Digital image enhancement of B-scan echocardiograms

Donald Edward Gayou
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Part of the Electrical and Electronics Commons

Recommended Citation
Gayou, Donald Edward, "Digital image enhancement of B-scan echocardiograms" (1979). Retrospective Theses and Dissertations. 7279. https://lib.dr.iastate.edu/rtd/7279

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This was produced from a copy of a document sent to us for microfilming. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help you understand markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure you of complete continuity.

2. When an image on the film is obliterated with a round black mark it is an indication that the film inspector noticed either blurred copy because of movement during exposure, or duplicate copy. Unless we meant to delete copyrighted materials that should not have been filmed, you will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed the photographer has followed a definite method in “sectioning” the material. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For any illustrations that cannot be reproduced satisfactorily by xerography, photographic prints can be purchased at additional cost and tipped into your xerographic copy. Requests can be made to our Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases we have filmed the best available copy.
GAYOU, DONALD EDWARD

DIGITAL IMAGE ENHANCEMENT OF B-SCAN ECHOCARDIOGRAMS

Iowa State University

University Microfilms International

Copyright 1979

by

Gayou, Donald Edward

All Rights Reserved
PLEASE NOTE:
In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs □
2. Colored illustrations □
3. Photographs with dark background ✓
4. Illustrations are poor copy □
5. Print shows through as there is text on both sides of page □
6. Indistinct, broken or small print on several pages □ throughout □
7. Tightly bound copy with print lost in spine □
8. Computer printout pages with indistinct print ✓
9. Page(s) □ lacking when material received, and not available from school or author □
10. Page(s) □ seem to be missing in numbering only as text follows □
11. Poor carbon copy □
12. Not original copy, several pages with blurred type □
13. Appendix pages are poor copy □
14. Original copy with light type □
15. Curling and wrinkled pages □
16. Other □
Digital image enhancement of B-scan echocardiograms

by

Donald Edward Gayou

A Dissertation Submitted to the Graduate Faculty in Partial Fulfillment of the Requirements for the Degree of DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Approved:

Signature was redacted for privacy.

Signature was redacted for privacy. In Charge of Major Work
Signature was redacted for privacy. For the Major Department
Signature was redacted for privacy. For the Graduate College

Iowa State University
Ames, Iowa

1979

Copyright © Donald Edward Gayou, 1979. All rights reserved.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>DIGITAL IMAGE PROCESSING</td>
<td>10</td>
</tr>
<tr>
<td>NEW ENHANCEMENT OPERATOR</td>
<td>26</td>
</tr>
<tr>
<td>PERFORMANCE TESTING</td>
<td>48</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>108</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>114</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>117</td>
</tr>
</tbody>
</table>
INTRODUCTION

Background

Digital image processing techniques are assuming an increasingly important role in many areas of science and technology. At present, many challenges exist in the area of medical image processing. In contrast to many industrial applications, where considerable a priori information is available, medical images are unpredictable in advance. For example, chest X-rays all contain ribs in a generally similar orientation. However, the exact dimensions, positions, shapes, densities, etc., are all variable, making automated analysis extremely difficult. Nonetheless, the increasing use of diagnostic imaging systems presents a compelling reason to explore such problems. One problem of immediate concern is the detection of heart disease, a leading cause of mortality in the United States.

Traditionally, physicians have relied on X-ray imaging systems to evaluate cardiac function. Since conventional X-ray is unable to image soft, moving tissue, catheterization of the heart and injection of radioopaque dye is necessary to display a cardiac cross-section as a cineangiogram. Of course, this procedure presents an inherent risk to the patient.

More recently, nuclear imaging techniques have become available which offer many advantages over cineangiography.
However, these methods are complicated, requiring extensive investments, in imaging equipment, facilities, and staff expertise. Clearly, what is needed is an imaging modality which is noninvasive, nonradioactive, and cost effective. The rapid increase in the use of medical ultrasound is directly attributable to these diverse features.

The early use of ultrasound in cardiology resulted in a very narrow, single-beam scan of a very limited area of the heart. With technological advances, it has become possible to "steer" the beam during the cardiac cycle to image different parts of the heart. Thus, it is possible to obtain two-dimensional cross-sections of the heart in real-time.

Unfortunately, these images suffer from various types of noise and degradation. The poor quality of these images arises for a number of reasons. First, the heart is an irregular object, undergoing complex motion. Its tissues are by no means homogeneous, and acoustic impedances are exceedingly complex. Secondly, the phased-array scanner suffers from certain defects, such as sidelobes, and other nonlinearities which create artifacts.

To improve or enhance these images, two alternatives exist. First, one may examine the imaging system itself, and through superior design, improve its performance characteristics. A second alternative is to somehow improve the quality of the images appearing on the screen without considering the
modality which generates them.

With these two strategies in mind, it was decided to enhance the images by applying methods from the relatively new field of digital image processing. If successful, techniques found useful for enhancement of B-scan images may also be useful for other gray scale images.

It is the purpose of this research to find techniques for enhancement of these images so that further investigations of automated measurement may proceed. Ideally, the image processing algorithms developed here will be generally useful for gray-scale images from a variety of sources.

In the following section, techniques for computer processing of medical ultrasonic images will be reviewed. However, before reviewing the literature, a brief explanation of ultrasonic principles is in order.

Ultrasonic waves are acoustic waves containing frequencies of several megahertz. These waves are generated when a piezoelectric crystal is shock-excited with a high-voltage pulse. If the crystal is suitably coupled to the anterior chest, an acoustic wave will propagate away from the transducer at a speed of approximately 1500 meters per second. As the acoustic wave encounters various cardiac structures, a fraction of its energy is reflected at each interface back to the transducer, with the exact quantity a function of interface tissue impedance and geometry. When the reflected echoes
reach the piezoelectric crystal, they create a voltage, which is suitably amplified and displayed on a CRT. Customarily, a display of returning echo amplitude versus time is called an A-scan. A more useful display for moving structures is called an M-scan, which displays object position as a function of time. With this type of display, a stationary object returning echoes will be imaged as a horizontal line. However, an object such as a heart valve, moving alternately closer to and further from the transducer, will be displayed as a sinusoid. The M-scan display is used extensively in echocardiography.

In order to generate a two-dimensional or B-scan, it is necessary to generate multiple beams using multiple transducers, or by moving a single transducer. With this introduction, the literature search may proceed.
LITERATURE REVIEW

To begin, a rather thorough review of ultrasound in medicine has been presented by Erikson, Fry and Jones (1). Unfortunately, specific reports of computer processing of medical ultrasound images are relatively sparse. Only three papers were found dealing with the enhancement of M-scan images.

Ledley and Wilson (2) performed one of the early analyses of an M-scan image of the left ventricle. This paper is interesting in that the authors performed extensive operations on the image in order to render it suitable for ventricular measurement. Also, all of this processing was performed automatically without human intervention. After digitization, the raw image is smoothed by a moving average filter. Then, the edge points are grouped into chains, the longest of which are connected together, and the remainder are removed. The resulting waveform is then further smoothed via harmonic analysis. After these steps have been taken, actual measurements may be taken to estimate cardiac parameters. While the various enhancement techniques used in this paper perform fairly well, they do not seem capable of being readily extended to other types of images.

As an alternative to complex computer analysis, Decodt, Mathey, and Swan (3) used a digitizing table to acquire the waveform, which then undergoes smoothing and analysis. In this instance, a human detects significant borders and discards nonsignificant ones.
In yet another application, Hirsch et al. (4) traced the wall motion of the ascending aorta using an automated tracking technique which requires human intervention to locate the starting point for the tracking algorithm. After tracking has concluded, the image is low pass filtered to reduce noise.

While these three papers exhibit considerable ingenuity, it was becoming increasingly clear that meaningful measurements could not be obtained from M-scan techniques. Therefore much effort was expended in the development of B-scan systems, capable of showing two-dimensional cross-sections of the heart. A number of techniques for doing this are possible.

