Night Vision and Electronic Sensors Directorate

AMSRD-CER-NV-TR-C259

Multiscale Anomaly Detection and Image Registration Algorithms for Airborne Landmine Detection

Fort Belvoir, Virginia 22060-5806

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2008	3. REPORT TYPE AND DATES COVERED Thesis - 2006

4. TITLE AND SUBTITLE Multiscale Anomaly Detection and Image Registration Algorithms for Airborne Landmine Detection	5. FUNDING NUMBERS DAAB07-01-D-G601; 0073
6. AUTHOR(S) Jeffrey D. Barnes	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) UNIVERSITY OF MISSOURI - ROLLA Rolla, Missouri	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army RDECOM CERDEC Night Vision and Electronic Sensors Directorate 10221 Burbeck Road Fort Belvoir, Virginia 22060	10. SPONSORING / MONITORING AGENCY REPORT NUMBER AMSRD-CER-NV-TR-C259
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited	12b. DISTRIBUTION CODE A

13. ABSTRACT *(Maximum 200 words)*

Multiscale techniques such as the wavelet transform have become a powerful tool for image compression, denoising, and analysis. This thesis outlines the use of such techniques for developing a multiscale processing stream for the purpose of airborne landmine detection. In this work, the Reed-Xiaoli (RX) multispectral anomaly detection algorithm is extended to perform detection within the shift-invariant wavelet representation of a given single-band image. A test statistic is derived and shown to better detect anomalies in a correlated noise background than the single-band RX algorithm. The results of a detection algorithm using the shift-invariant wavelet representation and a multi-layer perceptron neural network are also discussed. A scale-space image registration algorithm is presented, where the scale-invariant feature transform (SIFT) has been used to search for control points between two images. The use of SIFT allows for image features invariant to scaling, translation and rotation to be matched in feature space, resulting in more accurate image registration than traditional correlation-based techniques.

14. SUBJECT TERMS Reed-Xiaoli (RX), RX algorithm, scale-invariant feature transform (SIFT), landmine detection, LAMD, continuous-time wavelet transform (CWT), discrete wavelet transform (DWT), multiresolution analysis (MRA), multilayer perceptron feed-forward neural network (MLPNN)	15. NUMBER OF PAGES 103
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT None

Night Vision and Electronic Sensors Directorate

AMSRD-CER-NV-TR-C259

Multiscale Anomaly Detection and Image Registration Algorithms for Airborne Landmine Detection

A Thesis Presented By

Jeffrey Barnes

University of Missouri – Rolla

May 2008

Countermine Division
FORT BELVOIR, VIRGINIA 22060-5806

MULTISCALE ANOMALY DETECTION AND IMAGE REGISTRATION ALGORITHMS FOR AIRBORNE LANDMINE DETECTION

by

JEFFREY D. BARNES

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI–ROLLA

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

2006

Approved by

Sanjeev Agarwal, Advisor

Randy Moss

David Grow

ABSTRACT

Multiscale techniques such as the wavelet transform have become a powerful tool for image compression, denoising, and analysis. This thesis outlines the use of such techniques for developing a multiscale processing stream for the purpose of airborne landmine detection. In this work, the Reed-Xiaoli (RX) multispectral anomaly detection algorithm is extended to perform detection within the shift-invariant wavelet representation of a given single-band image. A test statistic is derived and shown to better detect anomalies in a correlated noise background than the single-band RX algorithm. The results of a detection algorithm using the shift-invariant wavelet representation and a multi-layer perceptron neural network are also discussed. A scale-space image registration algorithm is presented, where the scale-invariant feature transform (SIFT) has been used to search for control points between two images. The use of SIFT allows for image features invariant to scaling, translation and rotation to be matched in feature space, resulting in more accurate image registration than traditional correlation-based techniques.

ACKNOWLEDGMENT

I would first like to thank my advisor, Dr. Sanjeev Agarwal, for the countless discussions we have had regarding airborne mine detection and image processing which have helped to make this thesis possible. His support has allowed me to greatly enhance and expand my engineering skills, and the knowledge he has imparted to me will help me to be successful in my future technical endeavors. My gratitude is extended to Dr. David Grow for agreeing to serve on my committee and providing some important suggestions during the brief meetings that we have had during my graduate program. I also appreciate Dr. Randy Moss for being among my committee members.

I would like to thank my peers in the ARIA Laboratory, especially Deepak Menon and Spandan Tiwari, for many interesting discussions that have helped me to better understand the methods and techniques of mine detection. The airborne applications group at the Countermine Division of the U.S. Army's Night Vision and Electronic Sensors Directorate (NVESD) deserve many thanks for providing me with the opportunity to work on this project, and for hosting me during my visits to Ft. Belvoir. Finally, I must express the deepest gratitude to my family for constantly supporting me during my years at UMR.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Multiscale processing has become popular for signal and image analysis in recent times. The primary focus of this research has been to apply multiscale processing techniques with applications to an airborne mine detection system. Mine detection research aims to facilitate cost-effective humanitarian demining as well as to gain a significant advantage in tactical offense and defense capabilities. A wide variety of technologies have been under experimentation in order to develop safe and reliable solutions to the landmine detection problem. This thesis addresses the use of multiscale techniques in two important areas of any imaging-based airborne mine detection system: target detection and image registration.

1.1. AIRBORNE LANDMINE DETECTION

Most mine and minefield detection technologies can be classified as either being ground-based or airborne. Ground-based technologies usually include some device which scans the ground with some electromagnetic sensor. This sensor can be either handheld, or mounted on a vehicular platform. Specific details of such ground-based mine detection platforms can be found in [1].

The major drawback to any ground-based mine detection system is its limited standoff distance and large scanning time. This has led to the consideration of airborne systems as a viable alternative to ground-based detection. Collection of data over very large areas requires the use and development of automatic and/or aided target recognition (ATR/AiTR) systems. Most airborne platforms currently under research employ some type of active or passive electro-optical sensing technique to gather imagery. Data presented in this thesis have been collected with sensors operating in the mid-wave infrared (MWIR) spectral range. More recently collected data have been obtained from a variety of sensors operating in the MWIR, near infrared, and the visible red, green, and blue bands of the frequency spectrum. This newer data is currently held under a non-disclosure agreement and is not included in this thesis. The details of several types of imaging sensors can be found in [2].

Some of the U.S. Department of Defense programs that have been or are currently engaged in airborne landmine detection systems development are the following: Airborne Minefield Detection and Reconnaissance Systems (AMIDARS) [3], Remote Minefield Detection System (REMIDS) [4], Airborne Standoff Minefield Detection System (ASTAMIDS) [3], Coastal Battlefield Reconnaissance and Analysis System (COBRA) [5], Airborne Infrared Measurement System (AIRMS) [6], Airborne Far IR Minefield Detection System (AFIRMIS) [7], and the Lightweight Airborne Multispectral Minefield Detection System (LAMD) [8].

1.2. AUTOMATED ANOMALY DETECTION

Automatic target detection is an important step in any landmine detection system. When any significant amount of imagery has been collected from an airborne platform, it must be processed to provide the most relevant and meaningful information to the warfighter-in-the-loop (WIL) who is the higher-level decision maker. Many, if not most, images that are collected in this process have no minefield regions. It is necessary to filter out most of these images without minefields so that the WIL is not overwhelmed. An automated method to determine whether relevant information is contained within a given image must be employed.

The baseline detection algorithm in the COBRA program is the Reed-Xialoi (RX) algorithm as implemented by Holmes in [9]. The RX algorithm is also the prime candidate for implementation as the anomaly detector in the LAMD program. Other techniques for local anomaly detection include using Gaussian-Markov random fields (GMRF) [10] and stochastic expectation-maximization methods [11]. There is also significant research into false alarm mitigation (FAM) techniques [12] which aim to reduce anomalous detections which are not likely landmines. Further high-level processing, such as patterned minefield detection [13] and line detectors [14] can compliment the low-level mine detection algorithm to identify minefield-like patterns or distributions.

1.3. AUTOMATED IMAGE REGISTRATION

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints and/or by different sensors.

The major purpose of registration is to remove or suppress geometric distortions between the reference and sensed images, which are introduced due to different imaging conditions, and thus to bring images into geometric alignment [15]. For the airborne mine and minefield detection problem, image registration must be performed in order to provide a montage of imagery with targets for minefield analysis of the collected data. The registered images are also useful for the warfighter-in-the-loop analysis of minefield data. Co-registration between different bands of image data is required for sensor fusion and multi-band processing. An automated registration system can perform this operation with little or no human interaction.

1.4. WAVELET AND SCALE-SPACE TECHNIQUES

Wavelet based signal and image processing has been employed for many purposes over the past two decades [16]. One of the most recent uses of wavelets in image processing has been in the field of compression. The JPEG 2000 standard [17], which utilizes the wavelet transform to provide for compression, can deliver remarkably high compression rates when compared to its predecessor's (JPEG) discrete cosine transform based algorithm without drawbacks such as size constraints and blocky artifacts. The JPEG 2000 standard, along with a wavelet-difference reduction algorithm [18] are considered to be likely possibilities for baseline image compression within the airborne landmine detection programs.

Whereas the literature provides copious examples of the use of the wavelet transform for compression and denoising, it is used in this work as a tool for signal analysis. The critically sampled wavelet transform finds its place in the aforementioned fields, but is surprisingly a poor tool for extracting or identifying shape information from a given signal or image due to its lack of shift-invariance. This work makes use of the shift-invariant wavelet transform, in which a highly redundant representation arises from the transformed data making it a good tool for analysis. This type of wavelet transform is used for the anomaly detection algorithms presented in this thesis.

Scale-space techniques provide a method, similar to wavelets, for analyzing structures at different scales. Rather than using a wavelet to build a wavelet-domain description of a function, scale-space employs the Gaussian function on which to analyze signals and imagery. The scale-space in effect serves as a representation that attempts to mimic the mammalian visual cortex [19]. In this thesis the scale-invariant feature transform (SIFT) [20], a novel technique applying scale-space theory to image processing and object recognition, is implemented and evaluated for the purpose of image registration.

1.5. OVERVIEW OF THE THESIS

This thesis is organized into five sections. The first section introduces the reader to the landmine detection problem and gives a brief overview of the topics that have been discussed in detail throughout the thesis. The wavelet transform is introduced in the second section. The lack of shift-invariance in critically sampled transforms is discussed and examples are presented. The stationary, or shift-invariant, wavelet transform is also detailed in the this section. The third section is devoted to anomaly detection algorithms. The baseline RX algorithm is discussed in detail, and then used to derive a closely related algorithm which the author has called the multiscale RX-based algorithm. This algorithm performs anomaly detection in the wavelet domain. A multilayer neural network is then presented which is also used to perform detection on stationary wavelet coefficients. The results of both detection architectures are presented and discussed.

The fourth section details the use of scale-space techniques for automated image registration. The basic terminology and principles of image registration are presented, along with the current correlation-based method of control point selection. The scale-invariant feature transform is presented in detail along with discussion of applying the algorithm to the image registration problem. The thesis is concluded in the fifth section with a discussion of the results presented and relevant future work.

2. THE WAVELET TRANSFORM

2.1. DEFINITION OF THE CWT

The wavelet transform of a continuous-time signal $f(t)$ with respect to a *mother wavelet* $\psi(t)$ is known as the *continuous-time wavelet transform* (CWT) and is defined as

$$W(s,u) \triangleq \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{|s|}}\psi^*\left(\frac{t-u}{s}\right)dt, \tag{2.1}$$

where both $f(t)$ and $\psi(t)$ are square-integrable functions, i.e. $f(t), \psi(t) \in L^2$. The wavelet $\psi(t)$ has the additional property that

$$\int_{-\infty}^{\infty} \psi(t)dt = 0. \tag{2.2}$$

Note that Eq. 2.2 implies that the zeroth (DC) frequency component of the Fourier transform of $\psi(t)$ is zero, thus characterizing $\psi(t)$ as a bandpass function. Eq. 2.1 can be written more compactly by defining

$$\psi_{s,u}(t) \triangleq \frac{1}{\sqrt{|s|}}\psi\left(\frac{t-u}{s}\right). \tag{2.3}$$

Then

$$W(s,u) = \int_{-\infty}^{\infty} f(t)\psi^*_{s,u}(t)dt. \tag{2.4}$$

Note that $\psi_{1,0}(t) = \psi(t)$. The normalizing factor $1/\sqrt{|s|}$ ensures that the energy remains unchanged for all s and u, i.e.

$$\int_{-\infty}^{\infty} |\psi_{s,u}(t)|^2 dt = \int_{-\infty}^{\infty} |\psi(t)|^2 dt. \tag{2.5}$$

The parameter s represents the time dilation of the wavelet $\psi(t)$ and u represents its shift. Specifically, for a given value of s, $\psi_{s,u}(t)$ is a shift of $\psi_{s,0}(t)$ by an amount u along the time axis. $\psi_{s,0}(t)$ is a time-scaled (and amplitude-scaled) version of $\psi(t)$.

If the wavelet $\psi(t)$ satisfies the *admissibility condition*

$$C_\psi \equiv \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega \tag{2.6}$$

where $0 < C_\psi < \infty$ and $\hat{\psi}(\omega)$ is the Fourier transform of $\psi(t)$, then the inverse CWT exists and is given by

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{s^2} W(s,u) \psi_{s,u}(t)\ ds\ du. \tag{2.7}$$

It should be noted here that a wavelet need not satisfy the admissibility condition, and thus the inverse wavelet transform may not exist.

