7 x 7	1	1227
			2	28819
	Gaussian	9 x 9	1	1227
			2	28407
127	Median	3 x 3	1	1559
			2	28713
	Median	5 x 5	1	1541
			2	30161
	Median	7 x 7	1	1601
			2	2181
	Median	9 x 9	1	1586
			2	2104
	Gaussian	3 x 3	1	1482
			2	33982
	Gaussian	5 x 5	1	1500
			2	29809
	Gaussian	7 x 7	1	1496
			2	29711
	Gaussian	9 x 9	1	1496
			2	29658

Effects of filter size and type on Region grow exhibit right regions which only grew only a small part of the eye socket, while the right region in figure 4.1 suffers an extreme bleed - out. Interactive segmentation would allow the user to immediately correct this problem by allowing the modification of the parameters used to grow the regions. In this case, the bleed-out in frame 11 could be reduced or removed by decreasing the tolerance range while the regions in both frame 10 and 12 could be better filled by increasing the tolerance range.

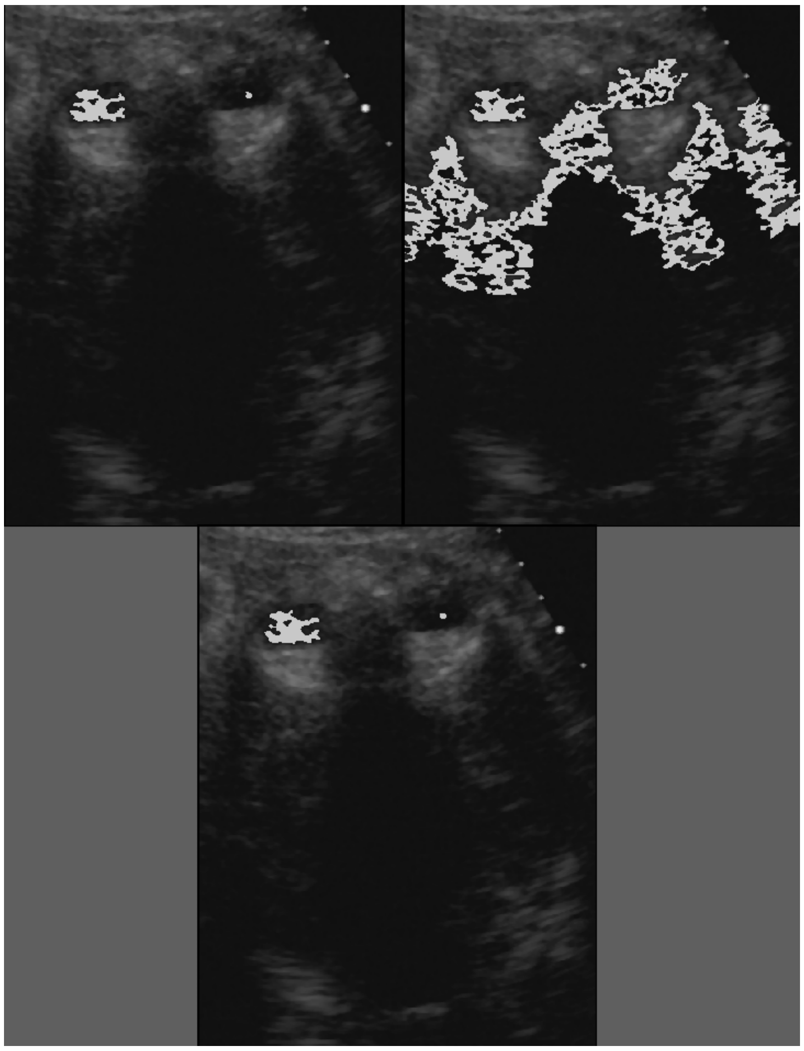


Figure 4.1

Frames 10, 11 and 12 (left to right top to bottom) processed in sequence using region grow and hard coded seed pixels and tolerance ranges.