One method is to use a single, manually scanned transducer whose coordinates control the position of successive M-scans on a CRT to form a cross-sectional display. A report of this method, using an EKG signal for synchronization was reported by King (5). Another technique is to mechanically oscillate a transducer, again with appropriate position transducers, to yield a real-time display of the heart. Such a system was successfully demonstrated by Griffith and Ledley (6).

An electronic approach to real-time imaging was presented by Bom et al. (7), who constructed a linear array of small transducers, in effect creating a multiple transducer linear
M-scan system. The disadvantage of this system is that the scan is linear, with a direct relationship between transducer array size and area imaged. Unfortunately, the heart may only be imaged from a number of windows of size determined by ribs, lung volume, etc. Thus, it is difficult to view a large cross section of the heart with this system.

A general solution to the real-time imaging problem was proposed by Somer (8), who suggested that all elements of a miniature piezoelectric array be fired almost simultaneously, with only a slight delay between successive transducers. In this manner, the individual wavefronts from each transducer form a single beam whose angle depends on the inter-transducer phase delay. Therefore, the beam can be steered in real-time.

An elegant implementation of this concept was reported by von Ramm and Thurstone (9). This system featured a Digital Equipment Corporation PDP-11 minicomputer controlling phased-array excitation to allow beam steering and variable delay lines to realize dynamic focusing of returning echoes. A number of systems are now commercially available based on this type of design. These systems image at thirty frames per second, with a ten to sixteen gray level CRT display. The transducer is small enough to be handheld, and the beam can be steered through an angle of sixty to eighty degrees.

As a result of these advancements, there has been wide-
spread acceptance of two-dimensional real-time scanning. In a 1978 article, Tajik et al., (10) report on twenty acoustic windows located on the chest where a transducer may be placed to obtain views of the heart and other great vessels.

A number of studies have begun to attempt quantification of cardiac geometry, such as Wyatt et al., (11).

Needless to say, before any high-level analysis can begin, object borders must be accurately determined. Although a human using a light-pen can perform these tasks, several studies have appeared which attempt to automatically enhance B-scan images.

Robinson et al. (12), using a scanning technique similar to King's, attempted to process a two-dimensional left ventricular cross-section to locate borders and reduce noise. In their system human intervention is required to draw in outlines of the ventricular border after smoothing with a moving average filter. This outline serves as a good first estimate for the computer to find borders in successive images. Detected borders are stored as linked chains. These chains then undergo extensive processing to determine the border. Essentially, this is a tracking algorithm with complex chain linking procedures.

In another approach, Parker, Pryor and Ridges (13) applied lateral filtering techniques to a two-dimensional image in an attempt to reduce off-axis effects created by the scanner.
The techniques presented in this chapter were designed specifically to process ultrasonic images, and as such do not seem easily extendable to other types of gray scale images. Of greater concern is the fact that the B-scan techniques require human intervention to function with any degree of success. Indeed, currently available measurement modules utilize a light-pen to trace borders and label objects in ultrasonic pictures. Since no general enhancement technique has been found in the ultrasound literature, the relatively new area of digital image processing will now be examined for algorithms capable of enhancing noisy images without considerable human input.
DIGITAL IMAGE PROCESSING

As has been stated in the previous chapter, a major difficulty in attempting high level analysis is that B-scan images are of such poor quality. Indeed, it appears that the enhancement of these images is mandatory before high level techniques can be applied. Therefore, this chapter will examine the field of digital image processing for algorithms which may be useful in the enhancement of digital images. It should be noted that image enhancement will be defined as those operations necessary to either improve the quality of an image for a human, or to permit subsequent high level operations by computer. Also, this review will be restricted to spatial domain algorithms. That is, optical or frequency domain processing techniques will not be considered. Before discussing algorithms, however, it may be desirable to discuss possible strategies for image enhancement.

It has been contended by Riseman and Arbib (14) that most image processing tasks can be divided into low and high level operations. Low level processing would include such tasks as noise reduction or smoothing, edge point detection, and linking of edge points into chains. High level operations would include essentially all pattern recognition techniques, measurements, object identification, and other operations requiring intelligence. In the context of echocardiography,
high level tasks include identification of cardiac structures, measurement of anatomical features and wall motion, and analysis of anomalous structures. With this viewpoint in mind, it was decided that enhancement would be defined for this research as a set of operations applied in sequence to an image. Therefore, image enhancement may be considered as a consecutive series of operations consisting of noise reduction, edge point detection, and edge point linking, as shown in Figure 3.1.

![Image enhancement block diagram.](image_url)

**Figure 3.1. Image enhancement block diagram.**

**Edge detection**

The first segmentation problem to be examined is that of edge detection, an area that has been surveyed by Davis (15). However, it is first necessary to define an image. An image is defined as an integer matrix, with values of integers representing intensity, brightness, or gray scale level. An edge is conventionally considered to be a gray-level discontinuity between two regions or objects. For example, Figure 3.2 represents a gray level step between two segments
of an image.

\[\begin{array}{ccc|ccc}
3 & 3 & 3 & 3 & 7 & 7 \\
3 & 3 & 3 & 3 & 7 & 7 \\
3 & 3 & 3 & 3 & 7 & 7 \\
\end{array}\]

Figure 3.2. Digital Edge
A line is drawn where the edge would commonly be found. However, since brightness is only defined at discrete points, the edge representation must also be defined at discrete points. Clearly, some operator capable of differentiation would be expected to detect this step. However, since differentiation is continuous, subtraction is commonly used. These differences result in a gradient pointing towards the edge. A typical edge detector may be implemented as follows.

Let a point with gray level \(G(9)\) be surrounded by eight neighbors with gray levels as show in Figure 3.3.

\[\begin{array}{ccc}
G(1) & G(2) & G(3) \\
G(8) & G(9) & G(4) \\
G(7) & G(6) & G(5) \end{array}\]

Figure 3.3. Eight-neighbors of a pixel.
It should be noted that the odd index neighbors are further
from $G(9)$. Define a vector from $G(9)$ to each neighbor as follows.

$$V(I) = G(I) - G(9)$$

Then,

$$\text{EDGEVALUE}(9) = \text{VECTORSUM}(V(I)) \quad \text{for all } I$$

To consider several practical cases, if the $G(I)$ are all the same value, the vector sum is zero, and no edge exists at $G(9)$. If several adjacent eight-neighbors have a much higher level than the other, indicating a border, then the vector sum will be large. For a noise-free image, pixels on both sides of an edge will have a high value, reporting an edge with two lines. While this problem can be surmounted with thinning algorithms, this algorithm and almost all gradient mask operators fail in noise, because noise cannot be successfully differentiated. Persoon (16) reports on a typical mask operator. Alternatively, one can generate edges by altering the areas over which the mask operates, as reported by Rosenfeld and Thurston (17).

One mask operator which is of a different type is reported by Kirsch (18). Using the eight-neighbor notation as above, the Kirsch operator replaces the center value as follows.

$$\text{EDGEVALUE}(9) = \max(1, \max(5 \cdot G(I) + G(I+1) + G(I+2) - 3 \cdot G(I+3) + ... G(I+7)))$$

where all coefficients are evaluated modulo eight and $I$
ranges from one to eight. This operator performs well because it attempts to find a locally optimal structural edge.

Usually, mask operators compute a new value for a center pixel by examining gray levels of pixels in a local neighborhood, three by three or five by five, etc. In general, such operators are referred to as local operators.

Very complex local operators are possible, perhaps the most complex of which is the Hueckel operator (19), which is discussed by Shaw (20). This operator defines an edge in a circular disk intersected by two parallel lines. Thus, three regions are created, each with a constant gray level. The edge is defined to lie within the center region, with magnitude equal to the difference of the two outer regions. When this operator is applied to an actual image, the local edge, if any, best fitting the ideal edge model is detected by an optimization technique. Thus, this operator reports the height or magnitude of the edge, its direction, its width, and a measure of goodness, or how closely the edge fits the ideal edge model.