The CWT can be viewed as an operator which maps a square-integrable function of a single real variable, time, into a function of two real variables, scale and translation. This wavelet representation is a function of all possible scales and translations of the mother wavelet, and its support covers the entire plane of $\mathbb{R}^2$. This representation is inappropriate for numeric computations, and is highly redundant in the sense that the entire continuum of wavelet coefficients is not needed to reconstruct the original transformed function. It is then not unreasonable to search for a representation of the form

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d(j,k) \psi(2^{-j}t - k), \tag{2.8}$$

where the values $d(k,l)$ are related to the values of $W(s,u)$ at $s = 2^k$ and $u = 2^k l$. This is known as dyadic sampling of the CWT since the consecutive values of the discrete scales and the sampling intervals differ by a factor of two. The two-dimensional sequence $d(k,l)$ is commonly referred to as the *discrete wavelet transform* (DWT). This name is somewhat misleading since this is still the transform of a continuous-time signal with discretization only in the s and u variables of the CWT. The same name will be used later for wavelet transforms of discrete-time signals or discrete images. The representation in Eq. 2.8 is analogous to the

Fourier series, where a summation of weighted sines or cosines over a countable set of frequencies is used to construct a periodic continuous-time signal.

2.2. MULTIRESOLUTION ANALYSIS

While there are two ways to introduce the DWT, with the CWT being one, the method popular in the signal processing literature dealing with wavelet theory is the *multiresolution analysis* (MRA) developed by Meyer and Mallat [16]. The MRA is an important building block for construction of orthogonal wavelets and development of the DWT and filterbank algorithms. Wavelet representations such as those in Eq. 2.8 arise naturally in the context of an MRA.

2.2.1. Definition. An MRA provides the foundation for approximating functions in a sequence of nested linear vector spaces. Not every sequence of nested vector spaces yields an MRA. An MRA consists of a sequence $\{\mathbf{V}_j\}_{j\in\mathbb{Z}}$ of closed subspaces where the following properties are satisfied:

1. $\mathbf{V}_j$ is invariant to any translation proportional to the scale 2^j.

$$f(t) \in \mathbf{V}_j \Leftrightarrow f(t-2^j k) \in \mathbf{V}_j, \quad \forall (j,k) \in \mathbb{Z}^2 \tag{2.9}$$

2. A sequence of vector spaces is nested.

$$\mathbf{V}_j \subset \mathbf{V}_{j-1}, \quad \forall j \in \mathbb{Z} \tag{2.10}$$

3. Dilating a function in $\mathbf{V}_j$ by a factor of two guarantees that an approximation is defined at a coarser resolution.

$$f(t) \in \mathbf{V}_j \Leftrightarrow f(2t) \in \mathbf{V}_{j-1}, \quad \forall j \in \mathbb{Z} \tag{2.11}$$

4. The intersection of these subspaces is a singleton set containing the all-zero function.

$$\lim_{j\to+\infty} \mathbf{V}_j = \bigcap_{j=-\infty}^{\infty} \mathbf{V}_j = \{0\} \tag{2.12}$$

5. The union of these subspaces is dense in the space of square-integrable functions.

$$\lim_{j\to-\infty} \mathbf{V}_j = \text{Closure}\left(\bigcup_{j=-\infty}^{\infty} \mathbf{V}_j\right) = L^2(\mathbb{R}) \tag{2.13}$$

6. There exists a *scaling function* $\phi(t)$ such that $\{\phi(t-n)\}_{n\in\mathbb{Z}}$ is a Riesz basis of $\mathbf{V}_0$. This implies that there exist $A > 0$ and $B > 0$ such that any $f \in \mathbf{V}_0$ can be uniquely decomposed into

$$f(t) = \sum_{n=-\infty}^{\infty} a_n \phi(t-n) \tag{2.14}$$

where

$$A\|f\|^2 \le \sum_{n=-\infty}^{\infty} |a_n|^2 \le B\|f\|^2. \tag{2.15}$$

It should be noted that there are alternate logically equivalent axioms that can be used to define an MRA.

2.2.2. Approximation and Detail Subspaces. From the above definition of an MRA, a complicated function can be divided into simpler ones that can studied separately. The scaling function of Property 6 generates the *approximation subspace* $\mathbf{V}_0$. Along with Properties 1 and 3, the scaling function satisfies the dilation equation

$$\phi(t) = \sum_k h_1[k]\phi(2t-k), \tag{2.16}$$

where $h_1[k] \in \ell^2(\mathbb{Z})$. The scaling function $\phi(t)$ is a superposition of translated and scaled copies of itself, hence its name. Eq. 2.16 is sometimes called a two-scale relation, since it relates $\phi(t)$ to itself at an adjacent scale. The result can be generalized to relate the scaling function at any given scale to the next scale by replacing t with $2^{-j}t$, giving

$$\phi(2^{-j}t) = \sum_k h_1[k]\phi(2^{-(j-1)}t-k). \tag{2.17}$$

The subspace $\mathbf{V}_j$ is generated by the bases $\{\phi(2^{-j}t - k)\}_{j,k\in\mathbb{Z}}$. Since $\mathbf{V}_j$ is a proper subspace of $\mathbf{V}_{j-1}$, the space left in $\mathbf{V}_{j-1}$ not covered by $\mathbf{V}_j$ is labeled $\mathbf{W}_j$ and is orthogonally complimentary to $\mathbf{V}_j$ in $\mathbf{V}_{j-1}$. This space is called the *detail subspace*. The subspace $\mathbf{W}_j$ is generated by the bases $\{\psi(2^{-j}t - k)\}_{j,k\in\mathbb{Z}}$. The wavelet ψ also obeys a two-scale relation

$$\psi(t) = \sum_k g_1[k]\phi(2t-k), \tag{2.18}$$

and satisfies the properties noted earlier – it is square-integrable and bandpass. Here the wavelet is composed of dilated and translated copies of the scaling function. Properties of the sequences $h_1[k]$ and $g_1[k]$ will soon be discussed in detail.

Using direct sum notation, the relations between the approximation subspace $\mathbf{V}_0$ and detail subspace $\mathbf{W}_0$ are

$$\mathbf{V}_{-1} = \mathbf{V}_0 \oplus \mathbf{W}_0 \;\text{ and }\; \mathbf{V}_0 \cap \mathbf{W}_0 = \{0\}, \tag{2.19}$$

which can be generalized to

$$\mathbf{V}_{j-1} = \mathbf{V}_j \oplus \mathbf{W}_j \;\text{ and }\; \mathbf{V}_j \cap \mathbf{W}_j = \{0\}. \tag{2.20}$$

Figure 2.1 shows the relationship between the approximation and detail subspaces. Applying Eq. 2.20 recursively yields

$$\mathbf{V}_{j-1} = \bigoplus_{n=j}^{\infty} \mathbf{W}_n. \tag{2.21}$$

As $j \to -\infty$, the space $L^2(\mathbb{R})$ can be written solely in terms of the detail subspaces:

$$\lim_{j\to-\infty} \mathbf{V}_j = \bigoplus_{n=-\infty}^{\infty} \mathbf{W}_n = L^2(\mathbb{R}). \tag{2.22}$$

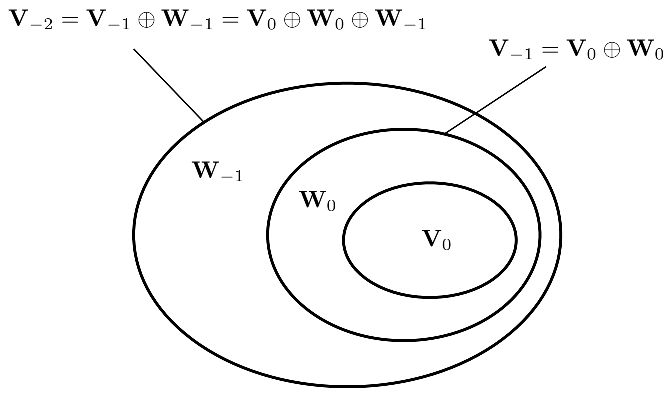

$\mathbf{V}_{-2} = \mathbf{V}_{-1} \oplus \mathbf{W}_{-1} = \mathbf{V}_0 \oplus \mathbf{W}_0 \oplus \mathbf{W}_{-1}$
$\mathbf{V}_{-1} = \mathbf{V}_0 \oplus \mathbf{W}_0$
$\mathbf{W}_{-1}$
$\mathbf{W}_0$
$\mathbf{V}_0$

Figure 2.1 Relationship between approximation and detail subspaces

This leads to the representation of a square-integrable function $f(t)$ by a weighted sum of wavelets over a countable set of translations and dilations, i.e.

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d(j,k)\psi(2^{-j}t - k). \tag{2.23}$$

2.3. DIGITAL FILTERING INTERPRETATION

2.3.1. Properties of Synthesis and Analysis Filters.

Supposing that an orthonormal MRA is to be constructed, the scaling function $\phi(t)$ must satisfy the following conditions:

1. The scaling function integrates to one:

$$\int_{-\infty}^{\infty} \phi(t)dt = 1. \tag{2.24}$$

2. It has unit energy:

$$\|\phi\|^2 = \int_{-\infty}^{\infty} |\phi(t)|^2 dt = 1. \tag{2.25}$$

3. The integer translations of $\phi(t)$ are orthogonal:

$$\langle \phi(t), \phi(t-n) \rangle = \delta(n). \tag{2.26}$$

From the two-scale relations in Eqs. 2.16 and 2.18, and along with the conditions in Eqs. 2.2 and 2.24, the sequences $h_1(n)$ and $g_1(n)$ then satisfy

$$\sum_n h_1(n) = 2, \tag{2.27}$$

and

$$\sum_n g_1(n) = 0. \tag{2.28}$$

These properties are found by integrating both sides of the two-scale relations. Eqs. 2.27 and 2.28 imply that the DC components of the discrete-time Fourier transform of $h_1(n)$ and $g_1(n)$ are 2 and 0 respectively. Thus $h_1(n)$ is a lowpass sequence and $g_1(n)$ is bandpass (or highpass).

Decomposition filters can now be defined from the sequences $h_1(n)$ and $g_1(n)$. The set of analysis (decomposition) filters is

$$\tilde{h}(n) = h_1(-n) \quad \text{and} \quad \tilde{g}(n) = g_1(-n), \tag{2.29}$$

and the synthesis (reconstruction) filters are

$$h(n) = \frac{h_1(n)}{2} \quad \text{and} \quad g(n) = \frac{g_1(n)}{2}. \tag{2.30}$$

The filters defined above can either be finite impulse response (FIR) or infinite impulse response (IIR) filters depending on the scaling function used to construct the sequences $h_1(n)$ and $g_1(n)$. Construction of these sequences from $\phi(t)$, and vice-versa, is discussed in detail in [16], [21], and [22]. It can be shown [21] that these four filters are perfect reconstruction quadrature mirror filters (PRQMFs). Each filter thus satisfies a power complementary condition,

$$\begin{aligned}
|\hat{h}(\omega)|^2 + |\hat{h}(\omega+\pi)|^2 &= 1, \\
|\hat{\tilde{h}}(\omega)|^2 + |\hat{\tilde{h}}(\omega+\pi)|^2 &= 1, \\
|\hat{g}(\omega)|^2 + |\hat{g}(\omega+\pi)|^2 &= 1, \\
|\hat{\tilde{g}}(\omega)|^2 + |\hat{\tilde{g}}(\omega+\pi)|^2 &= 1,
\end{aligned}$$

which has been expressed in the frequency domain. Relationships between the lowpass and highpass filters can also be presented. These follow from the properties of the sequences $h_1(n)$ and $g_1(n)$ [21].

$$\hat{h}(\omega)\hat{g}^*(\omega) + \hat{h}(\omega+\pi)\hat{g}^*(\omega+\pi) = 0 \tag{2.31}$$

$$\hat{\tilde{h}}(\omega)\hat{\tilde{g}}^*(\omega) + \hat{h}(\omega+\pi)\hat{g}^*(\omega+\pi) = 0 \tag{2.32}$$

The time-domain expressions below can be derived from those above:

$$\sum_n h(n)h(n+2k) = \frac{1}{2}\delta(k), \tag{2.33}$$

$$\sum_n g(n)g(n+2k) = \frac{1}{2}\delta(k), \tag{2.34}$$

$$\sum_n h(n)g(n+2k) = 0. \tag{2.35}$$

Similar properties can be written for the analysis filters since they are proportional to the reflections of $h(n)$ and $g(n)$. The equations above imply that the LPF response $h(n)$ is orthogonal to even translations of itself. The same holds for the HPF impulse response. Orthogonality also exists between the LPF response and even translations of the HPF response.