Following the modification of the software to allow interaction, a sequence of images was again segmented. Using the same frames, seed pixels, filter and filter size, figure 4.2 was created. The regions grown in these frames were grown interactively. By allowing the user to alter the parameters, more complete filling of the region was possible in these frames. The segmentation was not tied to a certain tolerance range, but the tolerance range varied from frame to frame. Therefore, the interactive system shows a decided improvement over a more automated version , where improvement is measured by how well the regions are grown.

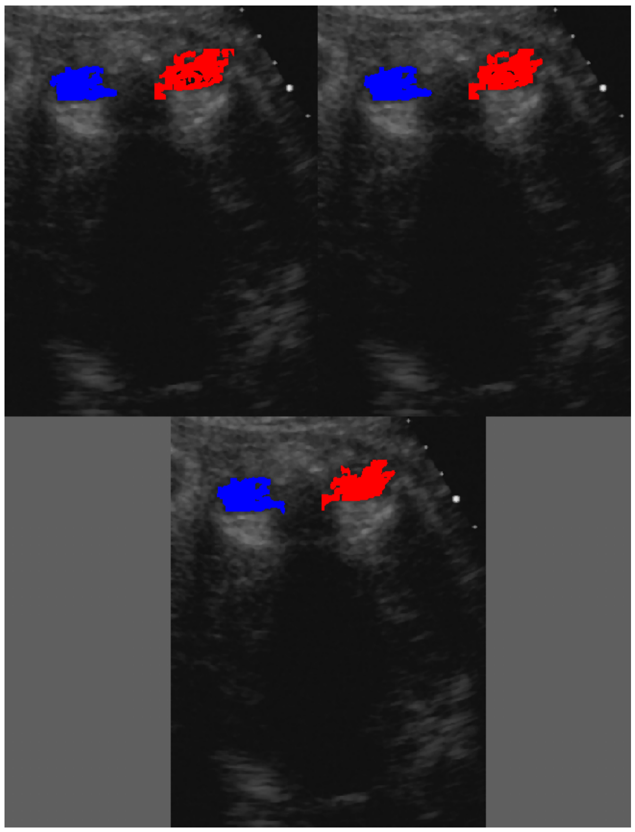


Figure 4.2

Same frames as in Figure 4.1. This time processed interactively.

Split and merge was implemented and tested on a number of frames, for visual evaluation. It provided some outlying regions not grown by region growing. Some of these outlying regions were completely outside the area of interest, while others were "pockets" within the region of interest, which region grow had missed, due to local sensitivity. Therefore, it appeared that the possibility of improving performance by combining the two procedures would be promising.

Further examples of the original images and the corresponding segmented images may be seen in figures 4.3, 4.4 and 4.5. On the left are the original images and on the right are the segmented versions.

Figure 4.3 shows Sequence 1 and Sequence 2. The image on the top right was processed through a 5 x 5 median filter and segmented using split and merge. The heart image of Sequence 2 on the lower right was also processed through a median 7 x 7 filter, but was segmented using the region growing algorithm. Figure 4.4 shows Sequence 3, Sequence 4 and Sequence 5, top to bottom. Again, originals are on the left and the corresponding segmented images are on the right. The top image was filtered with a 7 x 7 median filter and segmented using the overlay procedure. The middle image was filtered with a 5 x 5 median filter and segmented using the region growing procedure. The lower image was also filtered through a 7 x 7 median filter and was segmented using the split and merge algorithm. Finally, an example from Sequence 6 is shown in Figure 4.5. The image on the right was filtered with a 7 x 7 median filter and segmented by region growing. As mentioned earlier, only three regions were reasonably grown in this series of frames. Figure 4.6 shows a comparison of the performance of region grow (top left), split and merge (top right), overlay (bottom left) and overlay followed by a connected components procedure (bottom right). Tables 2 and 3 give a full listing for figures, 4.3, 4.4, 4.5 and 4.6, including the frame number, filter, filter size, segmentation procedure, parameters and seed pixels. For each region grown using split/merge the criteria for homogeneity was 95 percent of the pixels had to fall within 1 standard deviation of the region average.