As previously stated, local operators all examine an area close to themselves fitting some definition of edge, and returning some number related to edge. This often presents an additional problem. For example, where should a threshold be set? Above what value does a point define an edge?
The advantage of Hueckel's operator is that it automatically determines whether an edge is present, unlike the simple masks. Therefore, it is probable that multiple edges will be detected. This requires the use of thinning algorithms or nonmaximum suppression techniques, and one must decide which pixels are nonmaximum.

Clearly, most simple local operators produce too little useful information, while complex ones such as Hueckel's require extensive computing time. A completely different approach to edge detection is to use a global operator. For example, the gray level histogram of an image is a global quantity computed over the entire image.

Chow and Kaneko (21) utilize the histogram technique to detect boundaries of the heart in cardioangiograms. Many X-ray images are characterized by their poor quality. Basically, their technique was to subdivide the picture into a number of subregions. In each region a gray level histogram is computed. Presumably, if the subregion is entirely object or entirely background, a unimodal distribution will result. If the region contains a border between object and background, a bimodal distribution will result. A bimodal distribution will contain a valley, at which gray level segmentation can be performed via thresholding. Although the actual technique uses a more sophisticated statistical
approach, the technique outlined above represents the basic idea.

A different class of techniques is referred to as sequential techniques. To find an edge, for example, a starting point is determined. From this point, the algorithm searches or tracks to the next edge point. If, for example, a noise edge is tracked, it will be necessary to somehow backtrack to the correct edge. While such techniques have an intuitive appeal, their main difficulty is that the final path tracked is generally a function of where tracking began. Therefore, careful selection of starting point is crucial. Unfortunately, there is no good method for determining starting point.

In an effort to apply global information in tracking or heuristic search problems as proposed by Martelli (22), and Chien and Fu (23), a number of dynamic programming techniques have been applied. Such methods attempt to find a minimal cost path to a goal, effectively searching a graph. They are assisted by using a global criterion function which incorporates knowledge of the shape to be searched for. Naturally, techniques which utilize a priori information should perform better than those which do not. Unfortunately, most heuristic search techniques are awkward to implement and rather inflexible. In addition, if the search
space is large, vast amounts of computation may be required.

In this section, a number of methods for edge detection have been briefly discussed. It is important to note that if images were noise-free, then the segmentation problem is almost trivial. Since noise is always present to a greater or a lesser degree, it must be dealt with. Otherwise, true edges will be missed, and spurious edges will be generated. In the following section, algorithms for smoothing will be discussed.

Smoothing

Conceptually, if an object may be considered as a large area of constant gray level, and noise is considered as some random shift of each point, then a smoothing algorithm should be expected to reduce discontinuities, until the entire region is again homogeneous.

The very first idea that comes to mind is simply to replace each pixel by some average value over itself and its neighbors. For example, using the notation above,

\[ G(9) = \frac{1}{9} \sum G(I) \quad \text{for} \ I \text{ from one to nine.} \]

This is perhaps the simplest smoothing technique possible, and is shown applied in Figure 3.4.
Figure 3.4. Averaging.
Clearly, if the center point is noise, and of sufficient amplitude, the effect will not be to eliminate it but rather to diffuse, or more precisely, cause linear blur. Thus, noise points will tend to diffuse. If noise points are close together, the problem is compounded. Since the technique is applied to all points in the picture, there is also a deleterious effect on edges denoting boundaries between objects and background. For example, see Figure 3.5.

Figure 3.5. Edge blurring
Thus, what was a sharp, clearly defined edge has now been blurred. Thus, if the averaging process is applied repeatedly, the entire picture will indeed be smoothed,
approaching a constant value. Thus, it can be seen that simple, linear averaging schemes are unable to reduce noise and simultaneously preserve object edges, which carry information. Therefore, it may be concluded that since edges are nonlinear, some nonlinear method may be used to eliminate noise while at the same time preserving edges.

A simple nonlinear algorithm, called the median window, is commonly used for rather effective noise cleaning. The median window operates by finding the median gray value of a point and its eight-neighbors. It can readily be seen that the median window is a good noise reduction algorithm by observing Figure 3.6.

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 9 & 0 & 0 \\
0 & 0 & 0 & 0
\end{array}
\quad \rightarrow \quad
\begin{array}{cccc}
0 & 0 & 3 & 0 \\
0 & 0 & 3 & 0 \\
0 & 0 & 3 & 0
\end{array}
\]

Figure 3.6. Median enhancement.
The algorithm is clearly edge preserving, although it will not preserve corners, as demonstrated in Figure 3.7.

\[
\begin{array}{ccc}
5 & 5 & 3 \\
5 & 5 & 3
\end{array}
\quad \rightarrow \quad
\begin{array}{ccc}
5 & 3 & 3 \\
3 & 3 & 3
\end{array}
\]

Figure 3.7. Median enhancement.
The median window is useful because the median of an array of numbers is unaffected by the extreme values of the array. Thus, the median tends to reject extreme values, unlike the average, which must be affected by all values. Although its computation requires some type of sorting or ordering, it is a useful smoothing algorithm.

Perhaps the main criticism of the median window is that it is not structural. That is, the computation is performed without reference to the actual position of the neighbors relative to the center point. For example, see Figure 3.8.

```
5 5 5 5 5   5 5 5 5 5
5 0 0 0 5   5 0 0 0 5
5 0 0 0 5   5 0 5 0 5
5 0 0 0 5   5 0 0 0 5
5 5 5 5 5   5 5 5 5 5
```

Figure 3.8. Median Enhancement.

A very interesting structural smoothing operator has been reported by Nagao and Matsuyama (24). In their scheme, an elongated region is rotated about a point to be smoothed. At each position, the variance of the pixels in the region is computed. After each complete rotation, the average value of the area with the smallest variance is assigned to the center
pixel. This process is performed for every point in the picture in parallel. The process is then applied to the entire image again. If at any time during rotation, the variance of a neighborhood is found to be zero, then the associated point is left unchanged, providing a termination criterion for the iterative process. Although all points do not terminate simultaneously, successively fewer points undergo changes after each application. Essentially complete convergence is reported in ten iterations, with significant smoothing after several iterations.

The notable features of this smoothing algorithm are that it uses a variance measure to evaluate region smoothness, and that it is structural, assigning different values to different areas around a pixel. Perhaps a fault of this algorithm is that a number of iterations are required, depending, as the authors note, on the noise and shapes of regions in the picture. Otherwise, it illustrates the fact that the power of an algorithm is generally related to its complexity.

Linking

A third type of operation necessary in low level processing is edge point linking. That is, after a noisy image has been smoothed, and edge points detected, how are the points to be linked together to form edges or borders separating objects from background? It would appear that this
particular problem is at the interface between low level and high level processing. For example, given a noisy edge picture, how are the various edges to be associated? Which edges are due to noise, and which are parts of objects sought in the picture? One solution to this problem is to form a limited amount of linking locally. When this is completed, high level processing can begin, perhaps eliminating some edge lines while connecting others. Typically, if edges consist of very short line segments, they are merged if possible to form longer chains.

It is somewhat characteristic of linking algorithms that they are rarely mentioned in detail in the literature. Authors typically report linkers that use such criteria as length of lines, direction, proximity of end points, and intensity, etc.

One linker that is reported is by Nevatia (25). His algorithm groups edge points by orientation. All groups of a particular orientation are then put in a set of transformed coordinate strips. Within strips, pixels within a certain distance are connected first. Then, these strings are linked across strips if various criteria are satisfied. This technique seems to work reasonably well, suppressing randomly oriented edge points while simultaneously tending to preserve edge points within object lines. Perhaps a criticism
of the algorithm is that it only operates on straight lines. While the author states that the algorithm can be generalized to include curved segments, it appears that processing time would be increased.