2.3.2. Synthesis and Analysis with PRQMFs. Shown in Figure 2.2 is the decomposition and reconstruction paradigm of the discrete-time signal $f(n)$. This provides the basis for signal analysis and synthesis using the wavelet transform. The input signal is convolved with the impulse response functions of the lowpass and

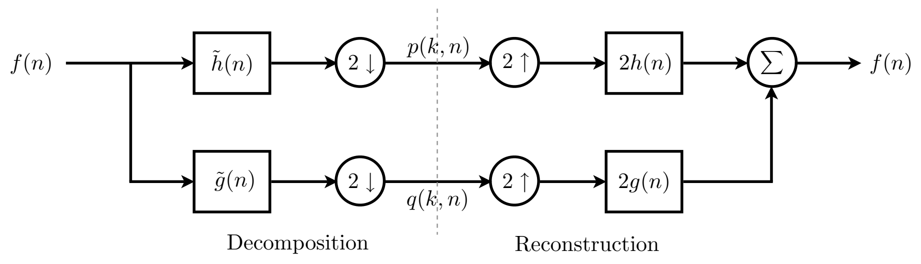


Figure 2.2 Signal decomposition and reconstruction with PRQMFs

highpass decomposition filters. These two resulting signals are then downsampled by a factor of two ($2\downarrow$) by only keeping the samples with even indices. The sequences p and q can be called the approximation and detail coefficients respectively. These coefficients can then be upsampled by a factor of two ($2\uparrow$) by inserting zeros between the adjacent samples. The upsampled coefficients are then convolved with their respective lowpass and highpass reconstruction filter and the result is summed, yielding the original signal $f(n)$.

The DWT is usually computed with multiple levels of approximation and detail coefficients. The cascade filterbank used to accomplish this multilevel decomposition is shown in Figure 2.3. The reconstruction filterbank is shown in Figure 2.4. Figures 2.2, 2.3, and 2.4 have been adapted from [21].

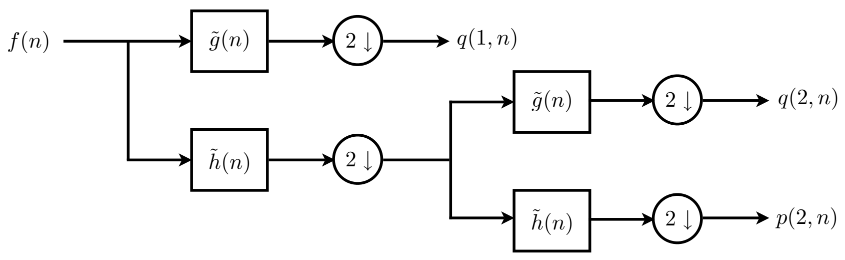


Figure 2.3 Two-level DWT analysis filterbank

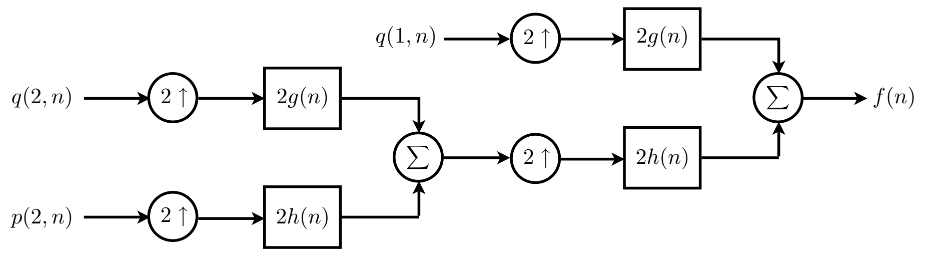


Figure 2.4 Two-level DWT synthesis filterbank

2.4. THE SHIFT-INVARIANT DWT

Unfortunately, for some purposes of signal analysis, the critically sampled DWT does not inherit the important property of shift-invariance. This means that for a given signature in a signal, the wavelet domain representation (approximation and detail coefficients) will not remain invariant to any shift of the signature. An example is shown in Figure 2.5. The signature in question is the rectangular pulse. The signal and its wavelet transform have been obtained (using the DB4 wavelet) and are shown on the left. The signal is then delayed by one sample in time and its wavelet transform coefficients are shown on the right. As can be seen in the figure, the representation is entirely different under the shift of the pulse. This difference is especially prominent in the detail coefficients q_1 and q_2.

In order to correct for the discrepancy in the wavelet transform of the two signals, the shift-invariant DWT, also called the stationary DWT, can be used to decompose the signal into its approximation and detail coefficients. Orthogonality of the detail and approximation coefficients is lost under this operation, but nevertheless it provides a good tool for analyzing signals at various scales. Shown in Figure 2.6 are the previous two signals and their respective wavelet coefficients obtained with the shift-invariant wavelet transform (using the same DB4 wavelet). The wavelet domain representation of the second signal is simply a single-sample translation of the first signal. Note from the figure that as compared to the critically sampled transform, where the number of wavelet coefficients is equal to the number of signal samples, the shift-invariant transform is highly redundant. Each level of representation has the same number of coefficients as the original signal.

This is due to the fact that the downsampling operation has been removed from the decomposition filterbank. This transform would not be appropriate solely for purposes of compression, but has certainly found applications in the area of signal denoising. For signal transmission purposes, redundancy can be partly mitigated via source coding techniques.

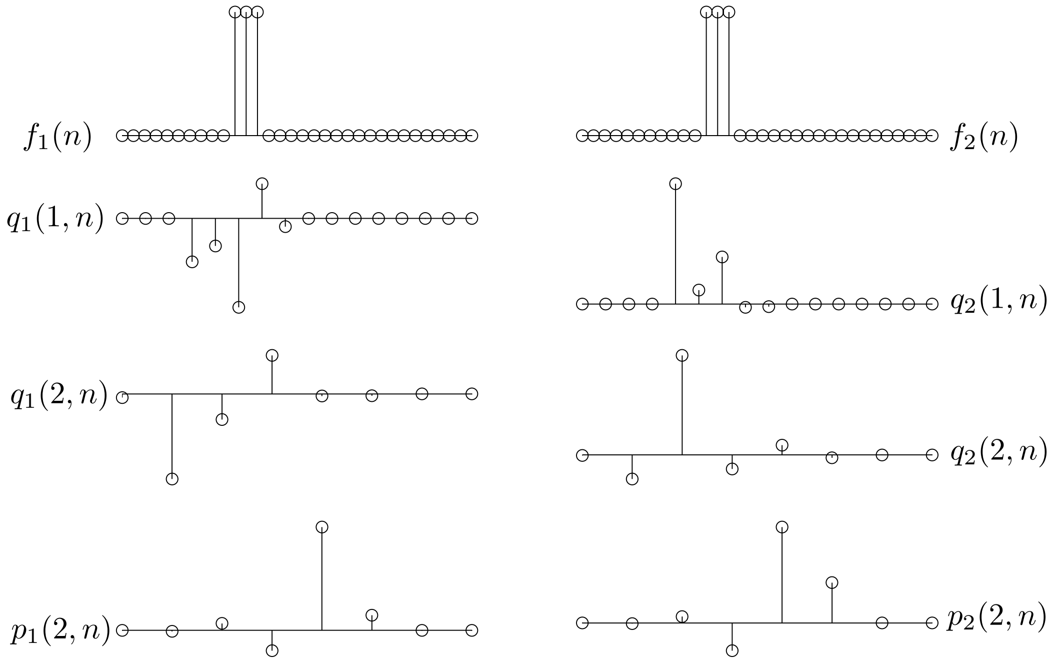


Figure 2.5 Two similar signals and their critically sampled DWTs

The shift-invariant wavelet transform coefficients are obtained in a similar fashion to that of the DWT, with some exceptions. As mentioned earlier, the downsampling operation is removed from the decomposition filterbank. If reconstruction is to be performed, the upsampling operations are removed from the reconstruction filterbank. Another major difference is that while the synthesis and analysis filters remain the same throughout the filterbanks for obtaining the DWT, they in fact change at every level of the filterbank for the shift-invariant transform. At

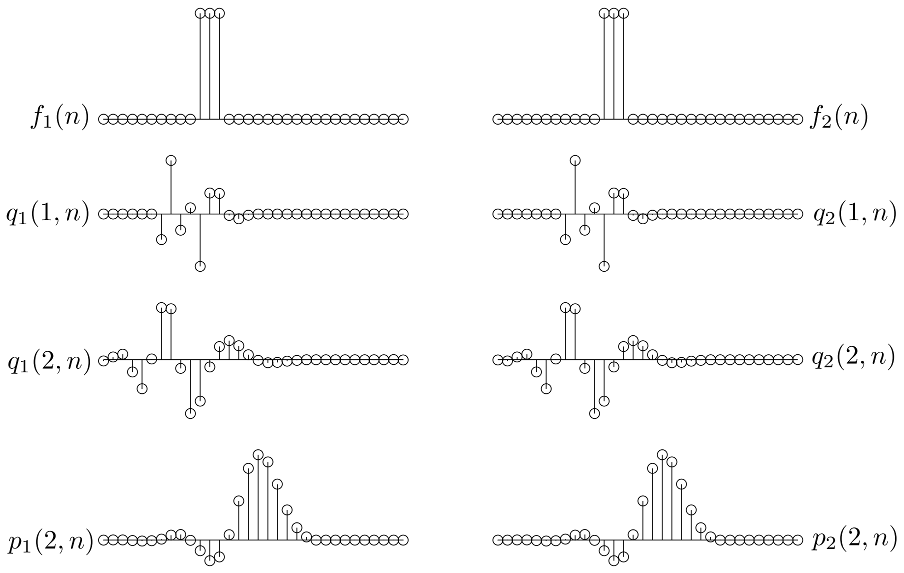


Figure 2.6 Two similar signals and their shift-invariant DWTs

each level, the filters are subjected to an upsampling operation. The decomposition paradigm is depicted in Figure 2.7.

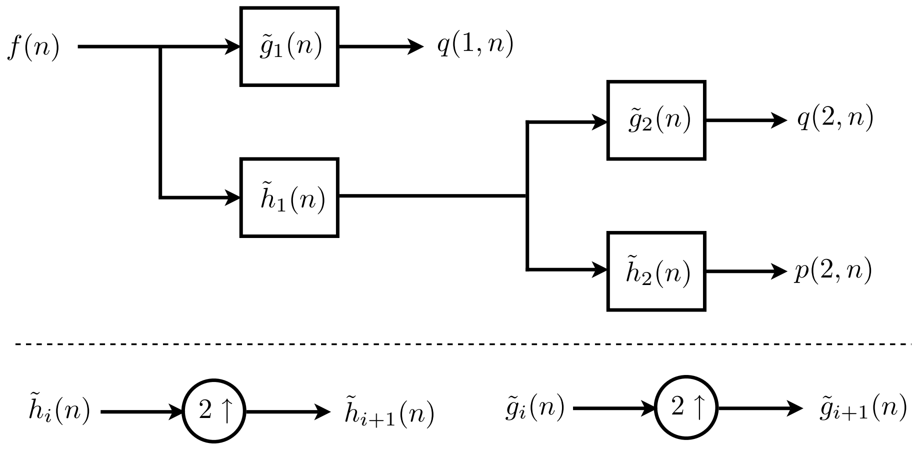


Figure 2.7 Decomposition filterbank for the shift-invariant wavelet transform

2.5. TWO-DIMENSIONAL ANALYSIS

The signals studied in this work are two-dimensional images, and the wavelet transform must be obtained for them. Fortunately, the two-dimensional wavelet transform is essentially a collection of one-dimensional transforms. A two dimensional scaling function and collection of two dimensional wavelets first need to be defined. A separable 2D scaling function is simply a product of two 1D scaling functions:

$$\phi(x,y) = \phi(x)\phi(y). \tag{2.36}$$

The 2D wavelets are constructed from the 1D scaling function and wavelet, i.e.

$$\psi_H(x,y) = \psi(x)\phi(y), \tag{2.37}$$

$$\psi_V(x,y) = \phi(x)\psi(y), \tag{2.38}$$

and

$$\psi_D(x,y) = \psi(x)\psi(y). \tag{2.39}$$

Some two-dimensional wavelets used for image analysis are not separable. Separability reduces computational complexity dramatically, since the two-dimensional transform is just a straightforward extension of the one-dimensional transform. The notation ψ_H, ψ_V, and ψ_D used above denotes the directional sensitivity of the wavelets. These wavelets measure functional variations–intensity or gray-level variations for images–along different directions. ψ_H measures variations along columns (e.g. horizontal edges), ψ_V measures variations along rows, and ψ_D corresponds to variations along diagonals. These scaling functions and wavelets are not actually used to derive LPF and HPF analysis and synthesis sequences, but instead they are the effective 2D wavelets that would be used if one chose not to use the fact that the wavelets are separable.