Table 2

Figure	Frame	Filter	Filter Size	Seg. Proc
4.3 top right	0	Median	5 x 5	Split/Merge
4.3 bottom right	60	Median	7 x 7	Region Grow
4.4 top right	433	Median	7 x 7	Overlay
4.4 middle right	1268	Median	5 x 5	Region Grow
4.4 lower right	40	Median	7 x 7	Split / Merge
4.5 lower	1002	Median	7 x 7	Region Grow
4.6 top left	0	Median	7 x 7	Region Grow
4.6 top right	0	Median	7 x 7	Split / Merge
4.6 bottom left	0	Median	7 x 7	Overlay 4 colors
4.6 bottom right	0	Median	7 x 7	Connected Component

Processing for figures 4.3 - 4.6

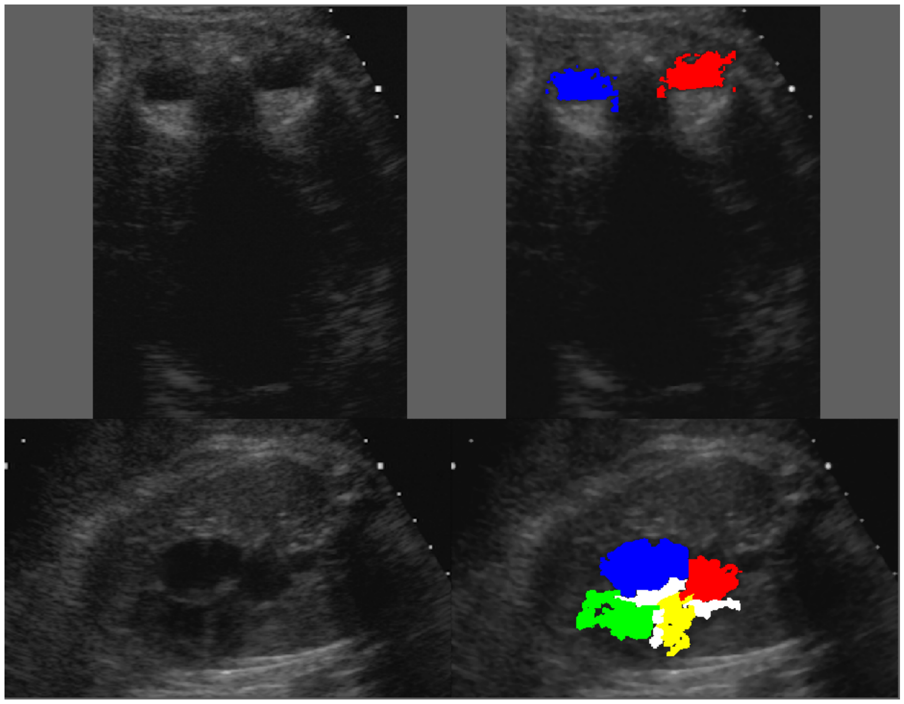


Figure 4.3

Sequence 1-Split/Merge (Top)   Sequence 2-Region growing (Bottom)

Table 3

Figure	Region #	Seed pixel	T	R
4.3 top right	1	106, 112		7.5
	2	273, 96		6.1
4.3 lower right	1	296, 284	3	
	2	293, 317	13	
	3	295, 304	15	
	4	377, 266	23	
	5	345, 265	8	
	6	282, 209	10	
	7	370, 223	9	
	8	250, 283	11	
	9	311, 268	7	
	10	315, 241	30	
4.4 top right	1	114, 31	8	2.6
	2	107, 91	9	2.5
4.4 middle right	1	160, 95	6	
	2	164, 119	7	
4.4 lower right	1	108, 45		2.0
	2	104, 108		2.7
4.5 lower	1	141, 84	14	
	2	176, 94	5	
	3	158, 161	8	
4.6 top left	1	292, 209	12	
	2	297, 243	12	
4.6 top right	1	292, 209		18.0
	2	297, 243		6.5
4.6 bottom left	1	292, 209	12	18.0
	2	297, 243	12	6.5
4.6 bot tom left	1	292, 209	12	18.0
	2	297, 243	12	6.5