Thus far in this chapter, algorithms have been discussed which perform edge detection, smoothing, or linking. Characteristically, the algorithms perform only one of these tasks. However, an enhancement technique reported by Zucker, Hummel, and Rosenfeld (26) has received a considerable amount of attention because it performs all of these enhancement functions simultaneously. This method, referred to as relaxation enhancement, is a parallel, iterative operator. That is, each point in the picture is assigned a set of labels corresponding to possible edge orientations. For each label, there is associated a probability that a line of a particular orientation passes through that point. Then, by examining the current probability arrays of the point and its neighbors, the probabilities are updated in the next iteration. For example, assume a point with large horizontal probability is surrounded by vertical probability points. On the next iteration, it will be updated by the preponderance of evidence from the vertical probability neighbors to itself become more vertical. In this manner, noise is reduced, and edges may be enhanced.
There are several problems with this technique. First, since it is iterative, it is no trivial matter to decide when to stop iterating. Secondly, a number of compatibility coefficients must be defined to permit computation of updated values. For an arbitrary image, it is not clear how these numbers should be selected. Finally, the algorithm must be initialized by using edge detectors to provide initial estimates of edge orientation for each pixel. Therefore, the final labeling will be affected by the edge detector used initially.

Perhaps the most important advantage of this method is that it provides a unified, parallel approach to low level processing.

In this chapter, a number of algorithms have been discussed, capable either of smoothing, edge detection, or linking. In general, very simple algorithms require additional information to be useful. For example, it is very simple to threshold an image, but it is another matter altogether to determine the level at which thresholding should occur. More complex algorithms such as dynamic programming suffer from inflexibility, while sequential tracking methods seem incapable of performing efficiently in noise. Other operators, such as Hueckel's, must operate over a region, necessarily missing very small edges. However, small
diameter local operators may be incapable of dealing with wide or blurred edges. The relaxation enhancement method is conceptually interesting, but is difficult to implement for automatic, general-purpose processing.

From this review, it seems clear that a low level enhancement operator should possess several characteristics. First, the operator should be capable of functioning in the presence of noise. The operator should be structural, to extract a maximum amount of information from the image. Also, the operator should be capable of being applied in parallel, in a noniterative manner. In addition, the operator should perform all three of the low level tasks defined above. Finally, the algorithm should be capable of performing without a priori information about the image or noise statistics. In the next chapter, a new enhancement operator will be presented which fulfills many of these requirements.
NEW ENHANCEMENT OPERATOR

In the previous chapter, a number of algorithms were examined capable of smoothing, edge detection, or linking. Unfortunately, none of these algorithms appeared sufficiently powerful to deal with noisy gray scale images, such as those obtained from a phased-array B-scan system. The purpose of this chapter will be to present a new enhancement operator, implemented in three algorithms, and capable of automatic processing of noisy gray scale images. The three algorithms, each based on a common edge model, will be capable of noise reduction or smoothing, edge detection, and edge point linking.

Before proceeding further, however, it is desirable to show results obtained when the smoothing and edge detection algorithms are applied to a series of three test images contaminated with varying amounts of noise. Accordingly, Figures 4.1 to 4.4 show results when the enhancement operator is applied to a square contaminated with noise of standard deviation zero, one, two, and three. Figures 4.5 to 4.6 show results when the operator is applied to a simulated phased-array image depicting a long-axis view of the left ventricle with noise of standard deviation zero and one. Finally, an object with three different gray levels is shown contaminated with noise of standard deviation zero and one in Figures 4.7 to 4.8.
Figure 4.1. Examples of enhancement.

(a) square.  (b) noise, standard deviation = 0.  
(c) smoothing.  (d) edge detection.
Figure 4.3. Examples of enhancement.

(a) square.  (b) noise, s.d. = 2.
(c) smoothing.  (d) edge detection.
(a) square.  (b) noise, s.d. = 3.
(c) smoothing. (d) edge detection.

Figure 4.4. Examples of enhancement.
(a) long axis B-scan.  (b) noise, s.d. = 0.
(c) smoothing.  (d) edge detection.

**Figure 4.5.** Examples of enhancement.
Figure 4.6. Examples of enhancement.

(a) long axis B-scan.  (b) noise, s.d. = 1.
(c) smoothing.  (d) edge detection.
(a) multiple gray level object. 
(b) noise, s.d. = 0. 
(c) smoothing. (d) edge detection.

Figure 4.7. Examples of enhancement.
(a) multiple gray level object.
(b) noise, s.d. = 1.
(c) smoothing.  (d) edge detection.

Figure 4.8.  Examples of enhancement.
The remainder of this chapter will be devoted to defining the new enhancement operator and its three constituent algorithms for noise reduction, edge detection, and edge point linking. Before discussing the algorithms, it will be necessary to present the edge model to be used, and a series of subroutines.

**Edge model**

To begin, assume that a noise-free gray scale digital image is available, consisting of an object of arbitrary shape and gray level against a background of a different gray level. Now, consider a ring or annulus surrounding a pixel. The thickness or width of the ring is assumed to be one pixel, and it is also assumed that the ring is a discrete approximation to a circle. Clearly, if the ring is centered in a background region of the picture, all gray levels contained within the ring will have identical values. This statement is also true if the ring is entirely contained within the object. If, however, the ring is centered on an edge pixel, there will be pixels of two gray values contained in the ring. These pixels will subdivide the ring into two segments, the intersection of which defines the orientation of the edge. In a noisy image, segmentation of the ring is determined by detecting the orientation that results in two segments, within each of which gray values are as similar as possible. If such an edge is detected, the appropriate points within
the ring are labeled as edge points. This is the basic model for an edge.

When this algorithm was conceived, it was implemented in Fortran to ascertain its usefulness. Indeed, an entire library of utility routines for image processing was developed to investigate algorithms. These Fortran programs were executed on a conventional, serial processor, with no attempt to optimize execution speed. These subroutines are extensive and complex, reflecting a need to utilize algorithms which are difficult to express in Fortran. Rather than present a Fortran listing, which would obscure the situation, it has been decided to present only the most relevant algorithms, stripped, insofar as possible, of the numerous details necessary in a Fortran language implementation.

Also, in recognition of the fact that this problem in image processing cannot execute in real time on a serial computer, a parallel computer will be postulated. This processor will be briefly described. Extensive description is beyond the scope of this research.

Parallel processor

In this processor, each pixel in the square matrix representing the digital image has associated with it a general-purpose processor which executes the algorithms to be presented below. Each pixel processor has its own memory for storage of intermediate results. By definition, each
processor can access not only the gray value of its associated pixel, but also values of pixels surrounding it, arranged in concentric rings. It should be noted that for many algorithms buffer images will be required. Thus, the image matrix is considered to have sufficient levels to allow implementation of any of the algorithms to be presented here. In effect, a three-dimensional image stack is assumed.

**Language**

It has been stated that a general-purpose processor is associated with each pixel. For purposes of clarity, it will be assumed that each processor executes commands in a high level language, such as PASCAL, with the usual program structures. However, subroutines will be called by a two word name, with the words separated by an underline. No explicit call is required. Programming details not presented here are assumed to be implemented in system software.

Since the algorithms to be presented are based on the idea of a ring or annulus of elements surrounding a pixel, a number of subroutines will be defined to acquire gray values from a ring and insert them in conventional one-dimensional arrays or vectors. Similar conventions will be followed for writing arrays into rings. By manipulating arrays, conventional programming practice can be used, enhancing understanding of the algorithms while suppressing extraneous detail. Before discussion of the algorithms can begin, it will be
necessary to define a number of function subroutines. It is assumed that A is an array of size N with unity origin index.

**Function subroutines**

- **SUM(A)** returns the sum of the elements of the array A
- **AVG(A)** returns the average of the elements of the array A
- **STD(A)** returns the sample standard deviation of the array A
- **MED(A)** returns the median value of array A
- **MAX(J,K)** returns the larger in value of the positive scalars J and K
- **ABS(J)** returns the absolute value of the scalar J
- **SUCC(A,J)** for the Jth element of ring A, returns the relative direction of the succeeding element directions are defined by the even index values in Figure 4.9
- **PRED(A,J)** for the Jth element of ring A, returns the relative direction of the preceding element directions are defined by the even index values in Figure 4.9

The standard indexing scheme for ring elements is shown in Figure 4.9.