The analysis filterbank for decomposing the discrete image $f(m,n)$ is shown in Figure 2.8. Only one iteration of the decomposition is shown. Further levels of the decomposition filterbank would operate on the approximation coefficients $p(m,n)$. The 2D shift-invariant wavelet transform filterbank is similar to that shown above except for the absence of the downsamplers. Figure 2.9 shows a single-level

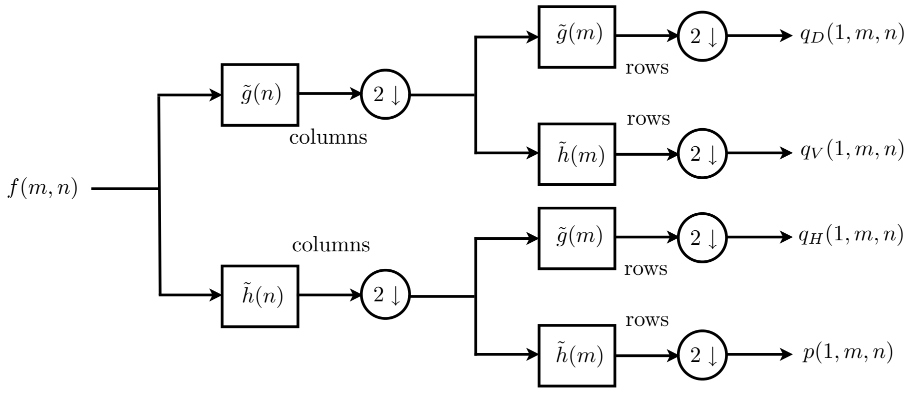


Figure 2.8 Decomposition filterbank for the two-dimensional DWT

decomposition filterbank. The filters would be upsampled at each iteration of the filterbank. An example of 2D critically sampled wavelet transform coefficients for a typical image is shown in Figure 2.10. The corresponding 2D stationary wavelet coefficients are shown in Figure 2.11

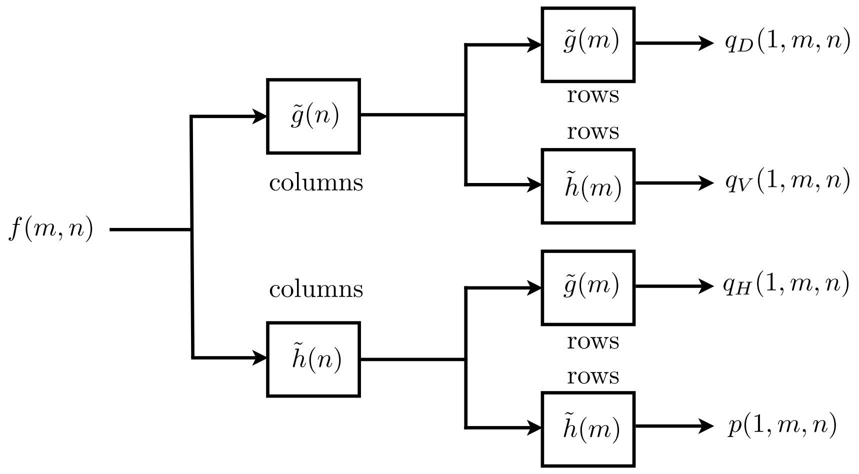
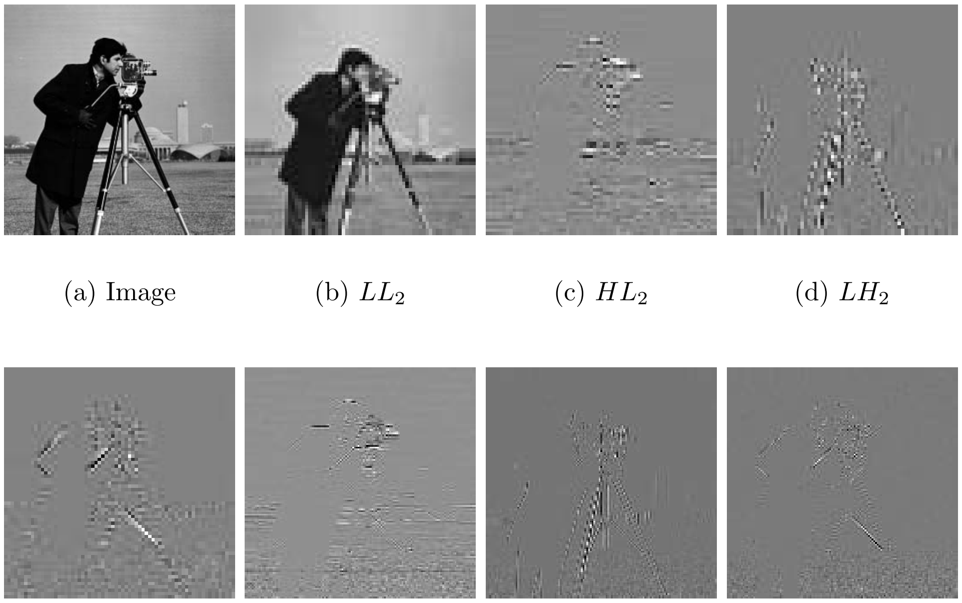
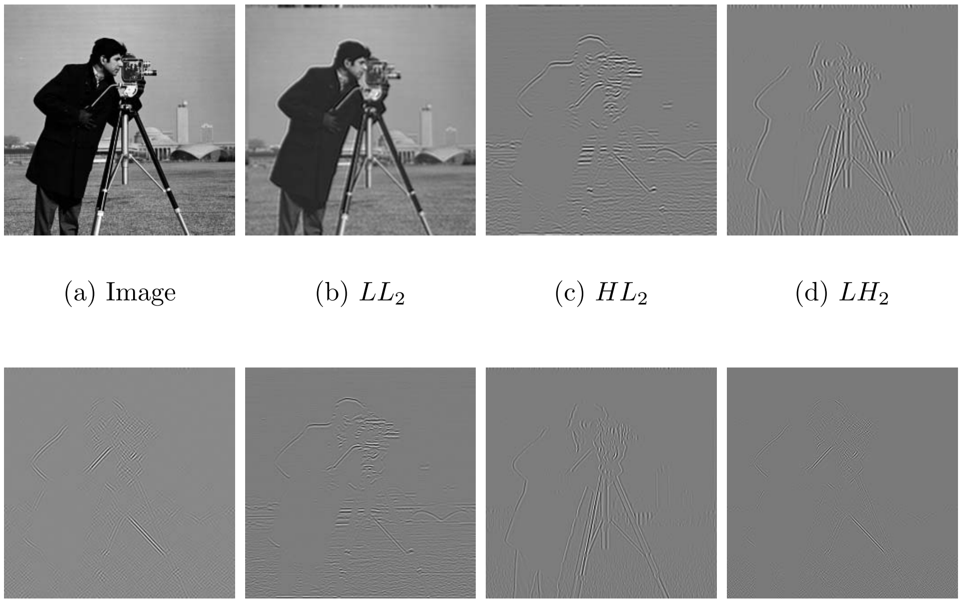


Figure 2.9 Decomposition filterbank for the two-dimensional shift-invariant DWT


(a) Image (b) LL_2 (c) HL_2 (d) LH_2

(e) HH_2 (f) HL_1 (g) LH_1 (h) HH_1

Figure 2.10 An image and its critically sampled wavelet transform coefficients


(a) Image (b) LL_2 (c) HL_2 (d) LH_2

(e) HH_2 (f) HL_1 (g) LH_1 (h) HH_1

Figure 2.11 An image and its stationary wavelet transform coefficients

3. ANOMALY DETECTION ALGORITHMS

Anomaly detection is one of the most critical components of any airborne landmine detection platform. Using wavelet transform data for target detection within digital imagery is a rather new and unexplored research area, and the literature is fairly sparse. Recent developments include those of Zhang and Desai [23], who have presented a target segmentation algorithm incorporating multiresolution analyses of images and their probability density functions (PDFs) with a Bayes classifier. Sadjadi [24] has developed a clustering algorithm to segment targets from clutter using a multiresolution texture-based methodology, again incorporating the PDFs of wavelet decomposition subbands.

The current baseline anomaly detector is the Reed-Xiaoli (RX) algorithm [25]. This section introduces the RX algorithm, and extends the multispectral RX algorithm to detect anomalies via the stationary wavelet decomposition of single-band imagery. Results for synthetic and real sensor data are presented and discussed. Another anomaly detection scheme using a multilayer feed-forward neural network and stationary wavelet coefficients is also presented.

3.1. THE REED-XIAOLI ALGORITHM

The RX algorithm is a constant false alarm rate (CFAR) detection algorithm which tests for the presence of a known optical or IR signal pattern which has unknown relative intensities within several signal-plus-noise channels [25]. The algorithm is derived as a generalized maximum likelihood ratio test (LRT) used for detecting anomalous image regions given a single-band or multiband image. The test assumes that statistical model for optical or IR clutter images follows a white Gaussian probability distribution function (PDF). It is also assumed that the local mean can change from region to region, while the covariance varies slowly across the image. This algorithm is the result of the previous work found in [26] and [27]. The development of the algorithm here follows the notation in [25].

3.1.1. Derivation of the RX Statistic. The first step of the RX algorithm is to remove the nonstationary local mean from each of the J bands of the multiband image. Each plane of the image X is convolved with an all ones window of size $L \times L$, denoted W. The local mean is

$$\bar{X} = \frac{1}{L^2}[X * W]. \tag{3.1}$$

The size of the window L can be chosen so that the third moment of the residual image $X_0 = X - \bar{X}$ is minimized. This results in an image with an approximately Gaussian PDF.

Next, let the column vectors $\mathbf{x}(n) = [x_1(n), x_2(n), \cdots, x_J(n)]^T$ for $n = 1, \ldots, N$ and $N > J$ represent the J correlated scenes (from the de-meaned image planes of X_0) which may or may not contain a target with known shape. The known signal pattern is represented as $\mathbf{S} = [s(1), s(2), \ldots, s(N)]$. Here, the two dimensional subimages representing $x_1, x_2, \ldots, x_J$ and $\mathbf{S}$ have been lexicographically ordered into one dimensional row vectors. A signal intensity vector is then defined as $\mathbf{b} = [b_1, b_2, \ldots, b_J]^T$. This is the vector of unknown signal (target) intensities in each of the J bands.

The LRT must distinguish between the following binary hypotheses:

$$\begin{aligned} H_0 &: \mathbf{X} = \mathbf{X}_0 \\ H_1 &: \mathbf{X} = \mathbf{X}_0 + \mathbf{bS}, \end{aligned} \tag{3.2}$$

where $\mathbf{x}_0$ is the vector of the residual clutter process, which is assumed to be approximately independent and Gaussian from pixel to pixel. This assumption arises from the fact that many IR images, after subtraction of the local nonstationary mean, have covariance matrices which can be approximated by diagonal matrices. The covariance matrix at a pixel location (m, n) in the residual image X_0 can be estimated from the subimage surrounding the pixel location with

$$\mathbf{M} = \mathrm{E}\left\{[\mathbf{x}(n) - \mathrm{E}\mathbf{x}(n)]\,[\mathbf{x}(n) - \mathrm{E}\mathbf{x}(n)]^T\right\} . \tag{3.3}$$

The parameter space for this LRT is defined as

$$\Omega \triangleq \{[\mathbf{b}, \mathbf{M}] : |\mathbf{M}| > 0\} . \tag{3.4}$$

Here it is assumed that all covariance matrices encountered are positive definite. The likelihood function is given by

$$L(\mathbf{b}, \mathbf{M}) = \frac{1}{2\pi^{NJ/2}|\mathbf{M}|^{N/2}} \exp\left[-\frac{1}{2}\sum_{n=1}^{N} \left(\mathbf{x}(n) - \mathrm{E}\mathbf{x}(n)\right)^T \mathbf{M}^{-1} \left(\mathbf{x}(n) - \mathrm{E}\mathbf{x}(n)\right)\right]. \tag{3.5}$$

Letting $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)]$, a $J \times N$ matrix of the image data, the likelihood can be written as

$$L(\mathbf{b}, \mathbf{M}) = \frac{1}{2\pi^{NJ/2}|\mathbf{M}|^{N/2}} \exp\left\{-\frac{1}{2} \operatorname{Tr}\left[\mathbf{M}^{-1} \left(\mathbf{X} - \mathrm{E}\mathbf{X}\right)\left(\mathbf{X} - \mathrm{E}\mathbf{X}\right)^T\right]\right\}. \tag{3.6}$$

The parameter space is divided into two complimentary spaces, ω and $\Omega \setminus \omega$, and the hypotheses are defined as follows

$$\begin{aligned} H_0 &\equiv [\mathbf{b}, \mathbf{M}] \in \omega \\ H_1 &\equiv [\mathbf{b}, \mathbf{M}] \in \Omega \setminus \omega \end{aligned} \tag{3.7}$$

where

$$\begin{aligned} \omega &= \{[\mathbf{0}, \mathbf{M}] : |\mathbf{M}| > 0\} \\ \Omega \setminus \omega &= \{[\mathbf{b}, \mathbf{M}] : |\mathbf{M}| > 0, \mathbf{b} \neq \mathbf{0}\}. \end{aligned}$$

The null hypothesis H_0 corresponds to no target being present in the given subimage, while H_1 implies the presence of a target. The maximum likelihood ratio test is then given by

$$\Lambda(\mathbf{X}) = \frac{\max\limits_{\mathbf{b},\mathbf{M}\in\Omega\setminus\omega} L(\mathbf{b}, \mathbf{M})}{\max\limits_{\mathbf{b},\mathbf{M}\in\omega} L(\mathbf{b}, \mathbf{M})} \underset{H_0}{\overset{H_1}{\gtrless}} k\ . \tag{3.8}$$

The maxima of both likelihood functions under each hypothesis are computed as

$$\max_{\mathbf{b},\mathbf{M}\in\Omega\setminus\omega} L(\mathbf{b}, \mathbf{M}) = \frac{1}{2\pi^{NJ/2}|\hat{\mathbf{M}}_{\mathbf{b}}|^{N/2}} \exp\left(\frac{-NJ}{2}\right) \tag{3.9}$$

and

$$\max_{\mathbf{b},\mathbf{M}\in\omega} L(\mathbf{b}, \mathbf{M}) = \frac{1}{2\pi^{NJ/2}|\hat{\mathbf{M}}_{\mathbf{0}}|^{N/2}} \exp\left(\frac{-NJ}{2}\right). \tag{3.10}$$