Parameters for growing regions in figures 4.3 - 4.6

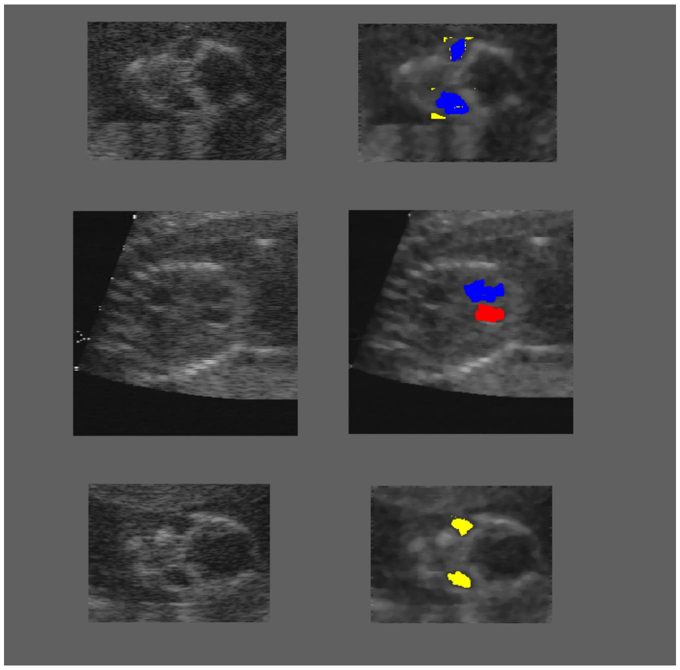


Figure 4.4

Sequence 3-Overlay (Top) Sequence 4-Region growing (Middle)
Sequence 5-Split/Merge (Bottom)