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

**Figure 4.9. Ring index values**
Also necessary are standard scalar comparison operators.

**Comparison operators**

- **.LT.** less than
- **.LE.** less than or equal to
- **.EQ.** equal to
- **.GT.** greater than
- **.GE.** greater than or equal to

Now, it is necessary to describe the use of a number of subroutines, whose basic purpose is to acquire gray levels from a ring surrounding the center pixel, and place these pixels in a conventional array to facilitate manipulation. Although in general, the radius of a ring may be any valid value, for this discussion, the radius will be assumed equal to one, and the ring will be approximated by the eight nearest neighbors. Thus, the radius will be presumed known globally, and will also appear explicitly as a variable named circ with value equal to eight.

**Subroutines**

READ ELEMENT(label, imagelabel, dmn) reads the gray value of the center pixel and assigns it the name label. If dmn is greater than one, then a linear array is read. The image from which the value is read is called imagelabel.

WRITE ELEMENT(label, imagelabel, dmn) writes a scalar named label into the pixel of a matrix called imagelabel. If dmn is greater than one, a linear array is written.

READ RING(label, imagelabel, dmn) beginning with the upper left pixel and proceeding clockwise, this subroutine places successive elements of the ring into a linear array called label of length equal to ring circumference. Since the ring radius is assumed one, the array size is eight. If dmn is
greater than one, label becomes a two-dimensional array of size \( dmn \) by eight. In effect, a cylindrical set of pixel values are read from a three-dimensional image matrix.

\[ \text{WRITE_RING}(\text{label}, \text{imgelabel}, \text{dmn}) \] beginning with the first element of the array called label, the elements are written into a ring in matrix imgelabel, starting with the upper left pixel and proceeding clockwise. If \( dmn \) is greater than one, a two-dimensional array is written. After a write operation, any previous information is destroyed.

\[ \text{SEGMENT_RING}(\text{label}, \text{Aname}, \text{Bname}, \text{AL}, \text{BL}) \] accepts a ring called label and segments the ring into two segments. Beginning with the upper left pixel, and proceeding clockwise, AL elements are placed into the array called Aname. Continuing in the clockwise direction, the remaining elements are placed in array Bname of length BL. The sum of AL and BL must equal the circumference of the ring. After this operation, the ring is still addressable.

\[ \text{ROTATE_RING}(\text{name}, \text{Aname}, \text{Bname}) \] accepts a previously segmented ring and rotates the A and B segments clockwise by one pixel. The rotation does not affect the pixels in the ring itself, but does update the arrays Aname and Bname.

\[ \text{SAVE_RING}(\text{name}, \text{Aname}, \text{Bname}) \] saves the values of elements in the segmented ring called name. It also saves the arrays Aname and Bname. The current values are not disturbed.

\[ \text{RESTORE_RING}(\text{name}, \text{Aname}, \text{Bname}) \] restores the saved values of the segmented ring called name and also restores the values of the arrays Aname and Bname. Any previous values are lost.

\[ \text{EXCHANGE_AB}(\text{name}, \text{Aname}, \text{Bname}, \text{AL}, \text{BL}) \] exchanges labels of the segmented ring called name. It also exchanges array names and lengths.

\[ \text{INDEX_RING}(\text{name}, \text{Aname}, \text{Bname}, \text{AX}, \text{BX}) \] accepts a segmented ring called name and returns two new arrays called AX and BX. These arrays contain the index values corresponding to the current positions of the A and B segments, relative to the upper left pixel, which has an index value of one, as shown in Figure 4.9.

\[ \text{EDGE_ARRAY}(\text{A}) \] accepts an array A and sets the first and last elements equal to one. All other elements are set to zero.

This concludes the enumeration of subroutines necessary to present the algorithms. The operator to be presented is
capable of edge detection, noise smoothing, and edge point linking. The algorithms will be presented in this order.

Since the algorithms are implemented in parallel, multiple step algorithms must ensure that one state of processing is completed before proceeding to the next. To represent this synchronization, the notation parallel:begin is used to denote the beginning of a parallel step for all pixel processors. Similarly, parallel:end is used to indicate the end of a parallel step, after which further processing may take place.

It must also be noted that the ring segmentation operation will divide the ring into two segments of variable size. For this discussion, the ring will be split into segments of size three and five. In general, the minimum size object capable of being detected is determined by the size of the smaller segment. Isolated objects whose maximum dimension is less than the minimum segment size will not be detected. The edge detection algorithm will now be presented.

**Edge detection algorithm**

parallel:begin
al = 3
bl = 5
circ = 8
read_ring(t,input,1)
\[\text{minsd} = \text{std}(t)\]
segment_ring(t,a,b,al,bl)
for count = 1 to circ
begin
if (max(std(a),std(b)).lt.minsd) then
begin
edgeflag = 1
minsd = max(std(a),std(b))
end
\[
\begin{align*}
\text{aav} &= \text{avg}(a) \\
\text{bav} &= \text{avg}(b) \\
\text{save_ring}(t,a,b) \\
\text{end} \\
\text{rotate_ring}(t,a,b) \\
\text{end} \\
\text{if } (\text{edgeflag} \cdot \text{eq.} 1) \text{ then} \\
\text{begin} \\
\text{restore_ring}(t,a,b) \\
\text{evalu} &= \text{abs}(\text{aav} - \text{bav}) \\
\text{if } (\text{bav} \cdot \text{gt.} \text{aav}) \text{ then} \text{exchange_ab}(t,a,b,a1,b1) \\
\text{edge_array}(a) \\
\text{for } \text{cnt} = 1 \text{ to } a1 \quad a(\text{cnt}) = a(\text{cnt}) \cdot \text{evalu} \\
\text{for } \text{cnt} = 1 \text{ to } b1 \quad b(\text{cnt}) = 0 \\
\text{read_ring}(u,\text{edgevl},1) \\
\text{for } \text{cnt} = 1 \text{ to } \text{circ} \quad u(\text{cnt}) = u(\text{cnt}) + t(\text{cnt}) \\
\text{write_ring}(u,\text{edgevl},1) \\
\text{read_ring}(v,\text{ecount},1) \\
\text{for } \text{cnt} = 1 \text{ to } \text{circ} \quad v(\text{cnt}) = v(\text{cnt}) + t(\text{cnt})/\text{evalu} \\
\text{write_ring}(v,\text{ecount},1) \\
\text{parallel:begin} \\
\text{parallel:begin} \\
\text{read_element}(x,\text{edgevl},1) \\
\text{read_element}(y,\text{ecount},1) \\
z = \frac{x}{y} \\
\text{write_element}(z,\text{result},1) \\
\text{end} \\
\text{parallel:begin} \\
\text{parallel:begin} \\
\text{end} \\
\text{end} \\
\end{align*}
\]

This algorithm will accept as input a gray scale image named input. The first action of the algorithm is to read a ring, and compute its standard deviation as defined. Next, the ring is segmented into two sections named a and b. Then, for each of circ positions of the segments, the standard deviations of arrays a and b are computed. The inequality in the algorithm functions to find the best edge, and sets edgeflag equal to one. However, if no edge satisfying the inequality is found, no edge is reported. Thus, automatic local
thresholding is achieved.

If an edge is detected, it will be a locally best edge, as more than one edge is possible in any given ring. Once the edge is found, its orientation is saved, and its magnitude is estimated. Arbitrarily, it is assumed that the edge resides on the pixel with larger gray value. Thus, as shown in Figure 4.10, the edges would be indicated.

```
  3 3 3 5 5 5   0 0 0 2 0 0
  3 3 3 5 5 5   0 0 0 2 0 0
  3 3 3 5 5 5   0 0 0 2 0 0
  3 3 3 5 5 5   0 0 0 2 0 0
```

Figure 4.10. Algorithm edge definition.

After an edge has been estimated, its magnitude and coordinates are added to a buffer field named edgevl. Also, a counter field named ecount is incremented. Each of these operations is fully parallel.