The maximum likelihood estimates (MLE's) of the unknown parameters under each hypothesis are

$$\hat{\mathbf{M}}_{\mathbf{0}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}(n)\mathbf{x}^T(n) = \frac{1}{N}\mathbf{X}\mathbf{X}^T \tag{3.11}$$

and

$$\hat{\mathbf{M}}_{\mathbf{b}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_{\mathbf{b}}(n){\mathbf{x}_{\mathbf{b}}}^T(n) = \frac{1}{N}(\mathbf{X}-\hat{\mathbf{b}}\mathbf{S})(\mathbf{X}-\hat{\mathbf{b}}\mathbf{S})^T, \tag{3.12}$$

where

$$\hat{\mathbf{b}} = \frac{\mathbf{X}\mathbf{S}^T}{\mathbf{S}\mathbf{S}^T} \ . \tag{3.13}$$

Note that $\mathbf{S}\mathbf{S}^T$ is a scalar. Substituting Eqs. 3.9 and 3.10 into Eq. 3.8 yields the LRT

$$\Lambda(\mathbf{X}) = \frac{|\hat{\mathbf{M}}_{\mathbf{0}}|^{N/2}}{|\hat{\mathbf{M}}_{\mathbf{b}}|^{N/2}} \ \underset{H_0}{\overset{H_1}{\gtrless}} \ k \ . \tag{3.14}$$

An equivalent test can be written as

$$\lambda(\mathbf{X}) = \frac{|\hat{\mathbf{M}}_{\mathbf{0}}|}{|\hat{\mathbf{M}}_{\mathbf{b}}|} \ \underset{H_0}{\overset{H_1}{\gtrless}} \ c \ , \tag{3.15}$$

where $c = k^{2/N}$. Substituting Eqs. 3.11 and 3.12 into Eq. 3.15 and simplifying yields the explicit test

$$\lambda(\mathbf{X}) = \frac{|\mathbf{X}\mathbf{X}^T|}{\left|\mathbf{X}\mathbf{X}^T - \dfrac{(\mathbf{X}\mathbf{S}^T)(\mathbf{X}\mathbf{S}^T)^T}{\mathbf{S}\mathbf{S}^T}\right|} \ \underset{H_0}{\overset{H_1}{\gtrless}} \ c \ , \tag{3.16}$$

where the likelihood ratio can be further simplified to

$$\lambda(\mathbf{X}) = \frac{1}{1 - \dfrac{(\mathbf{X}\mathbf{S}^T)^T(\mathbf{X}\mathbf{X}^T)^{-1}(\mathbf{X}\mathbf{S}^T)}{\mathbf{S}\mathbf{S}^T}} \ ,$$

finally producing the equivalent test

$$r(\mathbf{X}) = \frac{(\mathbf{X}\mathbf{S}^T)^T(\mathbf{X}\mathbf{X}^T)^{-1}(\mathbf{X}\mathbf{S}^T)}{\mathbf{S}\mathbf{S}^T} \underset{H_0}{\overset{H_1}{\gtrless}} r_0 \ . \tag{3.17}$$

The derivation of the distribution of the statistic under the null and non-null hypothesis is not outlined here, but can be found in [25]. The generalized signal-to-noise ratio (GSNR), needed to characterize the distributions, is defined as

$$\text{GSNR} = a \triangleq (\mathbf{b}^T\mathbf{M}^{-1}\mathbf{b})\|\mathbf{S}\|^2 \ . \tag{3.18}$$

The PDF of the test statistic r under the non-null hypothesis is given by the non-central F distribution

$$f(r|H_1) = \frac{\Gamma\left(\frac{N}{2}\right)}{\Gamma\left(\frac{N-J}{2}\right)\Gamma\left(\frac{J}{2}\right)}(1-r)^{\frac{N-J-2}{2}} r^{\frac{J-2}{2}} e^{-\frac{a}{2}} {}_1F_1\left(\frac{N}{2};\frac{J}{2};\frac{ar}{2}\right) \ , \tag{3.19}$$

where $0 < r < 1$ and ${}_1F_1(a;b;x)$ is the confluent hypergeometric function. Under the null hypothesis no signal is present ($a = 0$), and the PDF reduces to the standard beta distribution

$$f(r|H_0) = \frac{\Gamma\left(\frac{N}{2}\right)}{\Gamma\left(\frac{N-J}{2}\right)\Gamma\left(\frac{J}{2}\right)}(1-r)^{\frac{N-J-2}{2}} r^{\frac{J-2}{2}} \ , \tag{3.20}$$

again where $0 < r < 1$. The receiver operating characteristic (ROC) can be computed from these PDF's with the following equations:

$$P_{FA} = \int_{r_0}^{1} f(r|H_0)dr \ ; \quad P_D = \int_{r_0}^{1} f(r|H_1)dr, \tag{3.21}$$

where P_{FA} is the probability of a false alarm and P_D is the probability of detection.

3.1.2. RX Algorithm Implementation. The RX algorithm requires that the detection statistic in Eq. 3.17 be calculated for each pixel location in the original single or multi-band image. The algorithm can be implemented either in the spatial domain via 2-D convolution, or in the frequency domain with the 2-D fast Fourier transform (FFT). Each method has its own advantages and disadvantages. Regardless of the method chosen, the inverse of a $J \times J$ covariance matrix must be computed at each pixel location. Clearly this is a simple task for single band imagery, but computational complexity dramatically increases as J grows larger, making the RX algorithm highly inappropriate for detecting anomalies in hyperspectral imagery where it is not uncommon to encounter $J > 100$ [28].

Figures 3.1 and 3.2 show two possible sets of masks to use in calculating the target mean ($\hat{\mathbf{b}}$) and covariance ($\hat{\mathbf{M}}$) data. Figure 3.1 shows a set of square masks where the central shaded square (W_T) is the target mask, and the outer shaded "square annulus" (W_C) is the covariance mask. The white space between the masks represents a "blanking" region, which does not contribute to the calculation of the statistic. This region can contain shadows or other reflective effects of the target signature, which can corrupt the target mean and/or covariance estimates [29]. Each non-zero weight in each mask has a value of unity. The square masks with constant weight are very efficient for estimating the target mean and covariance data via convolution. Since recursive implementation of the convolution sum is possible, this obviates the need to repeatedly compute the sum of pixel values for the inner regions of the mask as it is convolved with the image. Only the values on the edges of the mask need to be considered and added or subtracted as needed, drastically reducing computational cost per pixel. The final convolution result is divided by the number of the non-zero pixel values in the mask, thus the effective mask has constant pixel values that sum to unity.

Estimation of target mean and covariance data with RX masks having more complex geometries usually requires the use of fast Fourier transform (FFT) based convolution, since computational complexity becomes too high with brute-force convolution. The masks shown in Figure 3.2 have the same attributes as the square masks, except for their geometry. There are three radii noted in the figure. R_T is the radius of the target mask used to estimate the target mean. R_B is the blanking

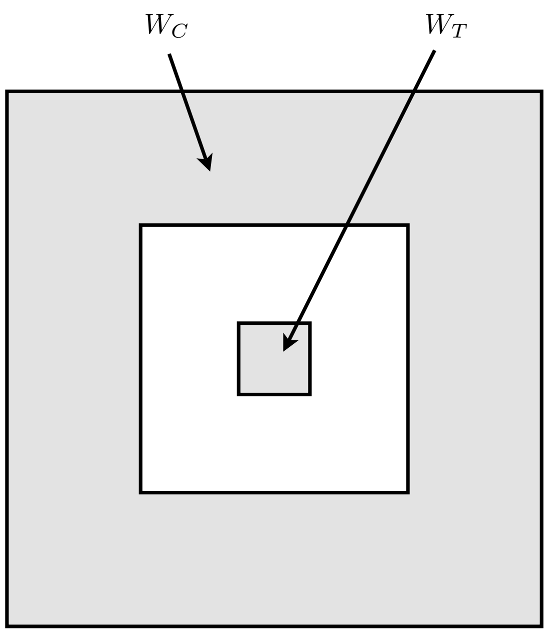


Figure 3.1 Square RX masks for calculation of target mean and covariance data

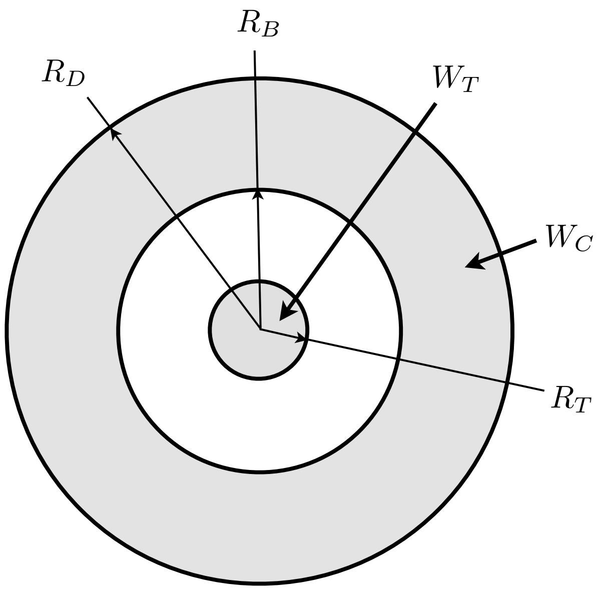


Figure 3.2 Circular RX masks for calculation of target mean and covariance data

radius used to define the area of the inner circle excluded from the covariance mask. R_D is the demeaning radius, which helps to define two masks: the mask used to demean each band of the original image, and the covariance mask. The covariance mask has a radial width of $R_D - R_B$. Circular masks are adept at producing well clustered RX statistics for high-contrast circular regions such as landmines, and the detection statistic is invariant to rotation; however, the complex geometry of circular masks calls for use of the FFT to calculate the mean and covariance data.

Shown in Figure 3.3 is a representative single-band image of a portion of a minefield. The output of the RX algorithm is shown in Figure 3.4 where circular masks have been used with parameters $R_T = 2$, $R_D = 20$ and $R_B = 7$. After an image has been processed with the RX algorithm, the output is then subjected to non-maximal suppression. This is an algorithm where pixels values that do not

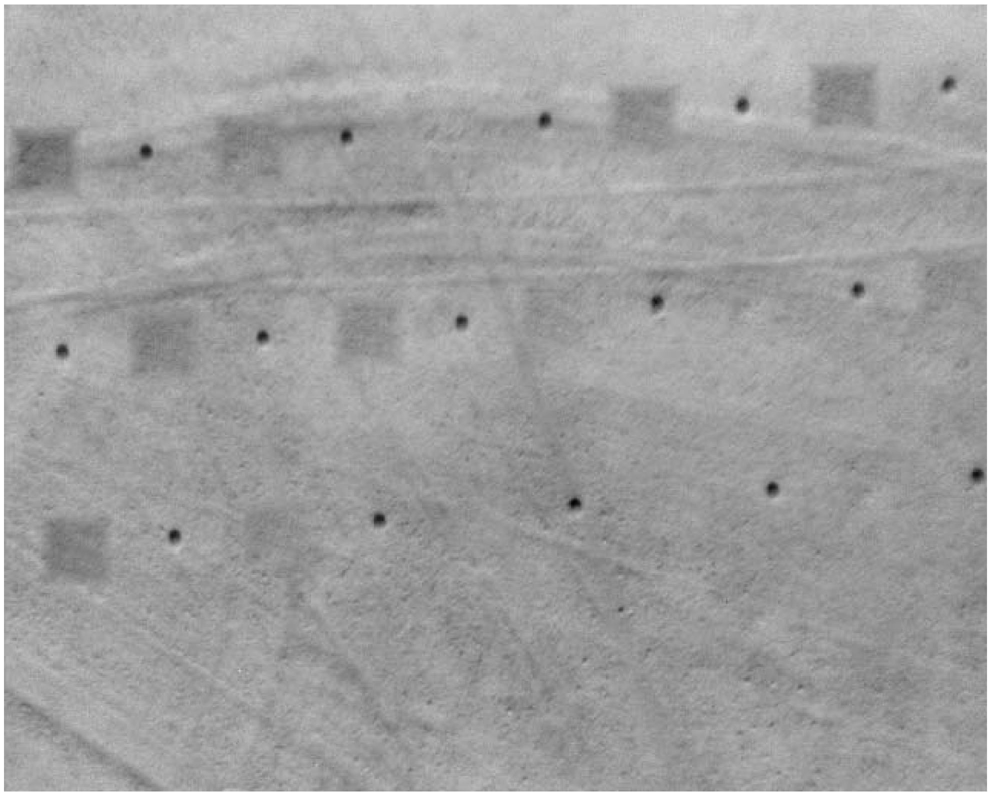


Figure 3.3 A representative LAMD single-band image with portion of patterned minefield

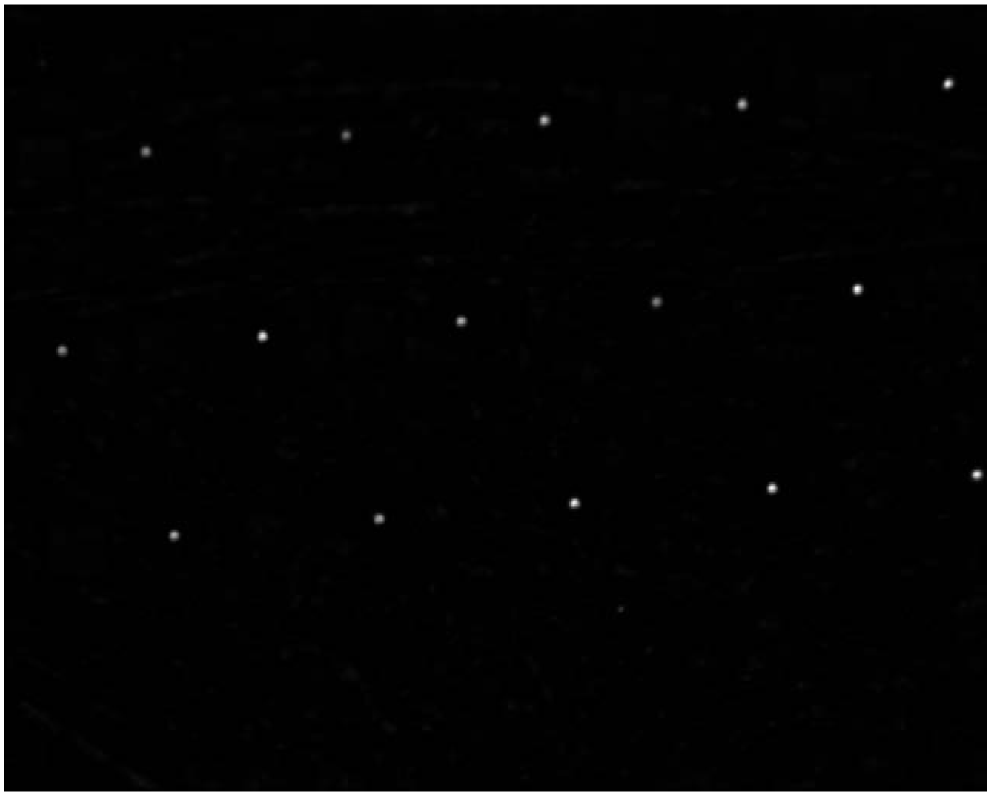


Figure 3.4 Output of the RX algorithm for image in Figure 3.3

exceed a certain threshold are first set to zero. The remaining pixels are then pruned around local maxima until a singlular value remains. This value is then used to determine the location and amplitude of the RX output for each anomaly. The distributions of the detection statistics after non-maximal supression of the RX output has been studied in [30]. Non-maximal suppression is also used for anomaly localization in the following algorithms.