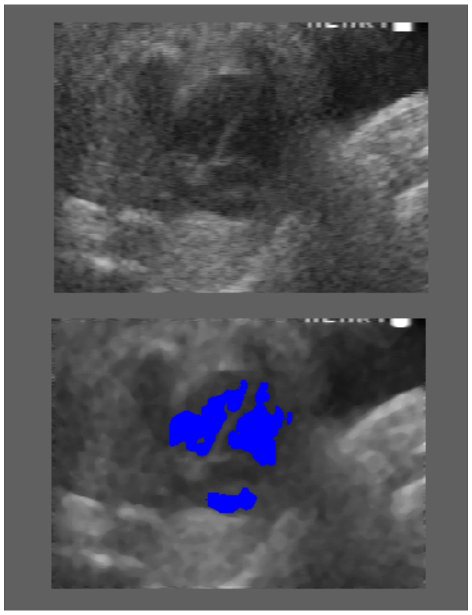


Figure 4.5

Sequence 6-Region growing

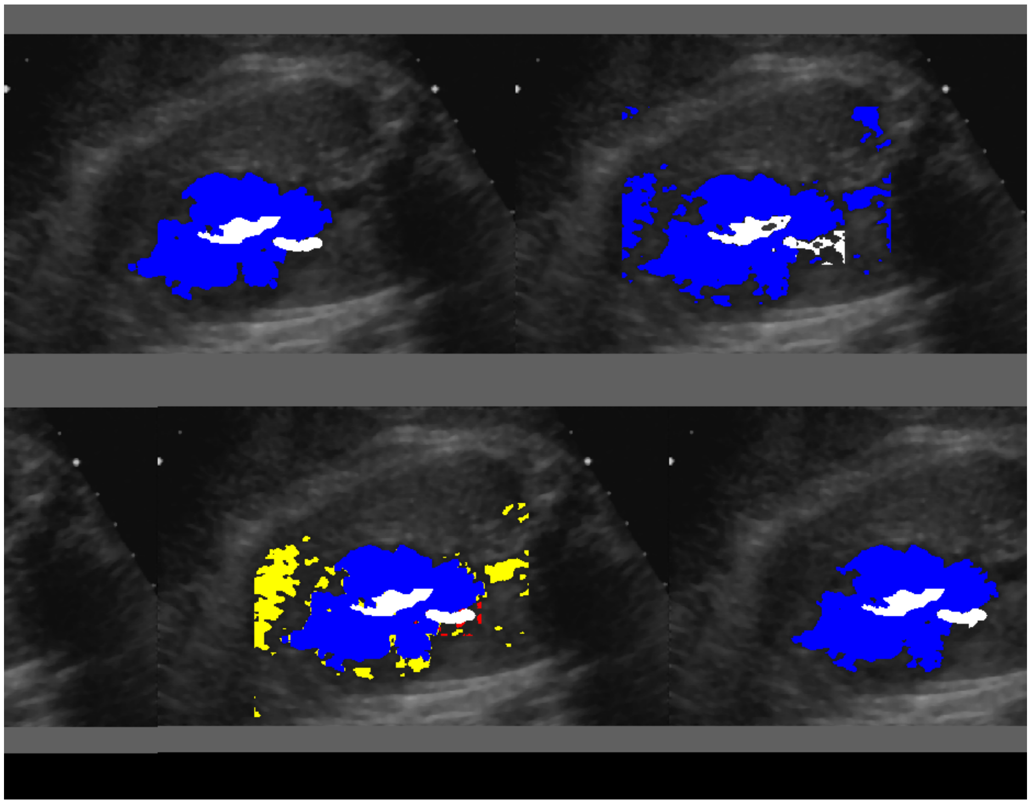


Figure 4.6

Sequence 2 Region growing(top left) Split/Merge(top right) Overlay(bottom left)Connected Component(bottom right)

As a visual comparison of region growing, split and merge and overlay, figure 4.6 shows region grow on the upper left, split and merge on the upper right, overlay on the lower left and overlay followed by a connected components procedure to remove the regions extraneous to the region of interest, on the lower right. Two separate regions were grown, as indicated in Table 3. The overlay procedure shows the regions grown by region growing in blue and white. The corresponding regions grown by split and merge are shown in yellow and red, respectively. It can be seen that each procedure grows some part of each region not grown by the opposite procedure, therefore improving the overall performance.

In order to better evaluate the performance of overlay, as compared to region growing and split and merge, a connected components procedure was added. First of all, this isolates only those points connected to the region of interest which have been provided by region grow or split and merge. Extraneous points, or groups of points, identified by split and merge are removed and the total number of points in this region determined. In addition, a means of color coding sets the points identified by both procedures to a given color, presently white. Points identified solely by region grow are set to the color set by the user for region grow and those points identified solely by split and merge are set to its corresponding color. This allows the user to see exactly which points each procedure adds to the region which the other procedure does not. An example of this may be seen in figure 4.7.

The top row (left to right) consists of the original image, the image processed through region grow, and the image processed through split and merge. The bottom row (again left to right) consists of the image processed using overlay (all points), overlay followed by the connected component procedure, and overlay followed by the connected components procedure and post-processed through dilation. Dilation serves to remove "holes" within a region and to smooth the edges of the region. Table 4 provides information on the size of each region, in pixels, grown by region growing and split and merge and the parameters used. In addition, the number of pixels within each connected component is listed, as well as the number of pixels added by dilation.