As a final step, the contents of edgevl are divided by ecount to produce an average edge image named result. It is believed that the magnitude measure used here is intuitively reasonable. Next, the smoothing algorithm will be represented. This algorithm will attempt to reduce noise while preserving object borders.
Smoothing algorithm

parallel:begin
al = 3
bl = 5
circ = 8
read ring(t,edge,1)
minsd = std(t)
segment ring(t,a,b,al,bl)
for count = 1 to circ
  begin
    if (max(std(a),std(b)).lt.minsd) then
      begin
        edgeflag = 1
        minsd = max(std(a),std(b))
        aav = avg(a)
        bav = avg(b)
        save ring(t,a,b)
      end
      rotate ring(t,a,b)
    end
  if (edgeflag.eq.1) then
    begin
      restore ring(t,a,b)
      amedian = med(a)
      for cnt = 1 to al  a(cnt) = amedian
      bmedian = med(b)
      for cnt = 1 to bl  b(cnt) = bmedian
    end
  else
    begin
      tmedian = med(t)
      for cnt = 1 to circ  t(cnt) = tmedian
    end
  end
read ring(u,smooth,8)
for cnt = 1 to circ  u(cnt,cnt) = t(cnt)
write ring(u,smooth,8)
parallel:end

parallel:begin
read element(x,smooth,8)
y = med(x)
write element(y,result,1)
parallel:end

This algorithm will accept as input an image named edge
and produce a smoothed output image named result.
As has been noted in a previous chapter, all noise reduction techniques should ideally eliminate noise while preserving edges. This algorithm attempts to do that by using the fundamental edge detector to find edges. Once an edge is located, this algorithm performs a smoothing operation on each segment of the ring. As demonstrated previously, the median smoother is a useful nonlinear technique, and is employed here to eliminate blur. Therefore, after an edge has been found, each element of a segment is replaced by the median of that segment. Thus, when this algorithm is completed, each element of a has the same value, and likewise for b. Then, these values are written into an eight level image matrix named smooth. It is noted that this procedure results in circ estimates of the smoothed value of each pixel. When this parallel process has terminated, the eight level matrix is filled. At this point, the median technique is used to decide on a single estimate for each pixel, by taking the median of the eight estimates for each point.

In the event that no edge is detected, the median of the entire ring is stored for all elements of the ring. Thus, by using a median of medians, noise is reduced. The performance of this algorithm will be discussed in the following chapter. It is evident that this algorithm can be applied more than once, although only single pass results will be presented.

The next algorithm to be discussed will be the edge
point linker.

**Edge point linker**

```
parallel: begin
al = 3
bl = 5
circ = 8
read ring(t,edge,1)
segment ring(t,a,b,al,bl)
for count = 1 to circ
  begin
    if (max(std(a),std(b)).lt.minsd) then
      begin
        edgeflag = 1
        minsd = max(std(a),std(b))
        aav = avg(a)
        bav = avg(b)
        save ring(t,a,b)
      end
    rotate ring(t,a,b)
  end
if (edgeflag.eq.l) then
  begin
    restore ring(t,a,b)
    if (bav.gt.aav) then exchange ab(t,a,b,al,bl)
    index ring(t,a,b,ax,bx)
    do cnt = 2 to al-1
    begin
      u(ax(cnt),succ(ax,cnt)) = u(ax(cnt),succ(ax,cnt)) +
      sum(a)
      u(ax(cnt),pred(ax,cnt)) = u(ax(cnt),pred(ax,cnt)) +
      sum(a)
    end
    u(ax(l),succ(ax,l)) = u(ax(l),succ(ax,l)) + sum(a)
    u(ax(al),pred(ax,al)) = u(ax(al),pred(ax,al)) + sum(a)
    write ring(u,linkmx,8)
  end
parallel:end
```

This algorithm accepts an edge point representation from a field called edge and attempts to perform local linking of edge points into longer chains. In general, edges of any shape may be linked. However, in this implementation, diagonal linking was not included. Thus, only vertical and
horizontal linking will be performed.

As stated, this algorithm applies the basic edge detector to determine an optimal local edge. If such an edge is detected, each pixel of the edge will generate links to its neighbors within the segment. Interior points will each generate two links, whereas end points of the segment will each form one link. The links are represented in the following manner. In general, each point can link in eight directions. Therefore, an eight level link matrix is utilized, resulting in an eight element array for each pixel, corresponding to the possible link directions. The magnitude of the link is equal to the sum of the gray values of the edge segment. Thus, to generate a link in a particular direction, it is necessary to add the edge sum to the array element corresponding to the particular direction. When completed, the output of the linker will be an eight level matrix. Higher level processing may then proceed.

In the following chapter, the performance of the algorithms discussed above will be examined.
PERFORMANCE TESTING

In the previous chapter, a new edge detection method was used in three algorithms capable of noise reduction, edge detection, and edge point linking. The purpose of this chapter is to evaluate the performance of the three algorithms which together comprise the new enhancement operator.

Since a rigorous mathematical analysis did not prove feasible, it was decided to characterize the performance of this operator by applying it to a series of test images and displaying the results after each algorithm has been applied. Also presented will be a quantitative simulation resulting in curves which indicate the ability of the edge detector to function in the presence of noise. With this introduction, the results may be presented.

Although it may be very difficult to devise reliable and effective image processing algorithms, it is perhaps equally difficult to measure the actual effectiveness of such algorithms. The basis for this problem seems to arise from the fact that the human eye-brain visual processor is not well-understood, although research in this area is ongoing. At present, the eye-brain system is essentially unrivalled in its ability to enhance images and recognize patterns. Thus, the measurement of performance of image processing algorithms is problematic.

In the face of such a dilemma, there are several ways
to proceed. For example, one may use techniques and algorithms which can be studied analytically. The algorithm proposed in the previous chapter was created to deal directly with the general problem of enhancement or low level processing of digital images with significant noise levels, such as those encountered in phased-array B-scan systems. At present, a rigorous analytical technique to predict performance of this algorithm has not been found, although it has been sought for. Rather than discard the algorithm because it cannot be rigorously analyzed, two other methods for evaluating its performance are available.

First, the algorithm may be applied to images and the results presented for human inspection. Indeed, it is customary in the image processing literature to illustrate the performance of the algorithm on a standard test picture or series of pictures. This qualitative evaluation seems especially appropriate when the intent of the algorithm is to enhance an image to improve its quality for a human observer. Therefore, such a series of test images will be presented, followed by a quantitative simulation testing the performance of the basic edge detector.

Due to a lack of image processing facilities, it became apparent that actual ultrasound images could not be processed. Therefore, the set of enhancement algorithms has been applied to a series of simulated images, whose gray scale repre-
sentations were shown in the previous chapter. These gray level pictures were generated from the numerical data produced by the Fortran implemented algorithms of the previous chapter. The method used to display the gray level images was developed by Hager (27). In this chapter, the actual numerical data will be presented. Details of the simulation generating these data will now be discussed.

**Image simulation**

In a phased-array B-scan display system, the CRT generally displays an image of size 128 by 128, with ten to sixteen gray levels. For this simulation, an image size of twenty by twenty pixels was chosen, with ten levels of gray, each represented by an integer from zero to nine. By specifying an integer for each of the 400 pixels, any gray scale image can be created. While the small size of this simulated image will produce a crude appearance, it is sufficient to demonstrate the algorithm. When an image is printed, zeroes will appear as blanks, in order to increase legibility of the picture.

**Noise**

Since the purpose of this simulation is to show the efficacy of the low level operator in the presence of noise, the method of simulation of noise is an important topic. To provide the most accurate simulation, it would be desirable to generate noise with statistics typically measured in phased-array systems. Unfortunately, noise models for ultrasonic
imaging systems have not yet emerged.

In view of this uncertainty, it was decided to use pseudo-random noise with a gaussian distribution. The normal distribution was chosen not only because it is a standard scientific subroutine function, but because it is commonly found in the literature. Also, the variance provides a readily adjustable parameter controlling the severity of the noise.