3.2. A MULTISCALE RX-BASED ALGORITHM

The multiband RX algorithm is extended here to detect anomalies within the wavelet coefficients of single-band imagery. The work presented here shows that the detection performance of this algorithm is higher than that of the single-band RX algorithm for anomalies in a correlated noise background. Better detection performance is thought to occur for two reasons: the nature of the wavelet transform to decorrelate signals, and the improvement in the theoretical ROC curves of the RX algorithm when operating under multiple bands (multiple scales in this case).

Reed *et al.*, [31] have proposed a multi-spectral anomaly detector using the wavelet transform, although the framework under which it is developed is quite different than the multiscale RX-based algorithm discussed below. Whereas Reed develops a generalized likelihood ratio test which operates only on one wavelet subband at a time, the test developed below operates on all of the wavelet subband data available to it at once. Reed's previous work addresses the use of critically sampled transform coefficients on which to perform detection. Here, a stationary wavelet transform is used to obtain the wavelet coefficients used for processing. This allows for shift-invariance in the wavelet-domain representation due to the redundancy of the coefficients caused by the absence of downsamplers in the decomposition filterbank. The wavelet used in most of these simulations is the Haar wavelet, since it introduces the least amount of translation in the coefficients. Other wavelets can be used, but those with a long support should be avoided to keep translation to a minimum. The development of the algorithm follows the same reasoning as the RX algorithm that was discussed in the previous section, however due to the nature of the wavelet coefficients the mathematical derivation is slightly different.

3.2.1. Derivation of the Test Statistic. The detector must distinguish between the following hypotheses:

$$\begin{aligned} H_0 &: \mathbf{X} = \mathbf{X}_0 \\ H_1 &: \mathbf{X} = \mathbf{X}_0 + \mathbf{BS}, \end{aligned} \tag{3.22}$$

where $\mathbf{X}$ represents the lexicographically ordered wavelet coefficients of different subbands. $\mathbf{X}$ is of size $J \times N$ where there are J wavelet decomposition subbands upon which the test operates, and N pixel locations within the target window. The MLE of the covariance matrix $\hat{\mathbf{M}}_\mathbf{0}$ under the null hypothesis is the same as that given in Eq. 3.11, while the MLE of the covariance matrix $\hat{\mathbf{M}}_\mathbf{B}$ under the non-null hypothesis is

$$\hat{\mathbf{M}}_\mathbf{B} = \frac{1}{N}(\mathbf{X} - \mathbf{BS})(\mathbf{X} - \mathbf{BS})^T, \tag{3.23}$$

where the target mean matrix $\mathbf{B}$ is given by

$$\mathbf{B} = (\mathbf{X}\mathbf{S}^T)(\mathbf{S}\mathbf{S}^T)^{-1}. \tag{3.24}$$

$\mathbf{B}$ can be thought of as a mixing matrix, or correlation matrix. In contrast to the previous formulation of the RX maximum likelihood estimates, $\mathbf{B}$ is a $J \times J$ matrix instead of a J-length column vector. $\mathbf{S}$ is a $J \times N$ template matrix, in contrast to an N-length row vector as previously defined. $\mathbf{S}$ must take into account that a target signature is decorrelated throughout the subbands, and cannot take on a single representation as in the RX algorithm. One possible mechanism to calculate the template signature $\mathbf{S}$ is discussed in section 3.2.2.

The likelihood ratio test remains the same as developed in the previous section as far as it is the ratio of the determinants of the covariance estimates under each hypothesis:

$$\lambda(\mathbf{X}) = \frac{|\hat{\mathbf{M}}_{\mathbf{0}}|}{|\hat{\mathbf{M}}_{\mathbf{B}}|} \underset{H_0}{\overset{H_1}{\gtrless}} c. \tag{3.25}$$

This maintains the interpretation that the likelihood ratio $\lambda(\mathbf{X})$ is the ratio of the hypervolumes of the parallelepipeds spanned by the column vectors of the covariance matrices under each hypothesis. Substituting Eq. 3.23 into Eq. 3.22 and expanding yields

$$\begin{aligned} N\hat{\mathbf{M}}_{\mathbf{B}} \;=\;& \mathbf{X}\mathbf{X}^T - \mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{S}\mathbf{X}^T - \mathbf{X}\big[\mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{S}\big]^T \\ & + \mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{S}\big[\mathbf{X}\mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{S}\big]^T. \end{aligned}$$

Under the properties of the matrix transpose and using the fact that the inverse of the symmetric matrix $\mathbf{S}\mathbf{S}^T$ is symmetric, the covariance matrix under H_1 can be simplified to

$$N\hat{\mathbf{M}}_{\mathbf{B}} = \mathbf{X}\mathbf{X}^T - (\mathbf{X}\mathbf{S}^T)(\mathbf{S}\mathbf{S}^T)^{-1}(\mathbf{X}\mathbf{S}^T)^T. \tag{3.26}$$

Substituting Eqs. 3.11 and 3.25 into Eq. 3.24 gives the explicit test

$$\lambda(\mathbf{X}) = \frac{\left|\mathbf{X}\mathbf{X}^T\right|}{\left|\mathbf{X}\mathbf{X}^T - (\mathbf{X}\mathbf{S}^T)(\mathbf{S}\mathbf{S}^T)^{-1}(\mathbf{X}\mathbf{S}^T)^T\right|} \underset{H_0}{\overset{H_1}{\gtrless}} c. \tag{3.27}$$

3.2.2. Target and Covariance Masks. Since target signatures are decorrelated throughout the wavelet subbands, the use of a single target mask is not appropriate for calculation of the target mean. A set of target masks is calculated for the expected wavelet representation of the anomaly at each scale and orientation. Described here is the methodology for choosing the target template matrix $\mathbf{S}$ used to calculate the test statistic.

In the spatial domain the target shape is assumed to be circular, but may lie in different poses containing various subpixel shifts as shown in Figure 3.5. The wavelet decomposition of the target in each pose is obtained, and then the coefficients are averaged over the different possible poses. This gives an expected target mask for each scale and orientation. The expected wavelet-domain masks for a single-level Haar decomposition of target signatures under 9 poses with $R_T = 3$ is shown in Figure 3.6.

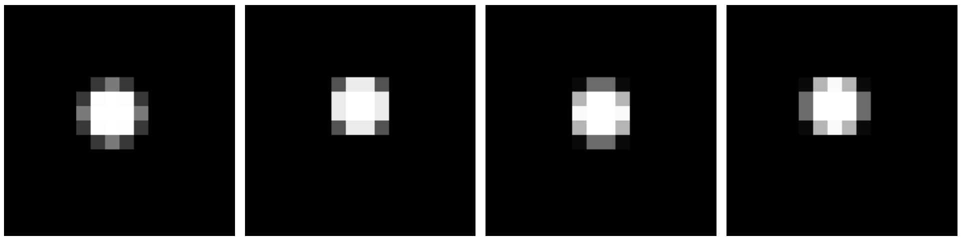


Figure 3.5 A target with $R_T = 2$ in with various subpixel shifted positions

The covariance mask that produces the best detection performance was found to be a simple square mask, in contrast to the annular masks discussed in the

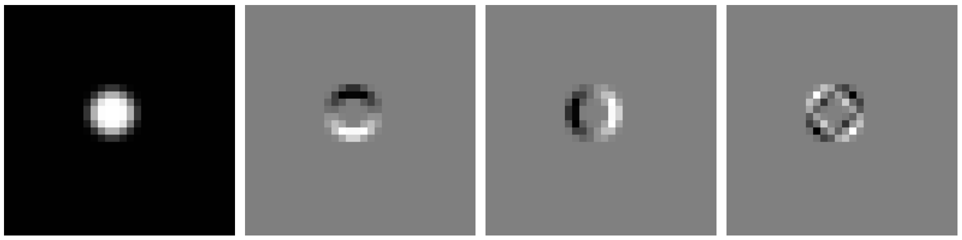


Figure 3.6 Expected wavelet-domain target masks over 9 poses of a target with $R_T = 3$

previous section. It is the author's opinion that a square mask inclusive of the target produces the best output since the determinant of the covariance under H_1 can be thought of as a metric which is a measure of the distance between the estimated covariance and the image data under the influence of the target mask matrix $\mathbf{S}$. When the distance is relatively small, the estimated covariance is close to the covariance of the target mask, and thus the eigenvalues of $\hat{\mathbf{M}}_\mathbf{B}$ are small with respect to those of $\hat{\mathbf{M}}_\mathbf{0}$ which in turn yields a high likelihood ratio $\lambda(\mathbf{X})$.

3.2.3. Algorithm Implementation. The multiscale RX-based algorithm is implemented in the frequency domain via the FFT, since the masks used to compute the target mean have a rather complex geometry. The algorithm is outlined below:

1. The first step of the algorithm is to precalculate the target masks $\mathbf{S}$ for a given target radius as outlined above. The Fourier transforms of the masks are calculated and labeled $\hat{s}_i(u, v)$. The inverse covariance matrix $(\mathbf{S}\mathbf{S}^T)^{-1}$ of the target masks is precalculated as well.
2. The second step in the algorithm is to obtain the stationary wavelet coefficients for the given image. A single level decomposition is used to obtain the coefficients. Two-level and three-level decompositions can be used, but contribute to higher computational complexity

with only slight performance gains. The Haar wavelet filters have so far provided the best detection results. The coefficients can be labeled $\mathbf{X}$ where $X_i(x,y)$ is the ith subband of the coefficient data.

3. Once the decomposition has been obtained, the approximation subband undergoes removal of its local mean using a circular window of a size 20 pixels. Since the detail subbands are zero-mean, no demeaning of them is necessary. The demeaned output is

$$X'_A(x,y) = X_A(x,y) - \mathcal{F}^{-1}\{\mathcal{F}\{X_A(x,y)\}\hat{d}(u,v)\}, \tag{3.28}$$

where $\hat{d}(u,v)$ is the Fourier transform of the circular window.

4. Local covariance estimates are then computed for the wavelet subband data. The covariance mask used for this purpose has rectangular geometry. The dimensions used here are 32×32. The covariance images are calculated as

$$M_{ij}(x,y) = \mathcal{F}^{-1}\{\mathcal{F}\{X_i(x,y)X_j(x,y)\}\hat{m}(u,v)\}, \tag{3.29}$$

where $\hat{m}(u,v)$ is the covariance mask in the frequency domain. $M_{ij}(x,y)$ is the ijth entry of the covariance matrix $\mathbf{M}$ at pixel location (x,y).

5. Target-mean data is computed for each subband via correlation with the target masks:

$$T_{ij}(x,y) = \mathcal{F}^{-1}\{\mathcal{F}\{X_i(x,y)\}\hat{s}_j^*(u,v)\}, \tag{3.30}$$

where T_{ij} is the ijth entry of the matrix $\mathbf{T} = \mathbf{X}\mathbf{S}^T$.

6. An output statistic λ is then calculated for each pixel location in the original input image with the equation

$$\lambda = \frac{|\mathbf{M}|}{|\mathbf{M} - \mathbf{T}(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{T}^T|}. \tag{3.31}$$

7. The output image $\lambda(x, y)$ is subjected to thresholding and non-maximal suppression. The remaining pixel locations and values are put into a list of target locations and ordered by likelihood.