In each region grown in figure 4.7 an improvement, based on the number of pixels added to a region, was seen. Individually region grow and split and merge were performed to grow the best region possible without a "bleed-out". These parameters were then used to perform the overlay procedure. Those pixels supplied by either procedure which were connected, were grown by repeating a call to region grow through the connected component procedure. As indicated in Table 4, under connected component, there was an increase in pixel number over region grow, split/merge or both. This shows that more of the region of interest was grown. In addition, even more of the respective regions were grown by following the connected component procedure with dilation postprocessing. See also [Subramanian-Lawrence-Mostafavi '97].

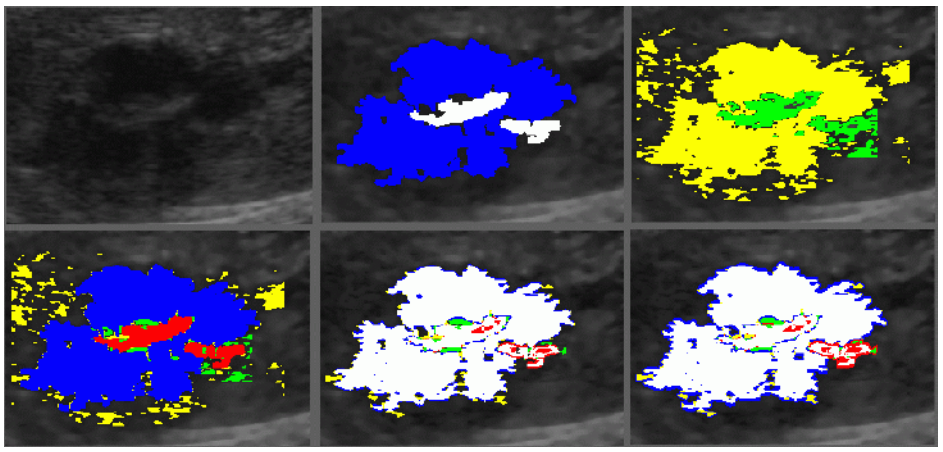


Figure 4.7

Top: Original - Region grow - Split/Merge
Bottom: Overlay-Overlay/Connected component-Overlay/Connected Component/Dilation

Table 4

Region #	seedpixel	T	R	RG	S/M	CC	Dil
1	238, 124	12	.8	305	494	369	102
2	207, 125	5	1.1	551	481	563	121
3	117, 75	11	12.5	19188	20695	19904	1686
4	144, 114	16	2.1	1575	1823	1860	22

Figure 4.7 Region information

T = Tolerance range R = number of standard deviations for acceptance

RG = Region grow S/M = Split / Merge

CC = Connected Component Dil = Dilation

Testing the rapidity with which a sequence of images could be segmented was carried out using these six sequences. Table 5 indicates the size of the images in the sequences, the number of frames in each sequence and the number of regions grown. Table 6 shows the time, in minutes, taken to filter each sequence and to segment each sequence, using each procedure. The number of images which were altered in each sequence are provided in parentheses beside each time. For testing purposes, each sequence was filtered using a median 7 x 7 filter, except for Sequence 4, which used a 5 x 5 median filter.

Table 5

Sequence	Image Size	# of Frames	# of Regions
Sequence 1	450 X 590	30	2
Sequence 2	640 X 400	30	2
Sequence 3	230 X 160	50	2
Sequence 4	260 X 260	50	2
Sequence 5	210 X 160	49	2
Sequence 6	325 X 235	50	3

Information on each sequence used in testing

Table 6

Sequence	Filtering	Regiongrow	Split/Merge	Overlay
Sequence 1	5	3 (2)	3 (3)	5 (3)
Sequence 2	5	6 (5)	5 (8)	7 (8)
Sequence 3	1	5 (6)	6 (11)	5 (3)
Sequence 4	1	6 (9)	8 (8)	9 (8)
Sequence 5	2	5 (5)	5 (8)	4 (2)
Sequence 6	2	8 (14)	7 (9)	10 (9)

Time (in minutes) to filter and segment each sequence.
Numbers in parentheses are number of frames altered

One of the primary purposes of this work was to create a software package which would allow a user to pre-process and segment a series of images in a reasonable amount of time. As is shown in Table 6, the filtering process was taking from 1 minute to 5 minutes. The larger images in Sequence 1 and Sequence 2 naturally required more time for median filtering. These times may be expected to decrease using other filtering methods, such as Gaussian, or by using smaller filters.