To contaminate an image with noise of a given standard deviation, the following procedure is followed. Each pixel is replaced by a value drawn from a gaussian distribution with mean equal to the value of the pixel being replaced, and standard deviation preselected and constant for the entire image. Thus, a new gray level is generated and replaces the original value. If the number generated is less than the minimum or greater than the maximum, it is clipped to minimum or maximum, respectively.

For this simulation, integer standard deviations of one, two, and three were chosen. It was determined empirically that noise of standard deviation three is severe.

In order to characterize the normality of the random number generator, one million random integers were generated for each standard deviation. For each integer generated, its absolute difference from the mean was computed and tabulated. Figure 5.1 shows the relative frequencies for standard deviations of one, two, and three. It should be noted that
Figure 5.1. Discrete gaussian distribution.
this noise is uncorrelated. That is, the values of surrounding pixels have no effect on the generation of a new gray value.

Now that procedures for generating images and noise have been discussed, it is necessary to explain additional processing needed to facilitate display of the results.

As noted in the previous chapter, a noisy image will undergo enhancement in three steps, consisting of noise reduction, edge point detection, and edge point linking. Since the output of the linker is a three-dimensional matrix, it seemed desirable to perform additional processing for the benefit of a human observer. Thus, the following high level procedure has been adapted to convert the three-dimensional matrix into a more easily viewable format.

Ideally, each edge point will have two edge points linked to it. To indicate the existence and direction of these two links, the notation shown in Figure 5.2 has been developed to represent the four rectangular directions possible for links.

```
 2
 8 ___ 4
 6
```

Figure 5.2. Link directions.

Thus, a line of horizontal edge points would each contain a 48 code after conversion from a three-dimensional format.
Vertical edge points would consist of 26 codes. Although links in eight directions are possible, only vertical and horizontal linking capability was implemented here. Thus, the missing odd numbers correspond to diagonal links. In the event that an edge point is also an end point of a chain, it is denoted by a 77. If an edge point has more than two links, it will be denoted by a 99. By using this method, the results of the edge point linker can be compressed into a single image. In some cases, where more than two links were possible, chains yielding the largest gray value were preferred. Thus, small branches were pruned away from higher gray level chains.

As a further aid to visualization, it seemed useful to label each chain with a unique integer, scanning from left to right and bottom to top. When a chain link is encountered, a standard graph traversal algorithm is invoked, which employs a stack to track and label chains. Thus, when this step is completed, each chain in the image is assigned an integer. The largest integer represents the total number of chains found in the image.

The format for presentation of the simulated images follows. First, the noise-free image will be presented, followed in succession by the same object contaminated with varying levels of noise. Each set of images will consist of input, smoothed image, edge detected image, linked image, and labeled image. Examples given include a square of gray level
seven against a background of level three, as shown in Figures 5.3 to 5.22. Also included is an abstract representation of a long-axis view of the left ventricle, shown in Figures 5.23 to 5.37. The large object at the bottom of the picture represents the posterior wall of the left ventricle, while the uppermost object represents the interventricular septum. The remaining object represents the anterior leaflet of the mitral valve. The next test image is a single object with several areas of differing gray level, as shown in Figures 5.38 to 5.52.

Simulation

In the previous section, a qualitative demonstration of enhancement was presented. An alternative technique is to perform a simulation which measures quantitatively the ability of the algorithm to function in the presence of noise. While it is unclear how noise-smoothing or edge linking should be assessed, it is possible to quantify the capability of the basic edge detector to find edges in the presence of noise.

To determine this capability, a ring consisting of eight nearest neighbors was formed. Beginning with the upper left pixel, four consecutive pixels proceeding clockwise, were set at gray level \( x+y \). The remaining four pixels were set to level \( x \), thus forming an edge with magnitude \( y \). After contaminating this ideal edge with noise, the edge detector is applied to determine if an edge of the initial size and
Figure 5.3. Input square, noise s.d. = 0.
Maximum brightness = 9.
Minimum brightness = 0.
Figure 5.4. Smoothed square, s.d. = 0
Figure 5.5. Edge square, s.d. = 0
Figure 5.6. Linked square, s.d. = 0.
Figure 5.7. Labeled square, s.d. = 0.
Figure 5.8. Input square, s.d. = 1.
Figure 5.9. Smoothed square, s.d. = 1.
**Figure 5.10.** Edge square, s.d. = 1.
Figure 5.11. Linked square, s.d. = 1.
Figure 5.12. Labeled square, s.d. = 1.
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.13. Input square, s.d. = 2.
Figure 5.14. Smoothed square, s.d. = 2.
Figure 5.15. Edge square, s.d. = 2.
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 46 | 68 | 77 | 46 | 77 | 77 | 77 |   |   |   |   |   |   |   |   |   |   |   |
| 46 | 48 | 48 | 77 | 24 | 28 | 77 | 77 | 46 | 68 | 77 | 46 | 68 |   |   |   |   |   |
| 26 | 77 | 77 | 77 | 46 | 58 | 24 | 28 | 77 | 24 | 28 |   |   |   |   |   |   |   |
| 26 | 77 | 26 | 77 | 26 | 77 | 77 | 77 | 77 | 77 |   |   |   |   |   |   |   |   |
| 24 | 48 | 48 | 77 | 26 | 77 |   | 24 | 28 |   |   |   |   |   |   |   |   |   |
| 77 | 46 | 48 | 48 | 48 | 48 | 68 |   |   |   |   |   |   |   |   |   |   |   |
| 26 | 77 | 77 | 24 | 48 | 68 |   |   |   |   |   |   |   |   |   |   |   |   |
| 26 | 46 | 48 | 77 | 77 | 77 | 26 |   |   |   |   |   |   |   |   |   |   |   |
| 77 | 68 | 77 | 68 | 77 | 26 | 26 | 24 | 28 | 77 | 46 | 48 | 77 |   |   |   |   |
| 77 | 77 | 77 | 77 | 26 |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 77 | 77 | 77 | 77 | 77 | 77 | 77 |   |   |   |   |   |   |   |   |   |   |   |
| 46 | 68 | 77 | 77 | 24 | 28 | 24 | 77 | 77 | 77 |   |   |   |   |   |   |   |   |
| 26 | 77 | 77 | 28 |   | 77 | 77 | 46 | 48 | 48 | 48 | 77 |   |   |   |   |   |   |
| 24 | 68 |   | 77 | 28 | 77 | 26 |   |   |   |   |   |   |   |   |   |   |   |
| 77 | 28 |   |   | 46 | 48 | 48 | 28 |   |   |   |   |   |   |   |   |   |   |
| 77 | 77 | 24 | 77 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Figure 5.16. Linked square, s.d. = 2.
Figure 5.17. Labeled square, s.d. = 2.
Figure 5.18. Input square, s.d. = 3.
Figure 5.19. Smoothed square, s.d. = 3.
Figure 5.20. Edge square, s.d. = 3.
Figure 5.21. Linked square, s.d. = 3.
Figure 5.22. Labeled square, s.d. = 3.
Figure 5.23. Simulated long axis B-scan input, s.d. = 0.
Figure 5.24. Smoothed B-scan, s.d. = 0.
Figure 5.25. Edge B-scan, s.d. = 0,
```
|  46 48 48 48 48 48 48 48 48 68 |
|  26       77 26               |
|  26       24 99               |
| 24 48 48 48 48 48 48 77 77     |
|  46 48 48 48 48 48 68          |
|  26       26                   |
|  26  46 99 48 28               |
| 24 48 99 28                     |
|  77 48 68                        |
|  46 48 28 77                     |
|  26  77 99                       |
| 99 68  26                         |
| 24 99  24 48 48 48 48 48 48 28   |
|  26                    46 99 28    |
| 24 48 99 68                     |
|  46 99 99 28                     |
| 24 99 48 48 48 48 48 99 28      |
|                                |

Figure 5.26. Linked B-scan, s.d. = 0.
```
Figure 5.27. Labeled B-scan, s.d. = 0.
Figure 5.28. Simulated long-axis B-scan input, s.d. = 1.
Figure 5.29. Smoothed B-scan, s.d. = 1.
Figure 5.30. Edge B-scan, s.d. = 1.
Figure 5.31. Linked B-scan, s.d. = 1.
Figure 5.32. Labeled B-scan, s.d. = 1.
Figure 5.33. Simulated long-axis B-scan input, s.d. = 2.
Figure 5.34. Smoothed B-scan, s.d. = 2.
Figure 5.35. Edge B-scan, s.d. = 2.
Figure 5.36. Linked B-scan, s.d. = 2.
Figure 5.37. Labeled B-scan, s.d. = 2.
Figure 5.38. Multiple gray level object input, s.d. = 0.
Figure 5.39. Smoothed object, s.d. = 0.
Figure 5.41. Linked object, s.d. = 0.
Figure 5.42. Labeled object, s.d. = 0.
Figure 5.43. Multiple gray level object input, s.d. = 1.
Figure 5.44. Smoothed object, s.d. = 1.
Figure 5.45. Edge object, s.d. = 1.
Figure 5.46. Linked object, s.d. = 1.
Figure 5.47. Labeled object, s.d. = 1.
Figure 5.48. Multiple gray level object input, s.d. = 2.
Figure 5.49. Smoothed object, s.d. = 2.
Figure 5.50. Edge object, s.d. = 2.
Figure 5.51. Linked object, s.d. = 2.
Figure 5.52. Labeled object, s.d. = 2.
orientation can be found. By repeating this procedure with noise of varying levels, and edges of many magnitudes, a graph can be constructed, showing frequency of edge detected as a function of edge magnitude for three standard deviations. This graph is shown in Figure 5.53.

