Signal Processing in a Real-Time
Three-Dimensional Acoustic Imaging System

by

Daniel Frederick Lohmeyer

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 4, 1998

© 1998 Daniel F Lohmeyer. All Rights Reserved

The author hereby grants to M.I.T. permission to reproduce and distribute publicly paper and electronic copies of this thesis and to grant others the right to do so.

Author ____________________________

Department of Electrical Engineering and Computer Science

February 4, 1998

Certified by ________________________

George Verghese

Thesis Supervisor

Accepted by ________________________

Arthur C. Smith

Chairman, Department Committee on Graduate Theses
Signal Processing in a Real-Time
Three-Dimensional Acoustic Imaging System

by

Daniel Frederick Lohmeyer

Submitted to the
Department of Electrical Engineering and Computer Science

February 4, 1998

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Lockheed Martin is currently developing a real-time three-dimensional acoustic imaging system, consisting of a portable device capable of generating volumetric images from ultrasonic signals reflected off target objects. This thesis first presents a functional description of the overall system from a signal- and image-processing perspective. Second, it analyzes the acoustic signal path to measure a frequency response which is then used to determine optimal transmission pulse characteristics. Third, this research identifies the problem of defective array elements and presents multiple approaches to correct for these flawed pixels. Finally, it explores several object recognition metrics and evaluates them on a testbed of synthetic data.

Thesis Supervisor: George Verghese
Title: Professor of Electrical Engineering, Massachusetts Institute of Technology
Acknowledgements

I would first like to thank the acoustic imaging team at Lockheed Martin for sponsoring my work. In particular, I would like to recognize Anthony Nicoli for his excellent leadership, for acquiring my research funding, and for continuously guiding the focus of my thesis as it evolved. I am grateful to Jason Stockwell for always prioritizing my thesis when other tasks arose and for providing the critical technical knowledge to make everything work. Last but not least, I thank Ken Erikson for his in-depth understanding of the system and his readiness to explain difficult aspects of it to me.

I am greatly indebted to Professor George Verghese, my M.I.T. thesis advisor, for his tremendous support in all facets of this research.

I sincerely appreciate the efforts of Tim White, Robert Schneider, Dale von Haase, Paul McQuillan, Susan Guralnik, and Markus Zahn in helping me secure funding for this work.

To my parents and family, I am forever indebted. Thank you for providing me with the opportunity to attend M.I.T. and supporting me every step of the way.

Thanks to my brother Mark for letting me borrow his car so I could get to work every day. I especially thank him for having a car that stops working whenever it rains, so I could stay home and focus entirely on my thesis those days.

Finally, I thank Rima Woo and Andrew Huang for helping me get this document printed and submitted by 5 p.m. on the due date.
Table of Contents

1 Introduction ........................................................................................................ 13
   1.1 Acoustic Imaging ......................................................................................... 13
   1.2 Thesis Overview ......................................................................................... 15

2 Hardware Description .......................................................................................... 19

3 Pulse Repetition Frequency .............................................................................. 23
   3.1 Introduction ................................................................................................. 23
   3.2 Problem Statement ...................................................................................... 23
   3.3 Linear System Range .................................................................................. 24
   3.4 Transmission Signal .................................................................................... 28
   3.5 System Characterization ............................................................................. 30
       3.5.1 Introduction .......................................................................................... 30
       3.5.2 Receive Transducer Array .................................................................... 31
       3.5.3 Target Distance Effects ....................................................................... 32
   3.6 Waveform Analysis ...................................................................................... 34
   3.7 Conclusion .................................................................................................. 35

4 Transmission Signal Duration .......................................................................... 37
   4.1 Introduction ................................................................................................. 37
   4.2 Problem Statement ...................................................................................... 38
   4.3 Optimal Signal Criterion ............................................................................ 39
   4.4 Ideal Boxcar Solution .................................................................................. 41
   4.5 Actual Signal Solution ................................................................................ 42
   4.6 Conclusion .................................................................................................. 43

5 Defective Pixel Compensation ............................................................................ 45
   5.1 Overview .................................................................................................... 45
       5.1.1 Previous Work ...................................................................................... 45
       5.1.2 Problem Statement .............................................................................. 46
   5.2 Defective Element Identification ................................................................ 46
       5.2.1 Introduction .......................................................................................... 46
       5.2.2 Causes of Pixel Failures ....................................................................... 47
       5.2.3 Defective Pixel Determination ............................................................... 48
       5.2.4 Summary .............................................................................................. 51
   5.3 Tracking Flawed Elements .......................................................................... 52
       5.3.1 Introduction .......................................................................................... 52
       5.3.2 Data Structures .................................................................................... 52
       5.3.3 Comparison between Structures ............................................................ 55
       5.3.4 Summary .............................................................................................. 57
   5.4 Failed Pixel Compensation ......................................................................... 58
       5.4.1 Introduction .......................................................................................... 58
       5.4.2 Last Valid Sample ................................................................................ 59
       5.4.3 Nearest Neighbor ................................................................................ 60
       5.4.4 Average Filtering ................................................................................. 60
       5.4.5 Median Filtering .................................................................................. 61
       5.4.6 MinMax Filtering ................................................................................ 62
List of Figures

Figure 1: Functional System Diagram ................................................................. 14
Figure 2: System Electronics Diagram ............................................................... 20
Figure 3: Frequency Dependent Water Attenuation ........................................... 25
Figure 4: Transducer Hybrid Assembly Linearity Measurement ........................... 27
Figure 5: Frequency Spectrum of Several Pulses ............................................... 29
Figure 6: Acoustic Pulse ..................................................................................... 29
Figure 7: System Block Diagram ....................................................................... 30
Figure 8: Detector Array Frequency Response .................................................. 32
Figure 9: Extended System Transfer Function .................................................... 33
Figure 10: Extended System Block Diagram ....................................................... 33
Figure 11: Acoustic Plane Configuration .............................................................. 37
Figure 12: Acoustic Signal in Different Planes ..................................................... 38
Figure 13: Signal and Plane Spacing ................................................................. 39
Figure 14: Expected Utility of a Boxcar Pulse ..................................................... 41
Figure 15: Optimal Receive Signal Envelope ...................................................... 42
Figure 16: Randomly Scattered Defective Pixels ................................................. 48
Figure 17: Pixel Intensity Histogram given Zero Insonification ............................. 49
Figure 18: Individual Pixel Responsivity ............................................................. 50
Figure 19: Boolean Image Map .......................................................................... 53
Figure 20: Defective Image and Corresponding Boolean Map .............................. 58
Figure 21: Image Corrected using Last Valid Sample Algorithm .......................... 59
Figure 22: Image Corrected using Nearest Neighbor Algorithm ........................... 60
Figure 23: Image Corrected using an Averaging Filter ........................................ 61
Figure 24: Image Corrected using a Median Filter .............................................. 62
Figure 25: Image Corrected using a MinMax Filter ............................................. 63
Figure 26: Image Recognition Testbed ............................................................... 74
Figure 27: Certainty Metric Error Evaluations ................................................. 75
Figure 28: Centering Algorithm Example ......................................................... 80
Figure 29: Scaling Algorithm Example .............................................................. 81
Figure 30: Angle Histogram ............................................................................. 81
Figure 31: Rotating Algorithm Example ............................................................ 82
Figure 32: Acoustic Image of Multiple Stacked Cylinders ................................... 86
Figure 33: Singular Values of an Acoustic Image .............................................. 87
Figure 34: Images Reconstructed using the SVD ............................................. 87
List of Tables

Table 1: Coordinate Table Example................................................................. 54
Table 2: Data Tracking Processor Requirements.............................................. 56
Table 3: Compensation Algorithm Performance Tradeoffs.............................. 64
Table 4: Verification Data of the Accuracy Equation....................................... 77
Table 5: Recognition Metric Processor Requirements...................................... 78
Table 6: SVD Computational Requirements.................................................... 89
Table 7: Recognition Metric Performance at Various Compression Levels.......... 92
1 Introduction

1.1 Acoustic Imaging

Underwater optical imaging is often difficult or even impossible in highly turbid waters such as those found in harbors and estuaries. Optical scattering from particulate matter in the 3 to 5 micron size range (6 to 10 optical wavelengths) is responsible for this poor visibility [1]. Acoustical imaging overcomes these limitations while retaining tactically significant image resolution. Ultrasound, with typical wavelengths of 0.5 mm ($10^3$ times longer than optical wavelengths), is virtually immune to this scattering. Only when particle sizes approach the ultrasound wavelength is there any significant dispersion. Suspended particle concentrations in the 100 micron size range are typically five orders of magnitude less common than those in the 5 micron range [1]. Particles of this size may be stirred up from the bottom by swimming or storm activity, so under some conditions they may become a factor even in acoustical imaging [2].

Conventional ultrasonic imaging systems use a single row of detector elements to scan a plane of the image, mechanically move the sensor, and then capture another plane. This results in cumbersome devices not suitable for portable imaging. Several such systems have been developed in the past 20 years [3,4,5,6], but due to their awkwardness, they are undesirable for many operations. Lockheed Martin is constructing an acoustical imager capable of three-dimensional (3D) imaging in real time, by using a two-dimensional detector array and range-gating the input with respect to time. This permits objects at different depths to be distinguished spatially and significantly enhances the
capabilities of acoustic imaging by allowing entire volumes to be quickly rendered with a portable device. A simplified picture of this system is presented in Figure 1 [7].

![Functional System Diagram](image)

**Figure 1**: Functional System Diagram

A wide variety of applications of this device for real-time 3D ultrasonic imaging exists in the medical, military, commercial, and scientific markets. Within the medical arena, physicians could have the capability to select internal body tissues for noninvasive 3D viewing by simply moving a detector probe. In conventional ultrasound systems, generating these images requires multiple scans over an immobile target body and then a timely computer algorithm to reconstruct the volumetric image. Lifting these restrictions would simplify current practices and open up new medical frontiers as well. Real-time ultrasonic imaging has already generated much interest for military underwater vision-enhancement systems. Equipped with this technology, divers and autonomous probes could extend their vision ten meters and still retain high (5 mm) resolution even in the
most turbulent or murky environments. Acoustic imaging also has several valuable commercial applications. For example, over-harvesting in the fishing industry has severely depleted many offshore species. If a remote probe could help a biologist study the breeding habits of deep-sea fish, the conditions under which they breed might be replicated in hatcheries. Acoustic imaging even has its place in the search for extraterrestrial life. Europa, one of Jupiter’s moons, is believed to be predominantly aquatic. If an autonomous robot could be deployed there, it could use ultrasound to search Europa’s oceans for traces of life.

1.2 Thesis Overview

The first two chapters in this thesis introduce the reader to acoustic imaging. Real-time three-dimensional ultrasonic imaging is a new technology and as a result, very few papers have been published on the topic. Thus, each chapter includes supplementary background information as necessary. However, in order to familiarize the reader with the hardware currently being used, Chapter 2 presents a high-level functional description of the Lockheed Martin system.

The body of this thesis is divided into three main chapters, 3 through 5. Chapter 3 examines the time-varying acoustic signal each detector element receives. It inspects the frequency response of various systems that affect the transmitted pulse, in order to derive a maximum Pulse Repetition Frequency (PRF). This research calculates the transfer function of the transducer array and characterizes the effects of water attenuation on imaging targets at different ranges. Finally, it determines a maximum PRF to ensure that
previous signals have been sufficiently attenuated and will not interfere with the current pulse.

Chapter 4 investigates several image processing algorithms to compensate for defective transducer arrays. Typical manufactured arrays experience 2-15% total element failure. These "dead" pixels return a static voltage irrespective of the incident ultrasonic energy, producing scattered pixels in the displayed image. If these flawed elements can be identified, various algorithms can then be deployed to conceal them. This chapter evaluates several strategies to locate, track, and correct for defective pixels, in order to produce a more comprehensible display.

Chapter 5 explores the topics of object detection and recognition in received acoustic imagery. The first section develops several elementary identification algorithms and evaluates them using synthetic data. In practice, a user will observe targets from a wide range of viewing perspectives. These differences in orientation must be reconciled before a recognition algorithm is applied. The second section develops functions to normalize an object’s location, size, and rotation. The third section explains the Singular Value Decomposition (SVD) theoretically and then utilizes it as a compression tool to conserve memory on the image processing board. The final section studies the effects of using compressed images on the recognition algorithm accuracy.

Chapter 6 summarizes the research performed in this thesis and draws several important conclusions. It also cites additional research areas for anyone interested in continuing these efforts.