The times required for segmenting each series of images, also given in Table 6, are comparable for each segmentation procedure. Times ranged from three minutes to ten minutes. The variations in timing may well be related to the quality of the images used, as well as, the subjectivity present in the user's decision to accept or alter any given frame. Poorer quality images require more modification of regions from frame to frame, thus increasing the overall segmentation time.

Secondly, to facilitate the segmentation of sequences, the coherence from frame to frame was utilized by using parameters from the preceding frame to grow regions in the current frame. The effectiveness of this may be seen in the number of frames altered, listed in the parentheses next to the times in Table 6. A high degree of coherence decreases the number of frames which must be altered.

Again, with poorer quality images, more images must be altered. It may also be seen that in test Sequence 3 and test Sequence 5 there were fewer images altered using the overlay procedure than using either the region growing or the split and merge procedures alone. In the remaining test sequences, the number of images altered in the overlay procedures were comparable to the number altered using the other procedures. While it did not occur in all sequences, the performance in Sequence 3 and Sequence 5 indicates that the combination of the two processes may offer an improvement over the performance of either procedure alone, since processing a sequence through the overlay procedure requires the modification of fewer images.

5 Conclusions and Future Work

Based on the findings listed in Chapter 4, a number of conclusions may be drawn. First, there is a frame to frame coherence present in the ultrasound images, as was seen in the initial set of images segmented in sequence and viewed in figure 4.1. By taking advantage of this coherence, the time required to segment a series of images can be greatly decreased. This was seen by the relatively low number of images which were altered during the segmentation process. Instead of each image in a sequence requiring individual segmentation, the parameters used in one frame may be applied to the next frame. Even if modification is required, the previously used parameters offer a starting point for that modification. All of this acts to decrease the time required to segment a sequence of images, which is of major importance in a clinical situation.

Next, the quality of the ultrasound images, and the knowledge base required for adequate interpretation and evaluation of the segmentation of these images necessitates an interactive system, as presented here. Improvement in the quality of the images obtained, as well as a method for removing some of the motion present may aid in developing a more automatic system.

Third, based on the testing done throughout this project, no singular procedure, region growing or split/merge, was found to perform better on a given data type (heart or face). Each procedure presents certain problems in segmentation of ultrasound images. Region growing's sensitivity to filter size and type, seed pixel intensity value, neighbor's intensity values and tolerance range make it very difficult to get optimum growth of a region without bleeding out into the surrounding area and possibly incorporating too much of the image. A great deal of trial and error was involved in determining a seed pixel and tolerance range for testing purposes. Once adequate parameters were found for a region in a particular frame, the variation from frame to frame, in some cases, often created such variation in performance of the region growing procedure as to negate any benefits provided by using previous parameters. Perhaps altering the software to allow the user to also change the seed pixel when in Modify mode would help address this problem.

Split/merge, on the other hand, often incorporates too much of the image, including areas well outside the bounds of the region of interest. The use of the 11 x 11 neighborhood in determining the characteristics of the region of interest make it more susceptible to local factors, especially seed pixel intensity value and neighbor's intensity value. Depending on the data set used, split/merge may leave more holes in a region than region grow, or fewer holes. Again, perhaps an alternate variation of the split/merge procedure may provide better results.