3.2.4. Results for Various Imagery. The above algorithm has been tested on both synthetic and real sensor data. The synthetic imagery allows for a straightforward comparison of the multiscale algorithm with the RX algorithm. Each synthetic image is of size 128×128, has a noise variance of $\sigma^2 = 25$ pixels, and has targets ($R_T = 1$) populated throughout with random amplitudes each with a possible SNR of 2, 5, 10, 15, and 20. The SNR here is the ratio of target amplitude to noise amplitude. The background has been subjected to a lowpass 3×3 box filter in order to correlate the noise. Two typical test images are shown in Figure 3.7. The overall results of the multiscale algorithm and RX algorithm are shown in Figure 3.8. Results for various SNRs are shown in Figures 3.9, 3.10, 3.11, and 3.12. The algorithm has been tested with typical LAMD minefield dataset as well. The results are shown in Figure 3.13.

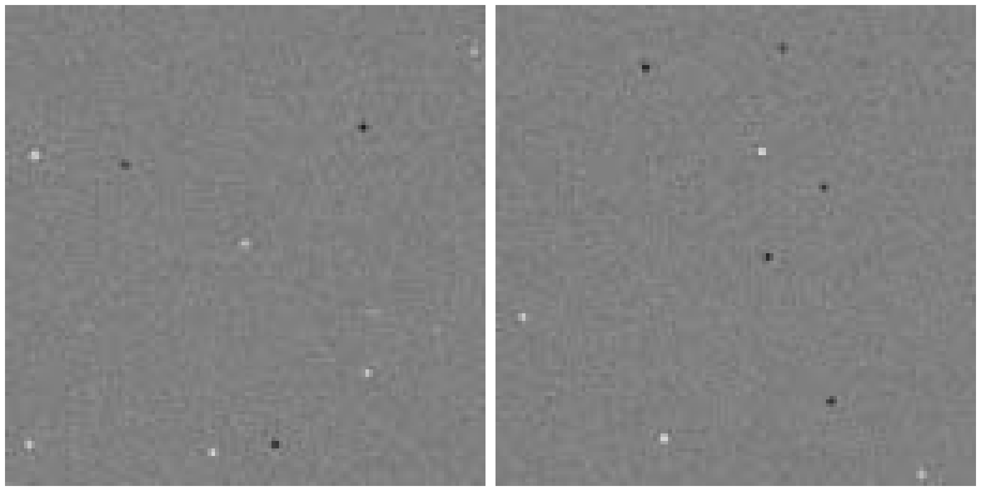


Figure 3.7 Two typical test images used for testing of the multiscale RX algorithm

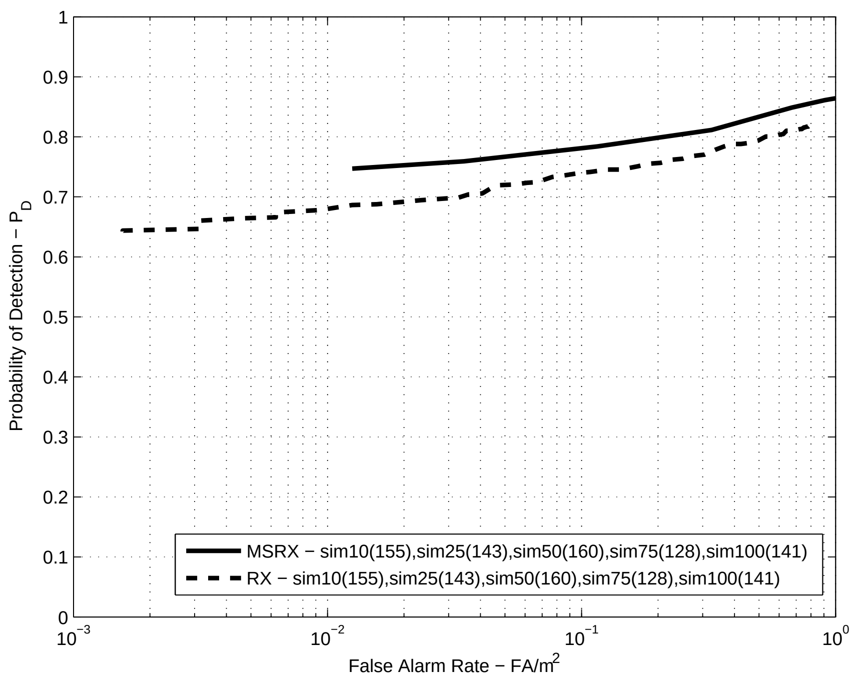


Figure 3.8 Overall performance of the multiscale algorithm for all SNRs

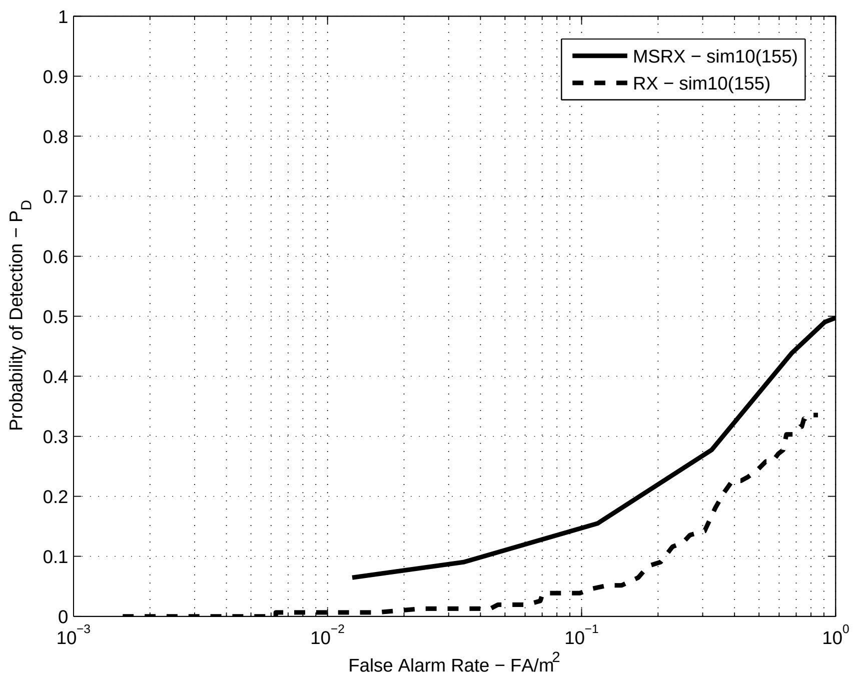


Figure 3.9 Performance of the multiscale algorithm for SNR = 2

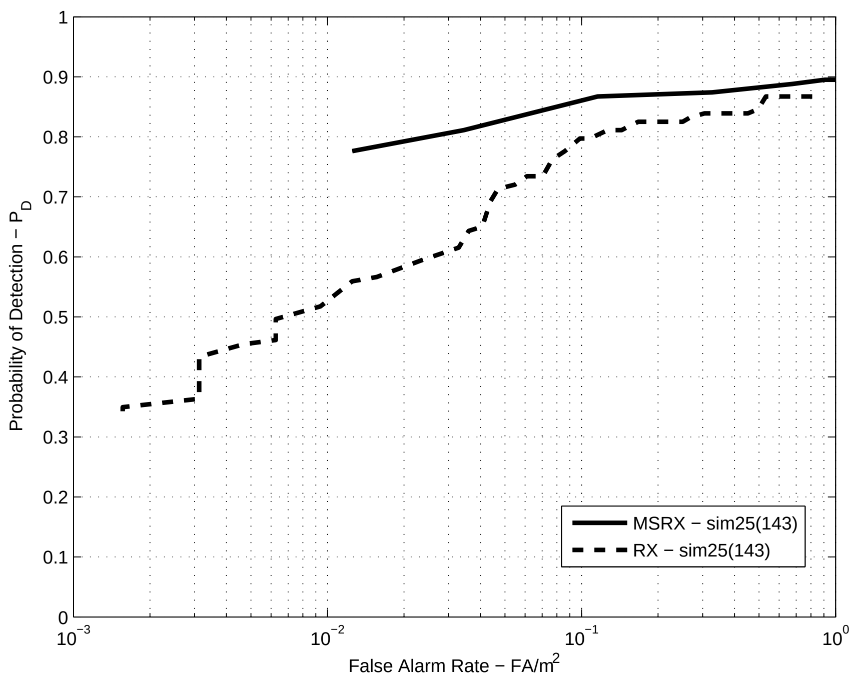


Figure 3.10 Performance of the multiscale algorithm for SNR = 5

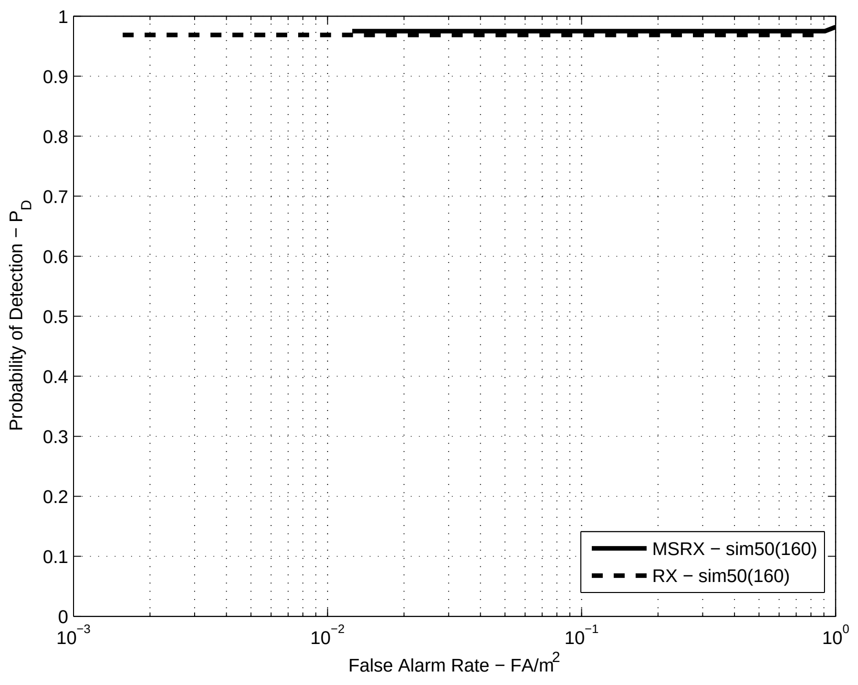


Figure 3.11 Performance of the multiscale algorithm for SNR = 10

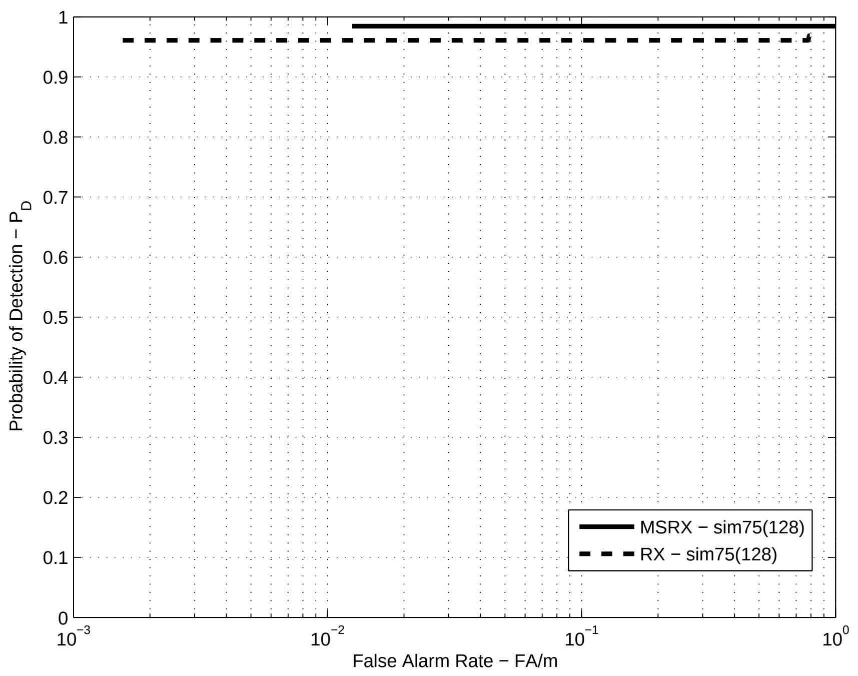


Figure 3.12 Performance of the multiscale algorithm for SNR = 15

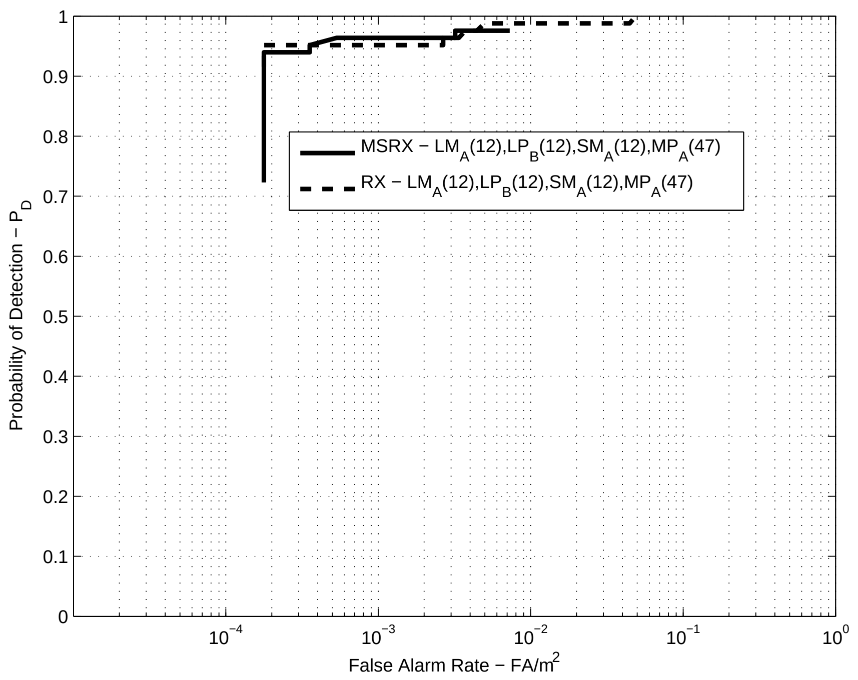


Figure 3.13 Performance of the multiscale algorithm for a typical minefield dataset

For the synthetic imagery, the multiscale algorithm outperforms RX when considering all targets of various SNRs. The detection performance for targets with high SNRs is similar to the RX algorithm, as expected. The clearest distinction in performance is at low SNRs and low false alarm rates where the multiscale algorithm clearly gives better detection results. For the LAMD sensor data, the multiscale algorithm and RX performance are similar. At the desired false alarm rate of 10^{-3} false alarms per meter^2 (approximately one false alarm per four frames), the multiscale algorithm offers a slight improvement in detection performance over the RX algorithm; however, the LAMD data evaluated in this study clearly has a high SNR (as seen in Figure 3.3) and hence the performance of the multiscale algorithm and RX is very similar.