The acoustic imaging system was not entirely functional at the time of this writing, so certain algorithms were analyzed using synthetic data. These procedures
should succeed in actual practice, because they were first evaluated with examples representative of real environments. However, particular techniques may excel in some applications, while other methods perform better when given different objectives. These considerations can be taken into account only after the system has been evaluated in the target environment. Hence, this research is tailored to be broadly applicable and open to modification whenever new information becomes available.
2 Hardware Description

The Lockheed Martin acoustic imager consists of several subsystems: an ultrasonic transmitter, acoustic lens, composite piezoelectric detector array, read-out integrated circuitry, acoustic imaging module, console board, digital image processing board, and a host computer system. The array, read-out circuitry, and imaging module reside in a small (8 cm diameter, 12 cm length) cylinder, which is deployed underwater. Long cables connect the imaging module to the console and image processing board, which reside in the host system out of the water. The host computer allows a user to configure and control all of the imaging subsystems. Figure 2 contains a diagram of the system electronics from the transducer array to the display screen [7].

An ultrasound transducer driven with a high voltage pulse emits a short (4 – 32 cycle) 3 MHz acoustic wave which propagates through water until it interacts with a target and may be reflected back toward the imager. This returned wave is focused through an acoustical lens onto a composite piezoelectric array. The transducer array contains a 64 by 64 pixel matrix optimized to receive a pulse modulated at 3 MHz, but may be tuned to other frequencies for different applications [8]. Each individual pixel cell measures 0.4 by 0.4 mm, which is less than the 3 MHz wavelength of 0.5 mm. The lens and transducer array can scan a 1x1x2 m volume underwater at a range of 5 m. The corresponding object resolution at such a range is approximately 2x2x1 cm, as a simple geometric ratio of target area to array size (100 cm / 64 ~ 2 cm) illustrates.
The focused image is incident on a Transducer Hybrid Assembly (THA), which contains a composite piezoelectric array and a Read Out Integrated Circuit (ROIC), which converts the transduced charge into a voltage. The ROIC amplifies this signal, samples it into discrete temporal values, and then exports it off the THA. The ROIC is capable of functioning in two modes, quadrature and sample & hold. Under quadrature detection...
mode, five planes$^1$ of data are collected per acoustic pulse. This data consists of four samples per pixel (+I, -I, +Q, -Q), ninety degrees out of phase, which may be used to reconstruct the continuous time wave [9]. In sample & hold mode, twenty samples are collected per acoustic pulse for each pixel. After acquisition in either mode, the stored data is read out through four 10 MHz taps to the processor board.

The ROIC shifts the waveform samples into an Acoustical Imaging Module (AIM), which digitizes the samples and transmits them over a high speed serial link to the console data processor board. The AIM is also responsible for driving the ROIC and acoustic transmitter with control signals from the host computer. The console board receives the digital samples from the AIM and reorders them into a conventional raster scan format. After a complete plane has been collected and ordered, the console board performs an offset correction on each pixel to compensate for detector array elements having different characteristic offsets. If the board is operating in quadrature detection mode, magnitude and phase information is calculated for that plane$^2$. The final console board task is to average data over frames, which improves the signal-to-noise ratio, reduces speckle effects common to coherent wave systems, and decreases the data transmission rate. The console board outputs magnitude and phase values for each pixel in every plane to the image processing board.

The last step in the data chain consists of a Digital Signal Processing (DSP) board.

---

1 A plane is defined as a temporal slice in the three-dimensional data set whose thickness and spacing are specified by the user.

2 Magnitude = $(I^2 + Q^2)^{1/2}$; Phase = $\arctan(I/Q)$; I = (+I) - (-I); Q = (+Q) - (-Q);
Basic DSP tasks include gain correction, 2D bilinear interpolation, gray scale compression, and RS-170 display formatting. Gain correction compensates for each transducer array element having a different pressure/voltage transfer function. Two-dimensional bilinear interpolation and a gray scale look-up table expand the image and translate its intensity values into shades of gray, which can then be RS-170 formatted and displayed on a monitor. All of these tasks are essential to system operation. Additional processing techniques may be used to further enhance and refine the acoustic imagery. The algorithms developed within this thesis will be deployed on the DSP board in addition to those previously mentioned.
3 Pulse Repetition Frequency

3.1 Introduction

The Lockheed Martin acoustic imager produces a short 3 MHz sinusoidal voltage pulse. An acoustic transducer converts this high-voltage signal into an ultrasound pressure wave, which suffers frequency-dependent attenuation as it propagates through water. This signal interacts with objects in the environment and may be reflected back towards the imaging system. An acoustical lens focuses returned waves onto a composite piezoelectric transducer array, which converts the incoming pressure signal to an electric charge. This charge, as a function of time, reflects the environment through which the wave journeyed, and can be used to construct an image of that world. The various systems through which the signal propagates affect the pulse’s shape. If these systems are modeled as being Linear and Time Invariant (LTI), then the output waveform equals the convolution of the input signal and each individual system’s impulse response.

3.2 Problem Statement

The Pulse Repetition Frequency (PRF) is defined as the inverse of the duration between successive signal transmissions. A PRF must be chosen to ensure that previously transmitted waves have been sufficiently attenuated and will not interfere with the current pulse. The prime determinant of this is water attenuation, which varies from \( \frac{1}{2} \) to 10 dB per meter, for frequencies between 1 and 5 MHz, respectively [10]. Therefore, considering the smallest attenuation factor of \( \frac{1}{2} \) dB per meter in a worst-case
analysis, a wave must propagate through 40 meters of water for the 1 MHz frequency component to be ten times weaker. However, for the maximum attenuation of 10 dB per meter, a wave would only have to propagate 2 meters for the 5 MHz component to be attenuated by the same fraction. Hence, low frequency components endure longer than high frequency components do.

High PRF values result in more waves returning to the acoustic imager, allowing it to track faster moving objects. However, high rates can also cause interference between successive transmissions and confuse range determination algorithms. Thus, specific tradeoffs must be considered when selecting a PRF. This chapter provides a framework to determine the maximum rate at which pulses can be sent by analyzing the various subsystems that affect the transmitted signal. The first section of this chapter identifies the Transducer Hybrid Assembly’s (THA) linear region and roughly characterizes it. The second section examines variations of pulse length and their effect on the signal’s frequency content. A third section applies Fourier analysis to the systems through which the pulse propagates. The fourth section experimentally measures the entire system transfer function and then employs it to derive the frequency response of the THA. Finally, the last section of this chapter calculates the spectrum of a received ultrasound pulse and uses it to derive a maximum PRF.

3.3 Linear System Range

In order to image distant objects, the transmitted signal must be strong enough to compensate for attenuation due to the water medium. Both the transmitter and water are roughly linear within a minimal operating range, but become non-linear at higher voltages.
and pressures [11]. Greater transmission amplification leads to a better SNR, but at the cost of increased power consumption. Hence, it is important to understand the ratio of input to output power for these systems within their entire dynamic range, in order to select a suitable operating level which allows targets to be imaged clearly without consuming excessive energy.

Unfortunately, the water medium is highly non-linear at typical operating pressures of 1-3 atmospheres [11]. These non-linearities produce harmonics of the fundamental frequency in the propagating wave. However, the water itself acts as a low-pass filter and attenuates these higher harmonics by several orders of magnitude [10]. Figure 3 illustrates seawater attenuation for a range of ultrasound frequencies. Above 1 MHz, pure water is very similar to seawater with regard to ultrasound, so this frequency response is valid in both fresh and salt water.

![Water Frequency Response](image)

**Figure 3:** Frequency Dependent Water Attenuation
Due to the severe attenuation of the higher harmonics, the only substantial frequency that is returned to the THA is that of the fundamental carrier. Thus, although the water generates non-linearities, these components can safely be ignored because their magnitude is trivial compared to that of the fundamental frequency. Therefore, the only subsystem which needs to be analyzed in its non-linear region is the THA itself.

Rather than fully describe the array’s linear and non-linear system ranges, they are simply characterized from a power efficiency standpoint. To fulfill this, the transmit transducer is driven with several sinusoids ranging from 5 to 150 mV r.m.s. This signal is converted to an acoustic pulse, transmitted through 15 cm of water, and received by the THA. The magnitude of this incident wave is calculated and recorded. An output versus input signal amplitude plot is shown in Figure 4. At lower voltages, the output is roughly proportional to the input signal, within experimental limitations. However, at higher levels, the intensity falls off. The point at which this curve becomes concave indicates the boundary between the linear and non-linear regions. In order for the THA to behave linearly, its incident pressure should be within the lower range of Figure 4 [7].
The primary goal of this analysis is to identify where the THA will behave linearly and to operate only within that region. The physical system components outside the electronics layer are all time invariant devices. Hence, the acoustic imager can be modeled as a Linear and Time Invariant (LTI) system, within this range. This is the primary achievement of this section. As a result, Fourier analysis can be applied to a class of input signals and the output waveform determined via convolution, which will allow construction of the maximum PRF later in this chapter.
3.4 Transmission Signal

A pure sinusoid occupies a very narrow frequency band, composed ideally of two impulses located at the fundamental frequency and its negative. However, the ultrasound transmission signal is a gated sinusoid, equivalent to an infinite sinusoid multiplied by a train of boxcars. Gating by a single boxcar in the time domain corresponds to convolution with a sinc function in the frequency domain. Thus, the transmitted signal contains a distribution of frequencies centered around its carrier. The breadth of this spectrum is inversely proportional to the length of the boxcar gate, so a longer pulse will occupy a narrower frequency band.

Four gated sinusoid pulses with lengths of 4, 16, 64, and 256-cycles have their spectra displayed in Figure 5 as dark solid curves. The shorter pulses have frequency representations that contain several large sidelobes, each visibly noticeable. As the number of cycles increases, these lobes occupy a narrower frequency band. Naturally, the wave’s energy content is directly proportional to its length.

The acoustic imager creates imperfect waveforms due to physical device limitations. Figure 6 shows one such wave. Four waves of different lengths were generated, sampled, and their Discrete Fourier Transform (DFT) computed. By interpolating these transforms, the signals’ Discrete Time Fourier Transform (DTFT) is approximated [12], shown as light dotted curves in Figure 5. These spectra will be compared later in this chapter with the frequency content of the received ultrasound wave, thereby enabling the selection of a maximum PRF.
Figure 5: Frequency Spectrum of Several Pulses

Figure 6: Acoustic Pulse
3.5 System Characterization

3.5.1 Introduction

The received ultrasound signal propagates through three sub-systems: an acoustic transmit transducer, a water medium, and a receive transducer array, as shown in Figure 7. These subsystems each possess a transfer function which affects the frequency content and shape of the propagating wave. Relying on the previous argument that described the attenuation of greater harmonics, the non-linearities within the transmit transducer and water medium can safely be ignored. A net transfer function is experimentally measured and used to calculate the frequency response of the transducer array. The final pages of this section examine the effects of water attenuation at increasing transmission distances. The results of this research will illustrate how imaging at different depths affects the received signal’s frequency spectrum, which in turn will allow a maximum PRF to be calculated.

Figure 7: System Block Diagram
3.5.2 Receive Transducer Array

The objective of this section is to determine the frequency response of the receive transducer array. To achieve this, the transfer function of the system shown in Figure 7 was measured twice, once with the transducer array and then again with a calibrated hydrophone in place of the array. The ratio of these two transfer functions can be used to calculate the array’s frequency response $H(j\omega) = Y(j\omega) / V(j\omega)$ via Equation 1.

$$
\frac{H_{\text{transducer array}}(j\omega)}{H_{\text{system with hydrophone}}(j\omega)} = \frac{H_{\text{hydrophone}}(j\omega) \cdot H_{\text{system with array}}(j\omega)}{H_{\text{system with hydrophone}}(j\omega)}
$$

**Equation 1:** Detector Array Frequency Response Calculation

The system transfer function was measured for both transducers in an aquarium tank by direct illumination at a distance of 15 cm. Water attenuation has negligible influence on the signal at this range [10] and is ignored. Equation 1 is then utilized to calculate the frequency response for just the transducer array; the result is shown in Figure 8 [7]. As this graph reveals, array elements are most responsive at 3.0 MHz, precisely where the system was designed to operate. The half-power bandwidth of the transducer array lies between 2.6 and 3.4 MHz. Other arrays may have responses slightly different from this one; however, Figure 8 is representative of most detector arrays.
3.5.3 Target Distance Effects

The system was previously calibrated to image at a distance of 15 cm from the detector. However, in real applications, this range may be increased up to 10 m. At greater distances, it becomes necessary to incorporate the effects of water attenuation into the transfer function. Because the individual subsystems are linear and time-invariant (under the previous assumption that higher harmonics are severely attenuated and can safely be ignored), the net response can be calculated by cascading an additional block into the system, as illustrated in Figure 10.
Cascading the original transfer function with an additional ten meters of water yields the frequency response shown in Figure 9. Although greater harmonics are severely attenuated and were previously ignored, the narrow frequency band around the carrier suffers frequency-dependent attenuation that cannot be dismissed. The longer exposure to water rapidly attenuates high frequencies and shifts the overall response down. The
resulting transfer function resembles a low-pass filter with cutoff point at 2.8 MHz. Note that the system’s response at 10 m is approximately one percent of that at 15 cm. This implies that a lower frequency pulse, such as one centered at 2.7 MHz, may be necessary to image distant targets.