Finally, it may also be concluded that the use of the overlay procedure can improve the segmentation of ultrasound images, in some cases. This was seen through both the visual inspection of figure 4.6 and 4.7, as well as the region size in pixels listed in Table 4. In addition, the decrease in the number of frames modified during the overlay segmentation of Sequence 3 and Sequence 5, as compared to region grow and split and merge, also indicate an improvement. However, while split/merge occasionally adds some pixels to a region grown by the region grow procedure, it is not consistant enough to state without reservation that there will always be an improvement. Clearly more research is needed in this area. There may be a more intricate means of combining the two procedures. In addition, other methods of segmentation should also be added to the software to provide further versatility. Further modification of the region grow and split and merge procedures should also be considered. Broadening the region grow algorithm to utilize acceptance criteria based on a neighborhood instead of the intensity of one seed pixel may prove beneficial, especially given the nature of the ultrasound data. In addition, using BSP trees to represent the images may also aid in segmentation. Subramanian and Naylor discuss the use of BSP trees in medical imaging in [Subramanian '92]. The use of hyperplanes may help define boundaries which are fuzzy or lacking in the ultrasound data. This would greatly aid segmentation.

While these modifications may help in the segmentation of ultrasound images, perhaps the biggest improvement needs to come from the area of data acquisition. Given the quality of data obtained from the CT and MRI modalities, and the performance of standard segmentation procedures on that data, some means of improving the quality and consistency of data acquired from ultrasound, while maintaining ultrasound's non-invasive nature and availability may be the greatest aid to segmentation.

BIBLIOGRAPHY

[Bartrum - Crow '77] R.J. Bartrum and H.C. Crow. Gray-Scale Ultrasound: A manual for Physicians and Technical Personnel. W.B. Saunders Company, Harcourt Brace Jovanovich, Inc., 1977.

[Bezdek '93] J.C. Bezdek, L.O. Hall, and L.P. Clark. Review of mri segmentation images using pattern recognition. Medical Physics, 20(4):1033-1048, 1993.

[Gonzalez '92] R.C. Gonzalez and R.E. Woods. Digital Image Processing. Addison Wesley, 1992.

[Haralick ’92] R.M. Haralick and L.G. Shapiro. Computer and Robot Vision. Addison Wesley, 1992

[Kremkau '89] F.W. Kremkau. Diagnostic Ultrasound. W.B. Saunders Company, Harcourt Brace Jovanovich Inc., 1989.

[Lengyel '95] J. Lengyel, D. Greenberg, and R. Popp. Time-Dependent Three-Dimensional Intravascular Ultrasound. Computer Graphics Proceedings, pages 457 - 464, 1995.

[Meinzer '92] Hans-Peter Meinzer, K. Meetz, D. Scheppelmann, U. Englemann,and Hans Jurgen Baur. Segmentation of Medical Images. 3D Visualization in Medicine ACM SIGGRAPH ’92.

[Ousterhout '94] J.K. Ousterhout. Tcl and the Tk Toolkit. Addison Wesley, 1994.

[Sakas '95] G. Sakas and S. Walter. Extracting surfaces from fuzzy 3d-ultrasound data. ACM Computer Graphics Proceedings, pages 465-474, August 1995.

[Sakas '95 Pre] G. Sakas, L. Schreyer, and M. Grimm. Preprocessing, segmenting and volume rendering 3d ultrasonic data. IEEE Computer Graphics and Applications, 15(4), July 1995.

[Subramanian '92] K.R. Subramanian and B.F. Naylor. Representing medical images with partitioning trees. In Proceedings of Visualization '92, Boston, MA, Oct. 19-23, 1992.

[Subramanian-Lawrence-Mostafavi ’97] K.R. Subramanian, D.M.Lawrence and M.T. Mostafavi. Interactive Segmentation and Analysis of Fetal Ultrasound Images. Eighth EG Workshop on ViSC, Boulogne sur Mer, 28 - 30 April, 1997.

[Thrall '86] D. Thrall. Textbook of Veterinary Diagnostic Radiology. W.B. Saunders Company, 1986.

[Wolbarst '93] Anthony Brinton Wolbarst. Physics of Radiology. Appleton and Lange, 1993.