The results obtained here for both qualitative and quantitative tests will be discussed in the next chapter.
Figure 5.53. Edge detector characteristic curves.
CONCLUSIONS

In the previous chapter, the new low level operator was applied to a series of simulated images under a variety of noise conditions to permit a qualitative evaluation. Also, the basic edge detector was subjected to a more quantitative test, allowing prediction of performance in the presence of gaussian noise.

To begin, it is noted that all objects were successfully processed when no noise was present in the qualitative simulations on the three test images. The square was properly removed from a background level unknown to the algorithm a priori. The simulated cardiac cross-section, shown as it would appear on a phased-array CRT, was also properly processed and outlined. It should be noted that diagonal lines would not be successfully linked, because such linking was not implemented. The multiple gray level object was also successfully bordered with only minor problems. From this test picture, it can be seen that low level algorithms must be able to deal with multiple gray levels in an image. Clearly, no single global threshold would produce all the information available in this image, such as moderate magnitude edges within the object. This algorithm appears to function well because it effectively uses local thresholds to detect local edges. Also, because the ring size can be minimal, objects of dimension equal to
ring diameter may be detected, presuming that noise is not excessive.

It is also noted that the operator is in no way dependent on the shapes or relative orientation of objects or borders. It is not dependent on the value of background level or any particular gray scale assignment.

In the noisy images, it is seem that the algorithm performs fairly well. The noise smoother appears to be rather effective, although it is only applied once. The square for standard deviation one is found readily, and the same is true for the other objects. However, when objects are close together relative to the diameter of the ring, noise points may inevitably cause merging. It can be concluded that for ten gray levels, gaussian noise of standard deviation one is very moderate, and may be sucessfully removed with this set of algorithms.

For noise of standard deviation equal to two, the noise is visibly more severe, with noticeably higher amplitudes. With noise of this level, the square can be partially distinguished, although deterioration is noticeable. The cardiac cross-section, being a more crowded picture, suffers because objects are merged, as does the multiple gray level object. It is believed that this degradation is due in part to the fact that various edge features approach the diameter of the ring. Noise immunity will be improved as object minimum
feature size increases. In an actual B-scan image, it is expected that objects would be represented by fairly large numbers of pixels, thus reducing resolution problems. Gaussian noise of standard deviation two may be considered as moderate to severe. The low level operator still functions to find edges, but with increasing error.

Each of the test pictures was subjected to noise of standard deviation equal to three. Only results for the square are shown. The distortion in the cardiac cross-section and multiple gray level object was severe, merging regions and drastically reducing the information content of the picture. For the square, it can be seen that it is badly distorted, although segments remain. Clearly, if noise of this magnitude is encountered in an imaging system, efforts should be directed towards improvement in the imaging system itself, rather than relying on image processing. Noise of standard deviation three may be considered as severe, disrupting not only true edges, but also creating numerous spurious edges and objects.

It should be noted that noise, by definition, produces uncertainty in a signal or an image. Thus, if noise is sufficiently severe, the image may be degraded irretrievably, and no low level techniques exist which can restore it. Therefore, algorithms such as the ones presented above will be most useful operating on images with slight to moderate noise.
Clearly, it would be very desirable to apply these algorithms to actual phased-array B-scan images. Also, it would be desirable to test the efficacy of the algorithms on other types of medical, scientific and industrial images. In this manner, the generality of this operator could be more readily determined.

Next, the results of the quantitative simulation will be discussed.

The curves represented in Figure 5.53 actually are characteristic curves, which predict the probability that an edge will be detected for a given edge amplitude and noise level. The results are expressed as probabilities because an edge in noise is a probabilistic phenomena. For a very long, straight edge contaminated by noise, the percentage of edge points detected will be very close to the value predicted by the characteristic curve. However, for objects with curvatures on the order of ring diameter, the effect of noise will be increased. The algorithm may be expected to give best results on objects of curvature less than ring diameter.

In general, the simulation results seem to agree with intuition. That is, the larger the edge magnitude, the greater the immunity to noise. A low magnitude edge will be degraded by low amplitude noise, whereas a high magnitude edge will not. Also, as noise increases, the probability of detecting an edge of a given magnitude decreases. Analysis of
this edge detector is complicated by the fact that variable segment sizes may be used, seeking edges of any orientation around each pixel. Also, the radius need not be restricted to a value of one.

In addition to the fact that these algorithms could not be rigorously analyzed, it is also noted that they are extremely complex, requiring multiple processors and extensive buffer memories to achieve real-time operation. However, in view of the enormous strides being made in LSI and VLSI technology, it would appear that cost-effective parallel processors and three-dimensional image memories will become available in the foreseeable future. Thus, the algorithms presented here have been designed not with respect to current technology but rather with respect to technology that may reasonably be expected to become available in the future.

**Summary**

A general-purpose low level image enhancement operator has been presented. It has been applied to simulated images to permit qualitative assessment of its qualities. The basic edge detector has also been studied via quantitative simulation, resulting in characteristic curves.

The algorithm developed has several significant features. First, the set of algorithms is fully parallel and non-iterative, operating on any gray scale image without human intervention or a priori information. The algorithms perform
the fundamental low level tasks of noise reduction, edge point detection, and edge point linking. This set of routines may be considered as a rather general image preprocessor, whose output may be utilized by any high level algorithm or pattern recognition system based on edge representations.

It is also believed that this algorithm provides a useful conceptual model of edges in digital images, and as such may be a contribution to the theory of digital image processing. The utility of this edge model as a concept is reinforced by its capabilities in enhancement of simulated images.
REFERENCES


ACKNOWLEDGMENTS

First of all, I would like to thank my major professor, Dr. David L. Carlson. His constant encouragement, enthusiasm, and personal kindness are deeply appreciated and gratefully acknowledged.

Next, I wish to thank Dr. A. V. Pohm for serving as co-major professor and for his encouragement.

I would also like to acknowledge Dr. H. C. Brearley and Dr. N. R. Cholvin for serving on my committee and for providing additional research experience for me.

Finally, I wish to thank Dr. R. C. Seagrave, who provided both facilities and computational resources for this research, in addition to serving on my committee.

Last of all, I wish to thank my parents for their unfailing encouragement and support.