3.6 Waveform Analysis

Now that the various system components that affect the acoustic waveform are understood, they can be employed to select the optimal Pulse Repetition Frequency (PRF). An acceptable pulse-to-pulse interference specification will depend heavily upon the operating environment. If the reflectivity coefficients among targets differ greatly, then a stricter interference level is required to discern weak reflectors from strong ones.

Based upon the previous section, a 2.7 MHz pulse shall be used for the remainder of this analysis. Its wavelength (0.55 mm) is approximately equal to that of a 3 MHz signal (0.50 mm), but the transfer function’s magnitude at 2.7 MHz is twice that at 3 MHz. A 2.7 MHz transmit pulse is attenuated at 3.1 dB per meter in water. The speed of sound in water is approximately 1,500 m/s. Thus, if the tolerance level specifies 40 dB of attenuation, then this signal must traverse a 13 meter path (an 8.7 ms delay) between transmissions. The system could then be operated at a maximum PRF of 116 Hz. Other environments may have less restrictive interference levels or operate at higher frequencies, resulting in a greater PRF. For example, if a 3 MHz wave is used, the same calculations could be repeated to yield a maximum PRF of 140 Hz. A general formula to compute the maximum PRF for a particular environment is given in Equation 2. This analysis can be reapplied to any application simply by changing the equation parameters.
1500-

Total Desired Attenuation

Equation 2: Maximum PRF Calculation

3.7 Conclusion

This chapter analyzed several areas of acoustic imaging to determine the maximum pulse repetition frequency dependent on a particular operating environment. Several key assumptions were made that resulted in a linear system from the transducer array to the processing electronics. This chapter first identified where the THA behaves linearly and then limited its analysis to that region. Next, it characterized the spectra of several waves with different lengths. The following section measured the overall system transfer function and calculated the frequency response of the transducer array. This response was then augmented to compensate for imaging at variable depths and levels of water attenuation. Once this extended transfer function became available, it was utilized to select an efficient operating frequency and calculate a maximum PRF for that setting. Equation 2 provides a framework for selecting a suitable PRF in order to maintain a tolerable pulse-to-pulse interference level given a specific attenuation rate. The resulting PRF will allow rapidly moving targets to be clearly imaged without confusing range determination algorithms and lead to a more comprehensible display.
4 Transmission Signal Duration

4.1 Introduction

The acoustic imager scans a target volume into a series of planes. A plane is defined as a three-dimensional block whose length and width correspond to the number of pixels in the transducer array. The thickness of each plane is configurable to an integral number of wavelengths, as is the spacing between successive planes. This is illustrated in Figure 11, in which $x$ and $y$ are the dimensions of the transducer array, $z$ is the plane depth, and $q$ equals the interplane spacing. The signal is sampled within these planes and a magnitude value which can be used to construct a range-relief image is then computed for each pixel. Fine range precision requires a short pulse to reside entirely within a single plane and not overlap into neighboring ones.

![Acoustic Plane Configuration](image)

**Figure 11:** Acoustic Plane Configuration
4.2 Problem Statement

In real imaging operations, the transmitted signal returns with a variable time delay corresponding to target depth, so the pulse is not necessarily aligned within a single plane. In Figure 12, Plane 1 contains an entire pulse with no overlap into neighboring planes. This is ideal. However, the second pulse resides in both Plane 2 and Plane 3, due to a time delay. This signal phase difference with respect to the planes decreases range resolution, because both planes contain a portion of the pulse. Additionally, part of the wave is “lost” in the interplane spacing, a region not sampled due to hardware limitations.

![Figure 12: Acoustic Signal in Different Planes](image)

Since the signal delay is target dependent, the optimal received pulse is defined as having the greatest probability of spanning an entire plane without interfering into neighboring planes. Mathematically, this involves maximizing the expected value of a weighted utility function dependent upon the target application. An example function might set utility equal to the area under the signal envelope in one plane minus that within all other planes. If precision were to become more desirable than detection, the error term could be weighted stronger to reflect that condition.
This chapter analyses how to determine the optimal transmission pulse given a particular utility function. The first section constructs a set of equations which quantify the criterion for an optimal signal. The next section uses these equations to derive an ideal solution without any constraints on the signal’s shape. The final section then applies this framework to a real acoustic pulse and determines the optimal transmission length for imaging targets in an aquatic environment.

4.3 Optimal Signal Criterion

The returned pulse arrives with some delay corresponding to the time necessary for the pulse to travel to and from a target object. This delay, \( \Delta \), is a random variable assumed to have a uniform probability distribution function (p.d.f.). Physically, this corresponds to different object depths, so a uniform distribution is a reasonable assumption. The received signal centered exactly at 0 mm is defined as \( y(t) \), but including the time delay, this becomes \( y(t-\Delta) \), as illustrated in Figure 13. Although

![Figure 13: Signal and Plane Spacing](image)

Figure 13: Signal and Plane Spacing
configurable, a 4 mm plane thickness and 1 mm interplane spacing will be used in this analysis. Equation 3 computes the absolute area underneath the signal \( y(t-\Delta) \) in Plane 2. Similarly, Equation 4 calculates the area within the two neighboring planes, which can be considered an error function.

\[
\text{Area}(\Delta) = \int_{\text{Plane \_ Begin}}^{\text{Plane \_ End}} |y(t-\Delta)| \cdot dt
\]

**Equation 3:** Signal Area Calculation

\[
\text{Error}(\Delta) = \int_{\text{Prior \_ Plane \_ Begin}}^{\text{Prior \_ Plane \_ End}} |y(t-\Delta)| \cdot dt + \int_{\text{Next \_ Plane \_ Begin}}^{\text{Next \_ Plane \_ End}} |y(t-\Delta)| \cdot dt
\]

**Equation 4:** Signal Error Calculation

These two terms can be combined to form a utility function that embodies two objectives, signal detection and range resolution. A long pulse will have a greater chance of being detected, but cannot provide fine range resolution. Likewise, a short pulse can discern between nearby objects, but is difficult to detect. One possible choice of utility functions is the difference between the \( \text{Area}(\Delta) \) and \( \text{Error}(\Delta) \) expressions, inversely scaled by the maximum wave amplitude. This function increases when more signal is present within a plane (detection) but decreases when that signal extends into other planes (resolution). This utility function will be used for the remainder of this research because it strikes a balance between detection and resolution. The expected value of this function can then be calculated, as shown in Equation 5, assuming a uniform p.d.f. for \( \Delta \) over the
range (-2.5, 2.5), in which $f_\Delta (\Delta) = \frac{1}{5}$. Together, these three equations generate a received pulse optimality criterion. The best choice of $y(t)$ is then found by maximizing Equation 5 over all realizable signals.

\[
E[\text{Utility}(\Delta)] = \frac{\int_{\Delta \in \text{Range } \begin{array}{c} \text{Begin} \\ \text{End} \end{array}} y(\Delta) \cdot [\text{Area}(\Delta) - \text{Error}(\Delta)] \cdot d\Delta}{\text{Max} \{ y(t) \}}
\]

**Equation 5:** Expected Utility

### 4.4 Ideal Boxcar Solution

With no restrictions on pulse shape, the optimal $y(t)$ which maximizes Equation 5 is a 5 mm boxcar. Although obvious, this conclusion can be reached by iterating through a range of signal lengths, applying the previous equations, and selecting the one which generated the greatest expected utility. For comparison, the expected utility for a boxcar over a range of lengths is shown in Figure 14. As this graph reveals, perturbing the signal

![Expected Utility vs. Boxcar Pulse Length](image)

**Figure 14:** Expected Utility of a Boxcar Pulse
duration around 5 mm doesn’t significantly alter the expected value. Hence, by shortening the pulse one millimeter, power consumption is reduced by 20%, but expected utility decreases only 5%. Therefore, if minimizing energy consumption becomes critical, the transmission duration can be slightly decreased with minimal impact on utility.

4.5 Actual Signal Solution

The actual received pulse has a distinctive low-frequency envelope much different than a boxcar. However, if an algebraic model is constructed to describe its envelope, then the same framework can be applied to calculate its optimal length. Such a model was experimentally found to be $y = \sin^2(\pi \cdot t / L)$, where $L$ is the signal length. By iterating through various lengths and subjecting the corresponding wave to the maximization in Equation 5, an optimal signal duration was obtained for the acoustic pulse, as Figure 15 illustrates. This signal is 9 mm long – approximately twice the target plane thickness.

![Optimal Receive Signal Envelope](image)

**Figure 15:** Optimal Receive Signal Envelope
Based upon a record of actual input/output acoustic pulses pairs, a 4.5 mm gated sinusoid input signal is required to generate a $y(t)$ of this form.

4.6 Conclusion

This chapter solved the problem of finding the optimal transmission signal length so that target objects can be reliably detected with fine range precision. The first section developed a set of equations that quantitatively describe an optimal signal. The second section iterated through numerous boxcar signals of variable length and selected the one that generated the greatest utility. Finally, the third section constructed an algebraic expression to represent the envelope of the actual pulse and then maximized its expected utility over all realizable lengths. When this process was applied to a system with a 4 mm plane thickness, 1 mm plane spacing, and a real acoustic pulse, the optimal transmit signal was determined to be a 4.5 mm gated-sinusoid.
5 Defective Pixel Compensation

5.1 Overview

A high-density detector array is critical to many imaging systems and yet is often the most difficult component to manufacture. For example, a typical acoustic transducer array has between 2% and 15% total element failure, resulting in an image with many pixels set at a fixed level, regardless of the incident signal’s intensity. The causes of these failures range from corrupted piezoelectric elements to faulty readout circuitry. In the displayed image, these errors take the form of randomly scattered pixels, rows, or columns, depending on the cause of the malfunction. Although it is impossible to retrieve useful information from defective pixels or those accessed through faulty electronics, it is still feasible to conceal them in the final image. For example, if only a small number of flawed array elements occur, then the defunct ones can be replaced by the average (or some other function) of those surrounding it. Hence, the dead values are hidden from the end user. However, if a large portion of the array is faulty, then more elaborate algorithms must be developed to reconstruct a useable image.

5.1.1 Previous Work

Defective pixels are by no means uncommon among detector arrays. These types of flaws often generate “shot noise,” characterized by scattered white and black pixels in the displayed image. The methods to eliminate these errors vary among applications [13, 14], but often employ non-linear processing techniques, such as median filtering. On the other hand, linear strategies, such as Gaussian or average filtering, are computationally
faster, but have the negative effect of smoothing a failed element over its surrounding pixels. By using a median or some other ranking filter, the sorting algorithm discards flawed values. This results in a clearer image than most linear methods generate, but often requires more processor cycles to complete.

5.1.2 Problem Statement

The goal of this chapter is to identify which array elements are defective, devise data structures to store such information, and then compensate for these flaws. Pixels fall into two broad categories: mostly functional or invalid. The methods used to determine which category each element belongs to are static in nature, and hence need to be performed only once when the system is initialized. After classification, these pixels require labeling and tracking. Without a priori knowledge, distinguishing between valid pixels exposed to a strong signal and defective ones saturated at a voltage rail is difficult. Hence, the initialization stage may require several stimulation levels. Once the defunct pixels are appropriately identified, a correction algorithm can be deployed to mask them. In the end, the resulting image should contain less evidence of array malfunctions. The problem this chapter poses, then, is how can defective elements be automatically identified, tracked, and compensated in the most efficient and effective manner?

5.2 Defective Element Identification

5.2.1 Introduction

Every detector contains flaws. These defects are typically permanent in nature, such as a burnt-out column amplifier or short-circuited interconnection wire. Hence,
array malfunctions only need to be identified during initialization, before the device is deployed. Failures can be divided into three categories based upon the areas they affect: rows, columns, and individual pixels. The underlying causes for these defects are still being discovered and such findings may lead to improvements. However, even with these refinements, the need to remove flaws will always be present. The goal of this section is to examine some of the fundamental problems with array-based acoustic imagery and to identify where these errors occur.

5.2.2 Causes of Pixel Failures

An entire transducer array row or column may fail to operate within single or multiple image planes. Although the underlying causes of these problems are not fully understood, some hypotheses exist. For example, an array row could malfunction if its row-select multiplexer channel dies. Similarly, columns will not generate meaningful data if their corresponding amplifier is burnt out. Random pixels fail and typically do so more frequently than an entire row or column does. Roughly 2 to 15% of all pixels suffer this type of failure. Fortunately, these flaws are uniformly distributed across the array, making compensation more effective. Random pixel failures are often caused by the interconnection between the piezoelectric element and the read-out circuitry being broken. This results in the active transducer element being physically separated from the electronics and permanently set at a fixed value. Figure 16 illustrates a variety of these flaws in a real acoustic image.
5.2.3 Defective Pixel Determination

During system calibration, a classification algorithm must correctly identify which pixels are defective. This process can be performed in several ways, two of which are considered here. The first method computes a histogram of pixel intensities given zero input signal, and then calculates one with full insonification. Elements inside a normal range of the histogram are valid, while extraordinary ones are defective. The second method tracks the input/output magnitude ratio through the pixel’s entire dynamic range. If this ratio is essentially constant, then that element has linear sensitivity and should be classified as valid; otherwise, it is defective. Slight deviations from the ideal response are also permitted, since most pixels will be functional but not perfect.
Histogram analysis is perhaps the simplest to implement and requires the least calibration. The fundamental concept is to discard any pixels that are drastically different then the majority. To test this, the acoustic transmitter is first disabled and the array sampled. An ideal pixel will return zero magnitude, while defective ones might generate any signal level. However, due to characteristic device offsets, even valid pixels may not return precisely zero. A histogram of actual pixel intensities is displayed in Figure 17. This logarithmic chart contains a heavy distribution of pixels with magnitude below 200 mV and a few above. Hence, 200 mV appears to be a good threshold for distinguishing between valid and invalid elements. A similar experiment can be performed with full insonification. In this scenario, valid pixels should be saturated, so the histogram will be heavily weighted toward large values. Any elements below a set threshold are classified as invalid and discarded. After both of these calibration tests are performed, the elements that failed either test are labeled as defective.

![Pixel Intensity Histogram, No Insonification](image)

**Figure 17:** Pixel Intensity Histogram given Zero Insonification
A second strategy to identify flawed elements examines the input/output mapping for each pixel in its full dynamic range. Ideally, as the acoustic stimulation level is increased, the output magnitude scales proportionately. Granted, pixels are not perfect, so some noise tolerance must be allowed. However, if the output signal is flat or decreases over time, then that pixel should be classified as defective. Figure 18 illustrates the input to output ratio for three synthetic elements. The first curve shows a functional pixel (1) generating a valid response to an increasing input signal. The second pixel (2), when illuminated by the same source, is unaffected. The third pixel (3) under the same conditions responds ambiguously. It might be slightly sensitive, but the output may have been caused by electronic crosstalk as well. Some mechanism is needed to automatically

Figure 18: Individual Pixel Responsivity
determine which curves correspond to functional pixels. Procedures that are quick to remove dead elements will also eliminate slightly sensitive, yet functional, ones. The selection criterion employed here will classify a pixel as valid if its slope is between 50% and 200% of the ideal slope. Based upon this restriction, the third curve (3) is classified as nonfunctional.

Either the histogram or input/output methods will classify most pixels correctly. However, by studying the input/output relationship, the true nature of what comprises a defective pixel is examined. If the element responds to an insonification source, then it is functional and should be used. Otherwise, it needs to be replaced. This philosophy is matched best by the input/output ratio method and will be used throughout the rest of this research.

5.2.4 Summary

Defects in either the acoustical transducer array or the integrated read-out circuitry are a direct consequence of fabrication difficulties. These flaws show up as permanently fixed pixel intensities in the displayed ultrasound image, irrespective of the volume actually being scanned. Although impossible to recover the complete image, if these defects can be identified, then they can be hidden from the end user. This section examined two techniques to automatically classify pixels as valid or flawed. The first was based on histogram analysis while the second examined individual pixel linearity. The second method is closely related to the true concept of a functional pixel, that being one which generates a response proportional to its input source. Hence, this input/output mapping technique will be used as the primary classification method. Once invalid pixels
have been identified, they must be tracked so that a correction algorithm can quickly locate them. Different approaches to this problem are studied in the next section.

5.3 Tracking Flawed Elements

5.3.1 Introduction

After a pixel, row, or column has been identified as defective, its location must be stored and corrective action taken on each incoming frame of data. Due to the relatively static nature of the array, which pixels are defective does not change significantly during device usage. With time more pixels may fail, so the problem is partially dynamic, but not to the extent that new failures need to be accounted for within a typical assignment. Hence, a data structure that stores this information needs to be updated only when the system is initialized. One mechanism of accomplishing this is to scan the image and store pixel-functionality information in a boolean map with a single bit corresponding to the validity of each element. Another method maintains a list of the coordinates for all faulty pixels. This section examines these two techniques.

5.3.2 Data Structures

In order to track defective pixels, a specialized data structure must be implemented to store such information. An ideal structure conforms to three specifications. First, it must be extremely quick to reference. If retrieving pixel validity information takes more than a few instructions, the cost of performing a full-image search may become prohibitive. Second, it does not require an excessive amount of memory, which is always a scarce resource. Third, it is functions well in a multi-processor
architecture. Two different data structures to fulfill these tasks are considered here: a boolean image map and a coordinate table, each with its own advantages and disadvantages.

A boolean image map uses a single bit to indicate whether an array element is valid or not. If the pixel is functional, then its corresponding map bit is set. On the other hand, if the pixel is defective, that bit is cleared. An example image map is shown in Figure 19, generated from the scan in Figure 16. To determine whether a pixel is valid, all that is necessary is to examine the boolean map at the appropriate location. If the bit there is cleared, then that pixel is defective and should be treated appropriately. This can be done within a constant time factor and requires a fixed memory allocation of precisely one bit per pixel. Hence, if the image is 64 by 64, a total of 4,096 bits or 512 bytes are needed. Additionally, the boolean map functions efficiently in a multi-processor environment. If every DSP chip handles a particular subsection of the entire image, each

![Boolean Image Map](image.png)

**Figure 19:** Boolean Image Map
processor only needs the map for the local region. Thus, an image map does not need to be replicated on each chip, nor do the processors have to communicate between one another.

The second tracking method maintains a sorted table containing the coordinates of every failed pixel. Such a data structure might look similar to Table 1. To determine which pixels need replacement, the system simply iterates through the table. This method offers neither the constant processing time nor the fixed memory requirements of the boolean image map. However, it always executes faster, because it stores only the flawed elements. If relatively few pixels are defective, this tactic will also consume less memory. Each list entry must store two pieces of information, the row and column in which the defective element lies and requires 2 bytes per pixel. If 100 elements are defective, then 200 bytes of memory must be allocated. This tactic scales well to a multi-processor environment, because the table can be subdivided into geographical regions, one for each processor. A more utilization-efficient strategy, available only if each chip has access to the full image, is to divide the entire table evenly between processors, regardless of geographical boundaries. This ensures that each DSP has an equivalent workload, but at the cost of additional memory consumption.

<table>
<thead>
<tr>
<th>Table Entry:</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Value:</td>
<td>(5,1)</td>
<td>(8,25)</td>
<td>(16,7)</td>
<td>(25,54)</td>
<td>(39,35)</td>
<td>(53,22)</td>
<td>(62,18)</td>
</tr>
</tbody>
</table>

**Table 1:** Coordinate Table Example
5.3.3 Comparison between Structures

Both of these data structures are well suited for working within a multi-processor environment, because they can be divided into distinct sections and processed independently. However, they differ by the number of computations required to access the data and the amount of memory necessary to store it. The one best suited for a particular application will depend primarily on array quality. If relatively few elements are flawed, then a coordinate table is the superior choice. However, if a sufficiently large number of pixels have failed, then the boolean image map becomes more efficient. Both the computational costs and memory requirements for these methods will be analyzed in this section, to establish a framework for selecting a structure based upon the target array’s quality.

Decoding a boolean image map is very straightforward. First, the memory address is computed, the value at that location read, and the single bit corresponding to the selected pixel unmasked. If that value is non-zero, then the element is functional and should not be replaced. Performing this scan requires a constant number of operations, regardless of how many pixels are defective. However, because the entire structure must be searched, this consumes more processor cycles than absolutely necessary. Memory storage is constant, as previously explained, and requires 512 bytes. Overall, the boolean image map remains an attractive option, because its requirements are very specific.

A coordinate table does not have the deterministic consumption properties of the image map. However, because only flawed pixels are stored, the need to scan over the entire image is eliminated. Additionally, the replacement algorithm is trivial to implement. The table is merely iterated through and a correction routine called at every
step. Both the number of operations and memory space needed for the coordinate table are directly proportional to the array defect rate.

A computational estimate for the number of instructions and memory storage needed for the boolean map and coordinate table for a 64 by 64 array is displayed in Table 2. A practical DSP implementation and derivation of these values is given in Appendix A and B. For very low defect rates, the coordinate table outperforms the boolean map in both categories. Above 10% pixel failure, the bitmap becomes more memory efficient; however, the coordinate table always consumes fewer processor cycles. Depending upon the availability of system resources, either one of these structures may be used.

<table>
<thead>
<tr>
<th>Pixels Defective:</th>
<th>2%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitmap # Inst's:</td>
<td>15,872</td>
<td>15,872</td>
<td>15,872</td>
<td>15,872</td>
<td>15,872</td>
<td>15,872</td>
</tr>
<tr>
<td>Table # Inst’s:</td>
<td>415</td>
<td>1,029</td>
<td>2,053</td>
<td>3,077</td>
<td>4,101</td>
<td>6,154</td>
</tr>
<tr>
<td>Bitmap # Bytes:</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Table # Bytes:</td>
<td>164</td>
<td>410</td>
<td>819</td>
<td>1,229</td>
<td>1,638</td>
<td>2,458</td>
</tr>
</tbody>
</table>

**Table 2: Data Tracking Processor Requirements**

A third option is to use a combination of the two structures. Although not obvious at this point what benefit that would provide, in the next section more advanced correction algorithms can be developed to take advantage of both. These routines depend on a boolean map, which unfortunately requires a large number of instructions to decode. However, if both data structures are present, then the coordinate table can be traversed to
locate defective pixels and the image map employed in the actual correction process. The prime disadvantage of this tactic is that it requires additional memory. However, if the compensation algorithm can be deployed more effectively with both structures present, the increase in visible performance may outweigh the additional memory cost.

5.3.4 Summary

This section studied two data structures to track defective pixels. Because arrays do not degrade extensively during device usage, flawed elements only need to be identified when the system is initialized. Once the device is active, a data structure is referenced to determine which pixels are invalid and need replacement. Two such structures were considered here: a boolean image map and a coordinate table. The boolean map possesses constant access time as well as fixed storage space. The coordinate table requires variable amounts of these quantities. For low defect rates, the image map consumes more memory than the coordinate table, but the reverse is true at high rates. The coordinate table is always faster and requires the fewest CPU instructions. A third alternative is to use both structures. This provides additional flexibility in the correction algorithm, but at the cost of increased memory consumption. All of these factors, including the defective pixel rate, will play an important role in selecting the best overall data structure.
5.4 Failed Pixel Compensation

5.4.1 Introduction

Once the invalid pixels in an array have been identified, they must be replaced. For this to occur, the defective pixel structure is referenced and a correction algorithm applied to each flawed element. The ideal routine will remove any evidence of failure with minimal computational cost. Five methods are examined in this section: Last Valid Sample, Nearest Neighbor, Average Filtering, Median Filtering, and MinMax Filtering. These algorithms have different processing requirements and may require the boolean image map. Which routine is best will depend on the data tracking structure available, which in turn will depend on array characteristics. To demonstrate these algorithms, each one is evaluated using the synthetic image and defect map in Figure 20, which contains several types of array failures. Appendix C provides a *Matlab* procedure to execute these five correction strategies. This test image is representative of an actual array and should serve as a useful indication of how well these methods will fare in practice.

![Original Image and Defect Map](image.png)

*Figure 20: Defective Image and Corresponding Boolean Map*
5.4.2 Last Valid Sample

The simplest, quickest, but least accurate pixel correction algorithm is to substitute the defective element with the previous one in the image, regardless of its prior validity. Because the flawed pixels are corrected in a raster-scan order, this replacement pixel is almost always guaranteed to contain a valid intensity, even though it may be substituted from a prior element itself. The only exception occurs when the defective pixel is the first element in a flawed column. In this case, there is no previous sample and the entire column remains flawed. Figure 21 illustrates its effectiveness on the test image. One difficulty with this method occurs when a high percentage of the array is flawed, especially if it contains a column failure. In this case, blocks of the same pixel will be replicated, resulting in stripe artifacts like the one seen in Figure 21. This technique can be rapidly executed in hardware, requiring just a single memory load and store operation. One benefit of this approach is that in addition to the actual image, it only requires the defective pixel coordinates. Thus, it can be implemented using either

Figure 21: Image Corrected using Last Valid Sample Algorithm
the image map or coordinate table as a data structure. Although the Last Valid Sample method does not necessarily generate the best images, it executes blazingly fast.

5.4.3 Nearest Neighbor

A second approach, Nearest Neighbor, replaces a flawed pixel with the closest valid one, provided it exists within a small neighborhood. This eliminates the stripe artifact seen earlier, but at greater computational cost. A second constraint of this algorithm is that it requires the boolean image map to determine which of the nearby pixels are valid. Figure 22 demonstrates this algorithm on the test image, which performs exceedingly well. Unfortunately, it executes slightly slower than Last Valid Sample. This is especially true when there are a large number of defective elements, in which case the closest valid pixel search may require several processor cycles to complete.

![Figure 22: Image Corrected using Nearest Neighbor Algorithm](image)

5.4.4 Average Filtering

A third correction strategy is to replace a defective element with the average of its surrounding pixels. This is fairly easy to implement, but suffers from two distinct
problems. First, any image shot noise will be smeared over the nearby area. Second, large blocks of flawed elements will remain if the averaging kernel is too small. Application of this filter is demonstrated in Figure 23. One advantage of this technique is that it does not require the full boolean map to operate, since pixels are averaged without examining their validity. However, if a large portion of the array became defective, this algorithm could be modified to use only valid elements, at an additional processing cost. An averaging filter is fairly efficient to implement and requires a fixed number of instructions. Hence, it would be useful on arrays with a small and evenly distributed number of defective pixels.

![Image Corrected using an Averaging Filter](image)

**Figure 23:** Image Corrected using an Averaging Filter

5.4.5 Median Filtering

A fourth approach to remove image defects is to replace the flawed element with the median value of its neighbors, as shown in Figure 24. This has the advantage of completely hiding small errors, works fairly well with large blocks, and does not require the boolean image map. Most of the defective elements are likely to fall on either the
high or low ends of the median filter and will be discarded. Hence, they do not need to be
removed before the sorting algorithm is performed. The prime disadvantage of this
technique lies in its latency, because the algorithm includes a sorting procedure.

Figure 24: Image Corrected using a Median Filter

5.4.6 MinMax Filtering

MinMax filtering is the fifth and final defective pixel compensation algorithm and
involves two iterations to remove flawed elements. First, an erosion filter is passed over
each failed pixel, which replaces the defective element with the minimum value of all its
neighbors. Next, a dilation filter is applied, which exchanges the flawed pixel with the
maximum value of its neighbors and itself. The combination of these two filters removes
any extraordinary values generated by defective pixels. An image corrected by MinMax
filtering is displayed in Figure 25. This method performs remarkably well on the
example in Figure 20 and does not require the boolean image map. One advantage of this
technique is that it can be executed quite fast, since only the minimum and maximum
values are calculated. Hence, it can be implemented quicker than a Median filter, although slower than Nearest Neighbor.

![Image Corrected using a MinMax Filter](image)

**Figure 25:** Image Corrected using a MinMax Filter

### 5.4.7 Summary

This section evaluated five different error correction algorithms to remove defects from the displayed ultrasound image: Last Valid Sample, Nearest Neighbor, Average Filtering, Median Filtering, and MinMax Filtering. A useful comparison of these methods takes three factors into consideration. First, the algorithm must be computationally efficient. Second, it should enhance the defective image and generate a comprehensible display. Third, it needs to operate with the data tracking structure being used. Table 3 provides such a comparison for the five different algorithms.

Based on the criteria for the Lockheed Martin system, the Last Valid Sample and Nearest Neighbor algorithms dominate. Last Valid Sample is extremely fast and does not require the hefty boolean image map. Although it generates noticeable stripes in place of column failures, it performs outstandingly well on random shot noise. However, Nearest
Neighbor does the best job removing image defects at a fast rate. Unfortunately, this comes with additional processor costs and requires the boolean map. Determining which of these two strategies to use will depend primarily on array quality and human tolerance toward flaws in the displayed image.

<table>
<thead>
<tr>
<th></th>
<th>Last Valid Sample</th>
<th>Nearest Neighbor</th>
<th>Average Filter</th>
<th>Median Filter</th>
<th>MinMax Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row Defect</strong></td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Compensation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Column Defect</strong></td>
<td>Poor</td>
<td>Good</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Compensation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Random Defect</strong></td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Compensation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Large Block</strong></td>
<td>Good</td>
<td>Fair</td>
<td>Poor</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>Compensation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Instruction</strong></td>
<td>Excellent</td>
<td>Good</td>
<td>Fair</td>
<td>Poor</td>
<td>Fair</td>
</tr>
<tr>
<td><strong>Costs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Requires</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Boolean Map</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Compensation Algorithm Performance Tradeoffs

5.5 Correction Algorithm Implementation

5.5.1 Introduction

After identification, tracking, and correction algorithms have been specified and developed, they must be installed in the target system. Due to vast differences between computational architectures, these implementations vary. This section focuses on the
current Lockheed Martin system, for which a minimal review of the DSP hardware is provided below. Note that this is a particular implementation of the general algorithm that depends on available equipment. Hence, an extensive software package is not developed here, but a representative sample presented instead. This section concludes with an assembler implementation of the Last Valid algorithm.

5.5.2 DSP Architecture

Lockheed Martin’s current DSP board choice is an Alex Computer Systems Sharcl000, populated with six Analog Devices ADSP-21062 Sharc processors. The Sharc is a high-performance 32-bit floating-point digital signal processor outfitted with a dual-ported on-chip SRAM and integrated I/O peripherals supported by a dedicated I/O bus [15]. With its on-chip instruction cache, the processor can execute an instruction every cycle. Four independent buses for dual data, instructions, and I/O, plus crossbar switch memory connections ensure that the Sharc will be capable of supporting high input/output data rates. The six Sharc’s are currently in a formation consisting of a master root node, four slave nodes, and a graphics display node outfitted with a RS-170 output for image display on a monitor.

5.5.3 Example Implementation of Last Valid Algorithm

The following assembler code implements the Last Valid algorithm. This function replaces a defective element with the one immediately preceding it. Last Valid requires the flawed element coordinates as input, which are assumed to be in registers $x$ and $y$. This procedure is very efficient, requiring only seven instructions to complete, and can even be inserted inline to eliminate unnecessary jump instructions.
5.6 Conclusion

This chapter investigated defective array elements and explored methods to correct them. Pixels fail for various reasons, generating shot noise or row and column outages. Two different methods to automatically classify pixels as functional or invalid were developed and tested on acoustic image data. The first method involves histogram analysis and is computationally simple; however, the input/output mapping technique functions better in practice. Due to the relatively stable nature of array failures, these classifications only need to be periodically applied.

Next, two data structures, a coordinate table and boolean image map, were devised to track and maintain pixel validity information. The coordinate table is extremely efficient, but grows excessively large at high failure rates. On the other hand, the boolean map is computationally slow, but maintains all the necessary information for more advanced correction procedures. The best choice of these data structures will ultimately depend upon the selected compensation algorithm’s requirements.
Finally, five compensations algorithms were analyzed and tested on synthetic acoustic data. The Last Valid Sample method executes the quickest, but leaves noticeable column failure artifacts. Both the Average and Median filters function appropriately, but fall short in pixel replacement accuracy. The Nearest Neighbor and MinMax filters both do a remarkable job correcting image defects, although Nearest Neighbor uses less processor cycles. Hence, either the Last Valid Sample or Nearest Neighbor routine should be implemented in practice, depending upon the target application. This chapter’s main result is that defective array elements can now be concealed using various correction routines, which will improve the displayed image’s quality.
6 Object Recognition

6.1 Overview

An advanced feature of the acoustic imaging system is the capability to recognize unknown objects. While this field has been widely studied and is still a developing area of research, standard algorithms require more memory and/or processing capabilities than available in a compact real-time system. Hence, a clever mechanism to compress the storage requirements without introducing excessive overhead is necessary for a good recognition procedure. A novel way of performing this incorporates the Singular Value Decomposition (SVD). By decomposing an image into a series of low-rank matrices and saving only the most significant eigenvectors for reconstruction, memory storage requirements can be cut by more than an order of magnitude, with negligible computational overhead.

This chapter tackles the problem of object recognition. The first section devises a target identification algorithm based upon a variety of error criteria. The second section explores the effects of object distance, translation, and rotation as well as methods to compensate for them. The third section presents a summary of the SVD as a mathematical tool. The fourth section studies practical applications of the SVD and their relation to acoustic imaging. The fifth section integrates the SVD into the object detection routine and examines tradeoffs between identification accuracy and memory compression. The final section cites possible extensions to the recognition algorithm for future work.
6.1.1 Previous Work

Massive amounts of research have been and continue to be devoted to the topic of object recognition. Advances in computational strength and storage capacity continuously push into reality the envelope of what was previously deemed impossible. For example, vehicles can now be autonomously controlled using a variety of sensors to detect road boundaries and obstacles, as well as estimate the vehicle’s position and distance traveled [16]. Generally, two types of recognition approaches exist: global property matching and local feature mapping [17]. These methods differ in the way they attempt to classify an unknown object. Global matching involves finding a geometric transformation of the scene to some common model. This warping must compensate for factors such as rotation, scale, and displacement. After the image has been transformed, it is compared against a set of candidate models and the best match selected. The local feature mapping approach differs by searching for parameters that are spatially localized, such as edges, corners, or holes. This typically involves constructing a hierarchical database of object features to be traversed when identifying an object. While this method has been successful for a variety of applications, it is difficult to implement efficiently in a pipelined multi-processor system. Hence, the global matching strategy is used for this research, because of its implementation advantages.

6.1.2 Problem Statement

A recognition algorithm must be able to accurately identify an unknown object or determine that no likely match exists. It needs to be computationally inexpensive and memory efficient, so that it can be implemented in real-time. Additionally, the routine
must function properly in a variety of environments. It should degrade gracefully with increasing image compression and system noise levels. Satisfying all of these constraints can be a very difficult task. Indeed, an entire doctoral thesis might be prepared on just this topic. Therefore, the problem is reduced in scope by making several key assumptions. First, objects are assumed to be segmented. That is, just one target exists in each frame. Second, the test images only contain rotational transformations in two dimensions. Perspective correction could be performed in three dimensions, but not without significant additional work. Other simplifications will appear throughout the chapter as they become necessary.

6.2 Target Identification

6.2.1 Introduction

Object recognition algorithms offer numerous advantages to any imaging system. Within the realm of acoustic imaging, a user could be automatically alerted to specific targets, such as mines, cables, or even aquatic creatures. Autonomous robots could recognize boundaries from acoustic images and navigate accordingly. For example, a remote probe might examine underwater oil pipelines for cracks or other defects. This section examines the basic concepts of object identification from a global matching perspective. Several types of recognition metrics are evaluated using a testbed of “unknown” objects. The performance characteristics and computational costs of these methods are then compared using this data.
6.2.2 Recognition Metrics

Object identification is typically performed by selecting the best match between an unknown image and a bank of candidate targets. The comparison between these pictures is based upon one of several metrics. This section studies three such methods, which include Least Distance, Least Squares, and Greatest Correlation. The performance of these metrics is evaluated by their ability to recognize a sample set of objects. These tests can be repeated on images collected from a variety of aquatic environments to select the best metric for that application.

The Least Distance algorithm calculates a summation of the absolute difference between each of the two images’ pixel elements. This error, $\varepsilon$, is defined in Equation 6, where $A(x,y)$ and $B(x,y)$ are the pixel values of the known and unknown image at those coordinates. The more similar $A$ and $B$ are, the lower this error criterion will be. Thus, the identification algorithm will select the candidate that minimizes $\varepsilon$.

$$
\varepsilon = \sum_{y=1}^{64} \sum_{x=1}^{64} |A(x,y) - B(x,y)|
$$

Equation 6: Least Distance Metric Calculation

The Least Squares algorithm is very similar to Least Distance, but uses the square of the error instead of the absolute value, as shown in Equation 7. This metric harshly punishes large deviations between two images. Hence, small differences between objects
will be tolerated, but large structural changes or isolated bad data points are not. Again, the identification algorithm selects the match that minimizes this error.

\[
\varepsilon = \sum_{y=1}^{64} \sum_{x=1}^{64} |A(x, y) - B(x, y)|^2
\]

**Equation 7:** Least Squares Metric Calculation

The Greatest Correlation algorithm is slightly different than the previous two metrics. Instead of looking at the difference between two frames, it computes their correlation, which is guaranteed to be within the interval [-1,1]. A high correlation indicates that the objects are closely matched, so an identification algorithm should seek to maximize this quantity. However, in order to use the same minimization procedure for all three methods, the negative of the correlation will serve as the actual metric, as given in Equation 8.

\[
\varepsilon = \left( \frac{- \sum_{y=1}^{64} \sum_{x=1}^{64} (A(x, y) - \text{avg}(A)) \cdot (B(x, y) - \text{avg}(B))}{\left( \sum_{y=1}^{64} \sum_{x=1}^{64} (A(x, y) - \text{avg}(A))^2 \right)^{\frac{1}{2}} \cdot \left( \sum_{y=1}^{64} \sum_{x=1}^{64} (B(x, y) - \text{avg}(B))^2 \right)^{\frac{1}{2}}} \right)^{\frac{1}{2}}
\]

**Equation 8:** Greatest Correlation Metric Calculation
6.2.3 Metric Performance

These three metrics were compared using a testbed of ten reference targets, shown as the white icons on the top row in Figure 26. From each candidate, three distorted images were generated by a smearing operation, as illustrated in the lower three rows of Figure 26. Each of these forty images, composed of the ten reference and thirty warped images, was compared against the candidates and a match selected based upon the metric in question. Additionally, the accuracy of the recognition algorithm was judged by computing a “certainty” measure for each match. For example, if the first and second candidates score similarly, then it is likely that either one could be the actual object. Thus, the accuracy calculation corresponds to how certain the algorithm is of its identification.

Figure 26: Image Recognition Testbed

Both the Least Distance and Least Squares algorithms select the match that has the least error. Similarly, if the negative of the Greatest Correlation metric is used, the best match again will occur at the minimum (possibly negative) value. Whenever a
comparison between a reference target and an unknown object is made, an error is calculated based on Equations 6 - 8. Figure 27 illustrates three possible sets of error values generated by comparing an object with ten candidates. The selection algorithm always chooses the smallest value, but the certainty of that match varies with the distance to the other errors. For example, on the first line of Figure 27, the minimum error is much lower than the rest. Hence, there is a high certainty that this value does indeed correspond to the correct match. However, in the second case, it is quite likely that either the 1st or 2nd match could be correct, so a 50% accuracy is appropriate. The third example contains two separate clumps of metric evaluations. Since each element in the first group is within close proximity to the others, it is probable that any one of these could be correct. Thus, the certainty is much lower.

![Figure 27: Certainty Metric Error Evaluations](image)

An accuracy function must be constructed to satisfy several criteria and produce results similar to those in Figure 27. The first criterion is that the relative scale of the error values should not matter, so that multiplying every number by a non-zero constant
does not change the certainty measurement. This has physical significance when the insonification mechanism varies between applications. For example, if the illumination source is increased by a factor of $\alpha$, then the corresponding metrics will scale by $\alpha$ as well. However, this should not affect the object match certainty, since both the reference and test objects are merely scaled by a constant. The second criterion is that the absolute value of the metric should not factor into the identification accuracy. Adding a constant to all the error values does not change the distance between them and hence should not affect certainty. This also has direct application to the acoustic imaging system, because every array has a characteristic offset. The examples in Figure 27 illustrate these two specifications by containing no axis markings. The hash marks provide no information concerning scale or a zero-crossing, yet the relative accuracy of each set is easily perceived.

Equation 9 presents a certainty indicator which satisfies these specifications. To see how this indicator performs, several groups of error values were generated and the accuracy of selecting the lowest one computed. Table 4 displays the results. Columns $e(1)$ through $e(10)$ correspond to ten metrics, similar to the hash marks in Figure 27. The

\[
\text{Accuracy} = \frac{\sum_{i=1}^{N-1} \frac{1}{i} [e(i+1) - e(i)]^2}{\sum_{i=1}^{N-1} [e(i+1) - e(i)]^2}
\]

\textbf{Equation 9: Accuracy Calculation}
first column contains the evaluation of Equation 9 with these values. The first three data rows demonstrate that this indicator obeys the arbitrary scaling and offset invariants. Rows four through six correspond to the examples shown in Figure 27. The fourth row illustrates a very certain match, resulting in a 95% accuracy rating. The fifth row contains two metric values close together, while all others are much greater. As expected, the identification is classified as 48% certain. The sixth row corresponds to the bottom example of Figure 27, consisting of two distinct clusters of values, which yields an accuracy of 20%. Finally, the seventh row demonstrates a very uncertain case in which nine error values are clumped together, but one is significantly greater. Because these nine are so close together, there is roughly a \(1/9\) chance that the lowest one is the actual match. Thus, this table indicates that Equation 9 satisfies all of the certainty function criteria and can be used to judge the relative accuracy of any identification metric.

<table>
<thead>
<tr>
<th>Certainty</th>
<th>e (1)</th>
<th>e (2)</th>
<th>e (3)</th>
<th>e (4)</th>
<th>e (5)</th>
<th>e (6)</th>
<th>e (7)</th>
<th>e (8)</th>
<th>e (9)</th>
<th>e (10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 %</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>31 %</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>31 %</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>95 %</td>
<td>1</td>
<td>20</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>28</td>
<td>30</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>48 %</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>28</td>
<td>30</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>20 %</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>20</td>
<td>21</td>
<td>25</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>12 %</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 4:** Verification Data of the Accuracy Equation

### 6.2.4 Computational Costs

The object recognition algorithms must be implemented on a multi-processor DSP board capable of operating in real-time. Resources necessary for these algorithms must
be kept minimal, since limited processing capability is available. The Least Distance metric relies entirely on addition and subtraction operations, and hence is the fastest to compute. For each pixel, the algorithm calculates the difference between the two images and adds that to the total summation. This requires two addition-like instructions. The Least Squares metric operates similarly, but squares the individual error terms before summing them. Therefore, it exercises one multiplication and two addition-like instructions. The Greatest Correlation approach first computes the average value of the image. This entails summing over the entire frame and then dividing by the total number of elements. Since this single division operation is negligible, the average cost per pixel is one addition instruction to find the average value. For the reference targets, this task can be performed beforehand and is not factored into real-time costs. After computing these average values, the algorithm executes two subtractions, three multiplications, and three summations. Table 5 summarizes all of these instruction counts. Note that this chart does not include memory instructions, because all three algorithms require the same data access operations.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Multiplication Operations</th>
<th>Addition-like Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least Distance</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Least Squares</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Greatest Correlation</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 5: Recognition Metric Processor Requirements
6.2.5 Summary

This section devised and examined three global matching object recognition algorithms under a common minimization framework. It also created a certainty function to assess the accuracy of these routines. This equation is constrained to function appropriately under several invariant conditions and produce results that physically make sense. This section also explored the computational costs of these algorithms. The Least Distance algorithm is the most efficient, with Least Squares second, and Greatest Correlation third. However, these algorithms performance have not yet been fully tested. Before they can be, several other issues must be addressed, such as translational and rotational differences between the target and reference images.

6.3 Image Alignment

6.3.1 Introduction

When unknown objects are imaged, it is extremely unlikely that they will be viewed from the same perspective as the reference targets. Hence, some common orientation must be mandated to which all images can be warped before the recognition algorithm is applied. This section addresses these transformations as three distinct tasks: centering the object within the screen, scaling it to a standard size, and rotating it to a common orientation.
6.3.2 Two-Dimensional Transformations

The initial alignment task is to center a target on the screen. This can be accomplished by computing the first moment of the pixel's intensity along both of the primary axes' directions, normalized by the image weight. These moments correspond to the distance between the object's center of mass and the screen origin. By subtracting that distance from each pixel's coordinates, the target is centered. This process is illustrated graphically in Figure 28 using source code listed in Appendix D.

![Original Image](image1)

![Centered Image](image2)

**Figure 28:** Centering Algorithm Example

After being centered, the target must be scaled to a standard dimension. This is performed using the first moment of the radial distance to each pixel from the center of the screen, normalized by the target's intensity. The image is then scaled along both axes inversely proportional to this moment. To reduce aliasing artifacts when the scaling occurs by fractional pixel distances, a weighted average of nearby elements is used. The results of this algorithm can be seen in Figure 29, which utilizes code from Appendix E.
Figure 29: Scaling Algorithm Example

Targets are viewed from a variety of perspectives, resulting in different orientations and rotations. In order to apply a recognition algorithm, all the images must first be aligned along some common angle. This can be accomplished using a novel technique based on histogram analysis. First, the angle from the center of the screen to

![Center Angle Histogram](image)

Figure 30: Angle Histogram
each pixel is computed and weighted by that pixel’s intensity and radial distance. Next, one hundred histogram “angle bins” are formed, each corresponding to a range of possible pixel angles. The current element’s intensity is then added to whichever bin represents its angle. After this operation is performed over the entire array, the bin containing the greatest count corresponds to the target section furthest from the screen center. An example histogram is shown in Figure 30.

A small modification is made to this algorithm for improved performance. Instead of just selecting the bin which contains the greatest count, a discrete-time low pass filter is first applied to the histogram. This smoothes the histogram and gives it greater noise immunity. After the maximum angle has been selected, the image is rotated in the opposite direction by that amount. This process is demonstrated in Figure 31, using source code listed in Appendix F.

![Original Image Rotated Image]

**Figure 31:** Rotating Algorithm Example
6.3.3 Three-Dimensional Transformations

The alignment solution thus far has been for two-dimensional objects, and hence will only work on targets perpendicular to the detector array. However, this is not sufficient in general, because the observer’s perspective may change along three angles. A complete solution must take into account object rotation in three dimensions. Extending the transformation problem in this manner is quite difficult and not investigated in this thesis. The reader is encouraged to examine the local feature mapping strategy, studied in [17], as an alternative approach to handle three-dimensional rotations more elegantly.

6.3.4 Summary

This section addressed the fundamental problem of image alignment, which arises whenever objects are not viewed from the same position and angle as the reference targets. It is imperative to correct for translational and rotational differences before a recognition procedure is executed. This section broke these tasks into three distinct transformations: shifting, scaling, and rotating. Algorithms relying on image moments were developed to complete these objectives. These routines were implemented in Matlab, but can be ported to a dedicated DSP processor when necessary.

6.4 Memory Compression

6.4.1 Introduction

In order to apply any of the three recognition algorithms, a database of candidate images must be stored in memory. Considering that an image is a 64 by 64 array and
each pixel has 256 intensity levels, every frame will require 4,096 bytes of memory. An average reference set may contain fifty such targets and require 204,800 bytes of storage. DSP processors frequently have a small amount of on-chip memory, heavily supplemented by a shared memory pool. If all the processors access this common storage area simultaneously, these requests may bottleneck the entire pipeline. An alternative solution stores the images within a dedicated processor memory region. For example, an Analog Devices Share DSP chip contains two 32,768-word banks of integrated memory. If this area is used to store the reference images, computations should perform faster since the processor can avoid accessing slower off-chip memory. Hence, a need to compress reference targets clearly exists; however, such a compression technique must demand very little overhead. Otherwise, the savings gained by storing images within on-chip memory will be lost to the increased computational burden. One approach to this problem utilizes the Singular Value Decomposition (SVD). Although commonly viewed as a mathematical tool, the SVD also has multiple practical applications, such as the one proposed here.

This section develops an efficient mechanism to compress acoustic images. It first presents the SVD from a theoretical viewpoint and then derives several useful properties from it. Next, this section integrates the SVD into the acoustic imaging system to compress and store reference images. Finally, it compares computational overhead and memory storage requirements for different levels of image compression.
6.4.2 Singular Value Decomposition

The Singular Value Decomposition (SVD) theorem states that any $m \times n$ matrix $A$ can be factored into

$$A = U_{m \times m} \cdot S_{m \times n} \cdot V_{n \times n}',$$

where $S$ is diagonal and $U$ and $V$ are unitary ($U'U = I$). The diagonal entries of $S$ consist of the singular values of $A$ and are sorted in decreasing magnitude, i.e.,

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0,$$

where $r$ is the rank of the matrix $A$. If both $U$ and $V$ are expressed in terms of their columns, i.e.

$$U = [ u_1 \ u_2 \ \ldots \ u_m ]$$
$$V = [ v_1 \ v_2 \ \ldots \ v_n ]$$

then Equation 10 presents a very convenient way to reconstruct the original matrix from its singular components [18].

$$A = \sum_{i=1}^{r} \sigma_i \cdot u_i \cdot v_i'$$

**Equation 10: SVD Reconstruction Calculation**

The left singular vectors $u_i$ are equivalent to the eigenvectors of $AA'$. Similarly, the right singular vectors $v_i$ consist of the eigenvectors of $A'A$. The singular values $\sigma_i$

---

3 The apostrophe in this notation denotes the Hermitian transpose which is the entry-by-entry complex conjugate of the transpose. 'I' denotes the identity matrix.
equal the nonzero eigenvalues of both $A'A$ and $AA'$. Equation 11 shows an example
SVD. This research does not pursue the construction of a matrix’s SVD. Most popular
mathematical software packages, such as Matlab, include a function to compute the SVD
of a matrix. For the curious reader, a detailed approach to this derivation is given in [19].

$$\begin{bmatrix}
1 & 2 & 3 \\
3 & 2 & 1 \\
1 & 2 & 1
\end{bmatrix} \cdot \begin{bmatrix}
.64 & .71 & .31 \\
.43 & 0 & .9 \\
5.43 & 0 & 0
\end{bmatrix} \cdot \begin{bmatrix}
.55 & -.71 & .45 \\
0 & 2 & 0 \\
0 & 0 & .74
\end{bmatrix}^T$$

**Equation 11:** Example SVD Decomposition

### 6.4.3 Application of the SVD to Acoustic Imaging

The SVD has several important applications, both theoretical and practical [19].

Equation 10 states one of its most important properties this is directly applicable to
acoustic image compression. Figure 32 contains a picture of an acoustically imaged
plastic tower built from several concentric cylinders stacked on top of each other.
Treating this image as a 64 by 64 matrix, its SVD can be computed. Figure 33 displays
its first thirty-two singular values. In this and most acoustic images, these singular values

**Figure 32:** Acoustic Image of Multiple Stacked Cylinders
Figure 33: Singular Values of an Acoustic Image

fall off extremely rapidly over the first few entries. The second thirty-two values are (by definition) even smaller and not shown here. The original image can be approximately reconstructed using Equation 10 and selecting a smaller $r$, the effective rank of the matrix, based on the rate at which the singular values decay.

The images shown in Figure 34 were reconstructed from their singular components via Equation 10 with $r$ equal to 1, 2, and 8, respectively. The first entry, generated from one set of eigenvectors, yields an incomprehensible image. However, with two singular values, the tower becomes more recognizable and with eight, serves as

Figure 34: Images Reconstructed using the SVD
a fair representation of the original image. With fuzzy or unfocused images, the benefit of increasing $r$ and using more eigenvectors is less apparent. Hence, an SVD compression with effective rank between two and eight should suffice to preserve most acoustic images.

The prime benefit from the SVD is to store compressed reference targets for object identification. Constructing the SVD of a matrix is computationally demanding and cannot be implemented in real-time on a DSP chip. However, a program could compute the reference objects’ SVD off-line and then download them into the DSP’s memory space before system deployment. Then, when comparing an unknown and a reference object, the candidates can be reconstructed on the fly. Hence, the SVD is an excellent compression technique for real-time image processing.

6.4.4 Memory Efficiency vs. Computational Costs

The SVD has proven to be a worthwhile image compression technique, but at what cost? Storing a 64 by 64 image in its raw format requires 4,096 bytes. An algorithm that preserves only the significant eigenvectors could decrease this storage requirement, but demands several processor cycles. This section will examine such tradeoffs between memory savings and computational costs at a variety of compression levels.

A 64 by 64 matrix’s left and right singular eigenvectors both require 64 bytes, so 128 bytes of memory are necessary for each rank-one matrix within the decomposed image. If two sets of eigenvectors are stored per image, this becomes 256 bytes, and so on. Equation 10 explicitly determines image reconstruction costs and must be executed...
to decompress each stored reference target. A slight enhancement can be made by factoring the $\sigma_i$ term into $u_i$ and $v_i$ before deployment, to save an instruction. Hence the algorithm only uses a single multiplication per pixel within each rank-one matrix. If more than one set of eigenvectors is used, the procedure will demand another addition operation for every one after the first. Thus, an image decomposed into two rank-one matrices will require one addition and two multiplication instructions.

Table 6 presents a summary of these memory requirements and computational costs. As this chart illustrates, using the smallest number of eigenvectors results in both memory and computational savings. Of course, the accuracy of the object recognition algorithm also decreases with fewer eigenvectors. Selecting the best ratio requires understanding how the identification metrics fare at different compression levels, which will be studied in the next section.

<table>
<thead>
<tr>
<th># Vectors</th>
<th>Memory</th>
<th>Multiplications</th>
<th>Additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Image</td>
<td>4,096</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1,024</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>512</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 6:** SVD Computational Requirements
6.4.5 Summary

The SVD plays an important role in memory conservation for a variety of purposes. Although typically viewed as just a mathematical theorem, it has direct application to the acoustic imaging system. Any frame can be treated as a matrix and its SVD computed. By preserving only its most significant singular eigenvectors, the original image can be approximately reconstructed using a fraction of the original storage allocation. This increases computational costs, but such expenses are minimal provided only a few eigenvectors are used. The next section incorporates the SVD into the object recognition framework and determines how accurate the various identification methods are using decompressed reference images.

6.5 Recognition Results

6.5.1 Introduction

The final section of this chapter compares how effective the different recognition metrics are at various levels of image compression. It integrates the SVD into the existing framework and calculates its impact on the certainty measurement. Finally, it examines the tradeoffs between these recognition metrics and compression levels using the synthetic data from Figure 26.

6.5.2 Overall Performance vs. System Requirements

Determining the overall performance of the object recognition system takes multiple factors into consideration, all of which were studied in this chapter. A set of reference targets is first constructed. Next, these images are shifted, scaled, and rotated
into a standard position and orientation. Afterwards, they are compressed and stored in memory before the system is deployed. After being activated, the system detects and attempts to identify unknown objects. This involves receiving acoustic image data, decompressing the stored reference targets, applying an image recognition algorithm, and calculating the certainty of the final identification.

A computer program simulated all of these steps in software using the candidates and test objects in Figure 26. It first compressed the reference images to 12%, 6%, and 3% of their original size and then restored them to normal. This degraded their quality and generated artifacts similar to those illustrated in Figure 34. Next, the program applied three recognition metrics, selected a best match for each one, and calculated the accuracy of that identification. Table 7 compactly presents this information. The first column indicates the “unknown” object, identified as A1, A2, ..., J3, J4. Each letter corresponds to a different target and the following number indicates the version, which maps to the smearing order in Figure 26. The upper row in Table 7 reveals both the compression level and the recognition metric used for that identification process. The symbols 64E, 4E, 2E, and 1E correspond an image compressed using sixty-four (fully recoverable), four, two, and one eigenvectors, respectively. The next two characters state the identification metric; LD, LS, and GC correspond to Least Distance, Least Squares, and Greatest Correlation, respectively. In the table body, the percentages indicate the certainty rate for the unknown object listed in the first column using the compression ratio and recognition metric given in the upper row. Most of the identifications were successful and are displayed in the table as positive entries. The incorrect ones are negative. The program also derived two statistics for each compression level and
recognition metric combination: the average certainty rate and the number of misidentifications. As the software increased the compression ratio, fewer objects were correctly identified. The actual recognition metric has a small, yet noticeable effect on the number of misidentifications. Based upon this information, a balanced algorithm might use the Least Distance algorithm with four singular eigenvectors. This combination demands only 12% of the full image memory requirement, costs seven additional instructions per pixel, and correctly identifies 98% of the unknown test objects.

Table 7: Recognition Metric Performance at Various Compression Levels
6.5.3 Conclusion

This chapter tackled the problem of target identification in a memory constrained system. The first section developed three different object recognition metrics and analyzed in detail their computational costs on a DSP processor. The second section addressed the difficulties with identifying targets at various locations and rotations. This resulted in procedures to center an object on the screen, scale it to a constant size, and rotate it to a predetermined orientation. A third section adapted the singular value decomposition to serve as an image compression tool and incorporated it into the object recognition framework. The SVD allows reference targets to be stored with as little as 3% of their original memory requirements and negligible computational overhead. The last section evaluated the effectiveness of different compression ratios and recognition metrics to prove that the SVD has potential as an image compression tool. A program that saves only four singular eigenvectors conserves 88% system memory and costs only seven instructions per pixel. Although the recognition algorithms tested within this chapter had functioned entirely on artificial data, this framework can be reapplied when real acoustic imagery becomes available. Different compression ratios and identification metrics can then be evaluated on those targets, to determine the optimal combination for that particular environment.
7 Conclusions

7.1 Summary of Work

This thesis addresses many issues critical to the development of the Lockheed Martin acoustic imaging system. These tasks are divided into four categories: pulse repetition frequency, signal duration, defective pixel compensation, and object recognition. The pulse repetition frequency chapter (3) examined the time-varying acoustic signal that each transducer element receives. The first section in this chapter characterized the system from an input/output perspective. Several assumptions were made that resulted in a linear and time-invariant system from the transducer array to the processing electronics. The second section experimentally measured the overall system transfer function and then employed it to derive the transducer array’s frequency response. The third section studied the effects of water attenuation on imaging distant targets and utilized this information to select a transmission frequency dependent on object range. In the case when targets are imaged through 10 meters of water, a 2.7 MHz acoustic pulse should be used to maximize the returned signal’s magnitude. The final section established a framework to calculate the maximum pulse repetition frequency for a particular environment. Given a 40 dB pulse-to-pulse attenuation specification when transmitting a 2.7 MHz wave, the system should be run at a maximum PRF of 116 Hz.

The following chapter (4) focused on signal duration to solve the problem of finding the optimal transmission signal length such that targets can be reliably detected with fine range precision. The first section within this chapter developed a set of
equations to quantitatively describe an optimal signal in terms of a utility function. This utility function encompasses two desirable system qualities, signal detection and range precision. The function selected for this research was defined as the difference between the signal envelope in one plane minus that within all other planes. This function was used because it strikes a fair balance between detection and precision. The second section presented the ideal signal, given no restrictions on pulse shape, by iterating through all possible waveforms and selecting the one that maximized the utility function. This ideal wave took the form of a boxcar with length equal to the plane thickness. The third section constructed an algebraic expression in the form of $y = \sin^2(\pi t / L)$ to represent the actual pulse envelope and then maximized its expected utility over all realizable lengths. When this process was applied to a real system with 4 mm plane thickness and 1 mm plane spacing, the optimal transmit signal was determined to be a 4.5 mm gated-sinusoid.

The chapter on defective pixel compensation (5) explored the problem of how to conceal detector flaws in the displayed ultrasound image. It first investigated various reasons why transducer array elements fail and then analyzed their symptoms. Examples of these flaws include row and column outages as well as random pixel failures. Next, it developed two procedures to decide autonomously whether array elements are functional or invalid. The first such procedure was based on histogram analysis and removed any extraordinary pixels. The other method traced the input/output intensity ratio for each pixel. Because this mapping closely reflects the definition of a functional pixel, it is used throughout this research. Third, this chapter established two data structures, a coordinate table and a boolean image map, to store pixel validity information. Fourth, it designed correction algorithms to conceal flawed pixels and tested them on acoustic imagery.
These procedures included Last Valid Sample, Nearest Neighbor, Average Filtering, Median Filtering, and MinMax Filtering. Fifth, several issues for implementing these techniques were quantified. Finally, the chapter concludes with a table that illustrates these algorithms’ performance on various types of defects and their associated computational costs.

The last chapter (6) explored object recognition from a global property matching approach. The first section constructed three identification metrics, Least Distance, Least Squares, and Greatest Correlation, and analyzed them from a computational efficiency standpoint. The second section addressed several problems associated with viewing a target from an infinite number of possible locations and orientations. Three procedures were created to handle the tasks of centering, scaling, and rotating an object. The third section utilized the singular value decomposition to compress candidate images for efficient memory storage. The SVD allows reference images to be stored in memory at a fraction of the original requirements. The final section integrated the SVD into the object identification framework and evaluated the recognition algorithms on a testbed of synthetic data. Based upon the information presented in this chapter, a balanced compensation algorithm might use the Least Distance algorithm with four singular eigenvectors. This combination demands only 12% of the full image memory requirement, costs seven additional instructions per pixel, and correctly identifies 98% of the unknown test objects.
7.2 Further Research

This work can be extended in a plethora of directions. One of the difficulties in writing this document was the scarcity of actual acoustic imagery, which could not be readily generated. However, enhancements in detector fabrication will eventually allow streams of acoustic video to be collected. The algorithms developed in this thesis can then be re-evaluated to determine their true effectiveness. Every chapter could be augmented in this manner using such data.

The existing chapters can also be extended in new directions. In a signal and system chapter, additional research could enhance the accuracy of the target depth calculation. Such research might also study matched filters capable of reliably detecting acoustic pulses over noisy channels. This work could entail using depth-dependent bandpass filtering to limit signal frequencies to a particular band, depending upon target range. Once acoustic imagery becomes prevalent, image enhancement algorithms can be developed to amplify the information content seen by a human observer [20]. Techniques such as edge enhancement, contrast expansion, image fusion, and orthogonal reconstruction could then become excellent directions for research. Other expansion possibilities exist in the object recognition chapter. A local feature mapping strategy might support more robust identification algorithms than the global property approach does. Also, compensation routines for differences in object location and orientation could be extended to include three-dimensional rotations.
Appendix

A. Boolean Image Map Computational Costs

This appendix contains assembler code to scan through the boolean image map, determine which pixels are defective, and jump to the appropriate correction routine. The tracking software first computes the pixel’s memory address and reads the word at that location. Next, it unmasks and tests the appropriate bit. If that bit is cleared, then the pixel is invalid and the program jumps to a correction routine. Otherwise, the process is repeated until the entire image has been checked. This can be accomplished with the following code:

```
MOV a, BitmapStart ; Initialize a, the linear address, to the bitmap beginning

Loop_Pixels:
    /* Scan through the image map by incrementing the address pointer every iteration */
    CMP d, a, BitmapEnd ; If address > map size, then entire image has been scanned
    IZ d, Control ; Processing is done, jump back to control routine
    LD b, a ; b = mem[a], image map byte
    ADD a, a, 1 ; a = a + 1 (Memory is word-addressed)
    MOV m, 1 ; m = 0b0000 0000 0000 0001 (Set the mask)

Unmask_Bit:
    /* Mask out all bits but the one of interest for this 32-bit word */
    /* Shift the masking bit over once for each iteration to cover all 32 pixels */
    AND c, b, m ; c = b & m
    JZL c, Fix_Pixel ; if (c=0), pixel is bad, jump to correction routine
    SHL m, 1 ; m = m << 1
    JNZ m, Unmask_Bit ; if (m != 0), still have more bits to check, continue
    JMP Loop_Pixels ; Continue for next 32-bit word
```
The exact number of processor cycles required to scan through a boolean image map can be computed using this code. The Loop_Pixels routine executes once for every data word, or \(64 \times 64 / 32 = 128\) times. The Unmask_Bit loop cycles through 32 times, once for each bit. Each cycle requires four instructions, resulting in 128 operations plus the return jump. The main Loop_Pixels routine contains five instructions, plus the Unmask_Bit loop, for a total of 134 operations. This is executed 128 times, resulting in a final count of 15,872 instructions per image. If additional initialization code is required, then this count may vary slightly, but not by more than a few operations.

B. Coordinate Table Computational Costs

This appendix contains assembler code to traverse a coordinate table, determine which pixels are defective, and jump to the appropriate compensation routine. A prime advantage of the coordinate table is that it only stores defective elements and requires fewer loop iterations. The table is organized as a contiguous memory block with its first entry equal to the total number of elements. Every successive address contains the 16-bit coordinates of a defective pixel, sorted in increasing order. The high byte contains the pixel’s y-coordinate while the low byte contains the x-coordinate.

The tracking routine shown below performs two tasks. First, it loads the total number of defective elements from the beginning of the coordinate table. Second, the program loops over all array elements, loads the defective pixel coordinates from memory, and calls the correction routine. This loop repeats until the entire table has been processed.
The number of processor cycles necessary to scan through the coordinate table can be computed from this code. The *Loop_Pixels* routine must be executed once for every entry in the coordinate table. The required number of iterations is calculated by multiplying the total number of pixels, 4096, by the defective pixel rate. For example, if 10% of the pixels are flawed, the loop must iterate 410 times. Five instructions are executed each iteration, in addition to the initialization code. Thus, the total number of operations can be computed using the following formula:

\[
N = 5 \times (% \text{Defect}) \times 4096 + 5
\]
C. Correction Algorithm Procedure

This appendix contains a *Matlab* routine which applies one of the four defective pixel correction strategies to a 64 by 64 image. It takes the original image, its corresponding pixel map, and the desired correction strategy as parameters. It then loops over the array’s failed elements and executes the selected algorithm on those pixels.

```matlab
function [out] = correct (image, map, mode)
    out = image; sum = 0; count = 0;

    % Loop over all of the internal pixels
    for j = 2:63; for i = 2:63
        if map(i,j) == 0 % If pixel is defective, correct it
            % Last Valid Sample
            if mode == 1; out(i,j) = out(i-1,j); end;
            % Nearest Neighbor
            if mode == 2
                if map(i-1,j) == 1; out(i,j) = image(i-1,j); else
                if map(i+1,j) == 1; out(i,j) = image(i+1,j); else
                if map(i,j-1) == 1; out(i,j) = image(i,j-1); else
                if map(i,j+1) == 1; out(i,j) = image(i,j+1); else
                end; end; end; end; end;
            % Average Filtering
            if mode == 3
                sum = sum + image(i-1,j); count = count + 1;
                if map(i+1,j) == 1; sum = sum + image(i+1,j); count = count + 1;
                if map(i,j-1) == 1; sum = sum + image(i,j-1); count = count + 1;
                if map(i,j+1) == 1; sum = sum + image(i,j+1); count = count + 1;
                if map(i-1,j-1) == 1; sum = sum + image(i-1,j-1); count = count + 1;
                if map(i-1,j+1) == 1; sum = sum + image(i-1,j+1); count = count + 1;
                if map(i+1,j-1) == 1; sum = sum + image(i+1,j-1); count = count + 1;
                if map(i+1,j+1) == 1; sum = sum + image(i+1,j+1); count = count + 1;
                if count == 0; out(i,j) = sum/count; end; end;
            % Median Filtering
            if mode == 4
                out(i,j) = median([out(i-1,j) out(i+1,j) out(i,j+1) out(i,j-1)
                                    out(i+1,j+1) out(i+1,j-1) out(i-1,j+1) out(i-1,j-1)]); end;
            % MinMax Filtering
            if mode == 5
                out(i,j) = min([out(i-1,j) out(i+1,j) out(i,j+1) out(i,j-1)
                                    out(i+1,j+1) out(i+1,j-1) out(i-1,j+1) out(i-1,j-1)]); end;
            end; end; end;
```

102
D. Offset Cancellation

The following *Matlab* function can be used to center an object within a 64 by 64 image. It first computes the primary moment of the pixel’s intensity along both of the primary axes’ direction. Next, it uses this value to determine the average weighted distance from the screen center to each pixel. The image is then shifted by that amount in the opposite direction.

```matlab
function T = center (A)
% Center Object in A, Compute first moment in X & Y
Mx=0; My=0;
for x = 1:64; Mx = Mx + ( (x-32.5) * sum(A(x,:)) ); end;
for y = 1:64; My = My + ( (y-32.5) * sum(A(:,y)) ); end;
% Normalize these distances by the weight of the object
Mx = round(Mx / sum(sum(A))); My = round(My / sum(sum(A)));
% Compute T as the shifted version of A
T = zeros(64,64);
for y = 1:64; for x = 1:64;
    if ( ((x+Mx) > 0) & ((x+Mx) < 65) & ((y+My) > 0) & ((y+My) < 65) )
        T(x,y) = A(x+Mx, y+My);
    end; end; end;
```

E. Object Scaling

After being centered within the image, an acoustic target must also be scaled to a standard dimension. The following procedure accomplishes this. The first moment of the radial distance to each pixel from the center of the screen is computed. Next, this value is normalized to the intensity of the target, so that it measures the object’s true size. Finally the image is scaled along both axes inversely proportional to this moment. A weighted average of pixels is utilized to reduce image aliasing artifacts when the image is resized.
function S = scale (T)

% Scale Object in T, Compute distance from origin for matrix T
Mxy = 0;
for y = 1:64; for x = 1:64;
    Mxy = Mxy + T(x,y) * sqrt((x-32.5)^2 + (y-32.5)^2);
end; end;

% Normalize this distance by the weight of the object
Mxy = Mxy / sum(sum(T));
Compression = Mxy / 16;

% Expand or Shrink the image as necessary
S = zeros(64,64);
for y = 1:64; for x = 1:64;
    sx = (x-32.5) * Compression + 32.5;
sy = (y-32.5) * Compression + 32.5;
    nx = round(sx); dx=sx-nx;
    ny = round(sy); dy=sy-ny;
    if ( (nx > 1) & (nx < 64) & (ny > 1) & (ny < 64) )
        if (sx > nx)
            S(x,y) = (l-dx)*T(nx, ny) + (dx)*T(nx+1,ny);
        else
            S(x,y) = (1+dx)*T(nx, ny) + (-dx)*T(nx-1,ny);
        end
        if (sy > ny)
            S(x,y) = S(x,y) + (1-dy)*T(nx, ny) + (dy)*T(nx,ny+1);
        else
            S(x,y) = S(x,y) + (1+dy)*T(nx, ny) + (-dy)*T(nx,ny-1);
        end
        S(x,y) = S(x,y) / 2;
    end;
end; end; end;

F. Image Rotation

Once an object is centered and scaled within the screen, it must be rotated to a common orientation. The following procedure accomplishes this using a novel strategy based on histogram analysis. First, the angle from the center of the screen to each pixel is computed and weighted by that pixel’s intensity and radial distance. Second, a histogram of these values is constructed and a discrete low-pass filter applied. Third, the bin with the greatest count is selected to represent the object’s bearing. Finally, the image is rotated in the reverse direction by that bin’s angle.
function R = rot(S)

% Rotate Object to a predetermined angle using angle histogram
Sang = zeros(100,1);
for y = 1:64; for x = 1:64;
    weight = S(x,y) * sqrt((x-32.5)^2 + (y-32.5)^2);
    ang = ceil((atan2((x-32.5), (y-32.5)) + pi) * 50 / pi);
    Sang(ang) = Sang(ang) + weight;
end; end;

% Apply a discrete low-pass filter to the histogram
s = zeros(100,1);
for i = 1:100;
    for j = 95:105;
        s(i) = s(i) + Sang(rem(i+j,100)+1);
    end;
end;

% Determine which histogram bin contains the most counts
[j,l] = max(s);
ang = -(j * pi / 50) + pi;

% Rotate Object in S correspondingly
R = zeros(64,64);
for y = 1:64; for x = 1:64;
    sx = cos(ang)*(x-32.5) - sin(ang)*(y-32.5) + 32.5;
    nx = round(sx); dx=sx-nx;
    sy = sin(ang)*(x-32.5) + cos(ang)*(y-32.5) + 32.5;
    ny = round(sy); dy=dy-ny;
    % Attempt to avoid aliasing artifacts
    if ( (nx > 1) & (nx < 64) & (ny > 1) & (ny < 64) )
        if (sx > nx)
            R(x,y) = (1-dx)*S(nx, ny) + (dx)*S(nx+1,ny);
        else
            R(x,y) = (1+dx)*S(nx, ny) + (-dx)*S(nx-1,ny);
        end
        if (sy > ny)
            R(x,y) = R(x,y) + (1-dy)*S(nx, ny) + (dy)*S(nx,ny+1);
        else
            R(x,y) = R(x,y) + (1+dy)*S(nx, ny) + (-dy)*S(nx,ny-1);
        end
    end
    R(x,y) = R(x,y) / 2;
end; end; end;
References