3.3. NEURAL NETWORK DETECTION ALGORITHM

Developed here is an anomaly detection algorithm based on stationary wavelet coefficients and a multilayer perceptron feed-forward neural network (MLPNN). The goal of investigating this architecture was to observe whether a simple learning algorithm could adapt to wavelet coefficient data as its input and provide a probabilistic measure of the presence or absence of an anomaly at its output. The successfulness of the network in detecting known anomalies via wavelet coefficient data provides insight into the feasibility of performing wavelet-based detection with other architectures. The network presented below is not to be confused with the wavelet neural networks that have been proposed in the literature. These networks use wavelets as their basis functions and are particularly suited for prediction of time-series [32].

3.3.1. Test Imagery and Wavelet Selection. The synthetic images used in this investigation are simplistic in nature so as to analyze baseline results. A representative synthetic image is shown in Figure 3.14(a). The image has a dynamic range of $[0, 255]$, and contains white Gaussian noise with $\mu = 128$ and $\sigma^2 = 25$. Targets are placed in the image randomly according to a uniform distribution, while making sure they do not fall within a certain proximity to another target or on the edge of the image. The targets have a radius of one pixel, and are subjected to random sub-pixel shifts. Targets also take on random amplitudes which fall both above and below the level of the Gaussian noise. The images are similar to those tested with the multiscale RX-based algorithm, but here the noise is not correlated. When an image is created, its ground truth is recorded and from this a desired output image is constructed. The input imagery are of size 128×128, and the corresponding desired output imagery is 64×64. The desired output values are either one, representing the presence of a target, or zero, depicting its absence. The desired output image for the synthetic image in Figure 3.14(a) is shown in Figure 3.14(b).

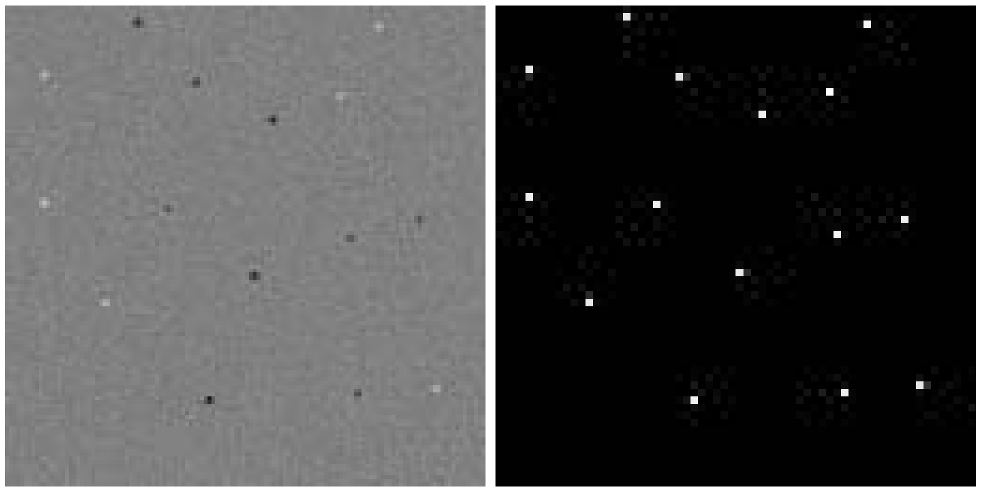
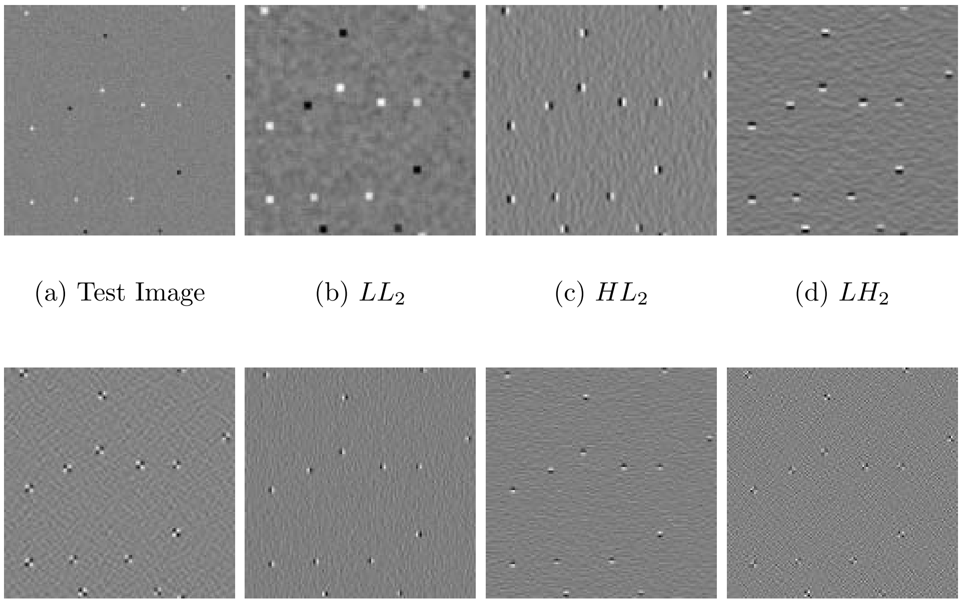


Figure 3.14 A test image and the desired output created from its ground truth

As in the multiscale RX-based algorithm, a Haar wavelet is utilized for decomposing the synthetic imagery into its wavelet coefficients. Again, a shift-invariant wavelet transform is used to obtain the wavelet coefficients which are used as inputs to the network. Shown in Figure 3.15 is a test image and its 2-level undecimated wavelet decomposition.

3.3.2. Training Data Selection. The training data set is chosen from a primary set of candidate data. These candidate data are generated from 200 synthetic images with the same parameters of the image in Figure 3.14, except



(a) Test Image (b) LL_2 (c) HL_2 (d) LH_2

(e) HH_2 (f) HL_1 (g) LH_1 (h) HH_1

Figure 3.15 A test image and its stationary wavelet transform coefficients

that each image contains approximately 25 targets. After obtaining the wavelet decomposition of each image, the resulting decomposition subbands are split into 4×4 subimages. Thus there are $4 \times 4 \times 7 = 112$ inputs into the neural network, representing the vertical, horizontal, and diagonal details of both levels, and the approximation coefficients of the second level. An example of a 4×4 block of the original image which contains a target (a), its wavelet coefficients (b) - (h), and its desired output (i) are shown in Figure 3.16.

There are approximately 5000 (200 images $\times$ 25 targets) sets of wavelet coefficients representing 4×4 blocks that contain a target in the original imagery. All of these 5000 sets are chosen as training data. Also chosen are 15,000 sets of wavelet coefficients in the candidate imagery that represent the absence of a target. This results in a training set with 25% of the inputs (wavelet coefficients) representative of targets in the original image, and 75% without. This 3:1 ratio of inputs representing “non-targets” to inputs representing targets is chosen since a much greater area

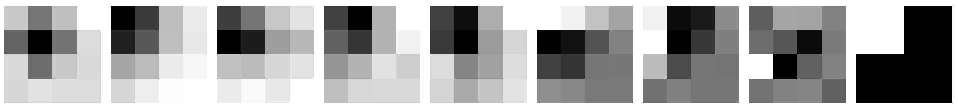

(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 3.16 4×4 block of the original image with target, 4×4 blocks of wavelet transform coefficients, and 2×2 desired output

of the synthetic image used for testing the network is populated with low variance noise, not target signatures. A 1:1 ratio between the target and non-target data was found to contribute to a higher false alarm rate at the network output, likely because the neural network tended to overtrain on the data containing targets.

The input training data and desired output need to be arranged in a vector. The desired output data would be arranged as $[1 \;\; 0 \;\; 0 \;\; 0]^T$ for the example shown in Figure 3.16(i). For the input, let a 4×4 block of coefficients in a certain subband, say the approximation, be represented as $\mathbf{C}_{LL_2}$, where the block of coefficients have been rearranged into a 16×1 column vector. The input data are preprocessed to have zero mean, and are normalized by their standard deviation. Call this resulting data $\mathbf{C'}_{LL_2}$. The preprocessed input data (of length 112) are presented to the network in the following form:

$$\mathbf{x} = [\mathbf{C'}_{LH_1} \; \mathbf{C'}_{LH_2} \; \mathbf{C'}_{HL_1} \; \mathbf{C'}_{HL_2} \; \mathbf{C'}_{HH_1} \; \mathbf{C'}_{HH_2} \; \mathbf{C'}_{LL_2}]^T . \qquad (3.32)$$

3.3.3. Network Architecture and Training. The neural network used here has a feedforward multilayer architecture and is trained by the vector-matrix formulation of the gradient-descent backpropagation algorithm [33]. There are 112 inputs to the network, 60 hidden layer neurons, and 4 output neurons. A network diagram is shown in Figure 3.17.

In this network, a hyperbolic tangent sigmoid transfer function is chosen for the hidden layer neurons. This transfer function captures a significant range of the preprocessed input data in its linear region, and saturates the outliers. Its range is $[-1, 1]$. A logarithmic sigmoid transfer function is chosen for the output

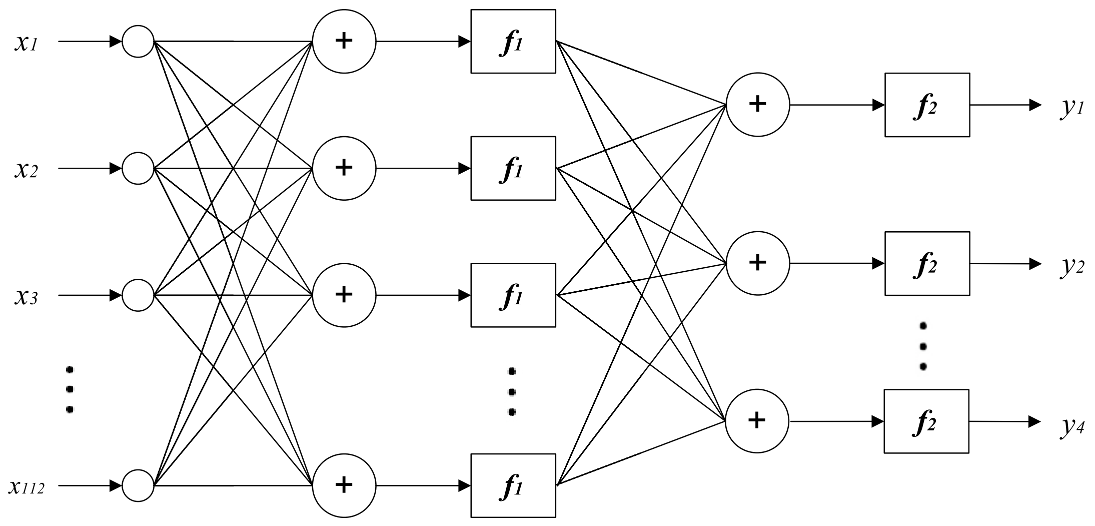


Figure 3.17 Neural network architecture - 112 inputs, 60 hidden layer neurons, 4 output neurons

layer neurons. This function allows the output data to take on a probabilistic representation, since its range is $[0, 1]$. Also, the desired output used to train the network contains only values of either 0 or 1, so this function is more appropriate than a tangent sigmoid. Mathematically these transfer functions are expressed as

$$f_1(v) = \frac{2}{1 + e^{-2v}} - 1; \quad f_2(v) = \frac{1}{1 + e^{-v}}. \tag{3.33}$$

The approximately 20,000 inputs and corresponding desired outputs are first randomized before being presented to the network. One hundred epochs of training with a learning rate of 0.01 gives the mean-squared-error (MSE) performance curve shown in Figure 3.18. The MSE converges to only a third of the initial value at the first epoch. Although this would seem to characterize bad performance, the error remains high because of the sparsity of the desired output data. Most of the desired output data are zero, and only a few have a value of one. The block diagram in Figure 3.19 depicts the training methodology.

3.3.4. Network Testing and Performance Evaluation. Once the network has been trained, its performance needs to be characterized. For network

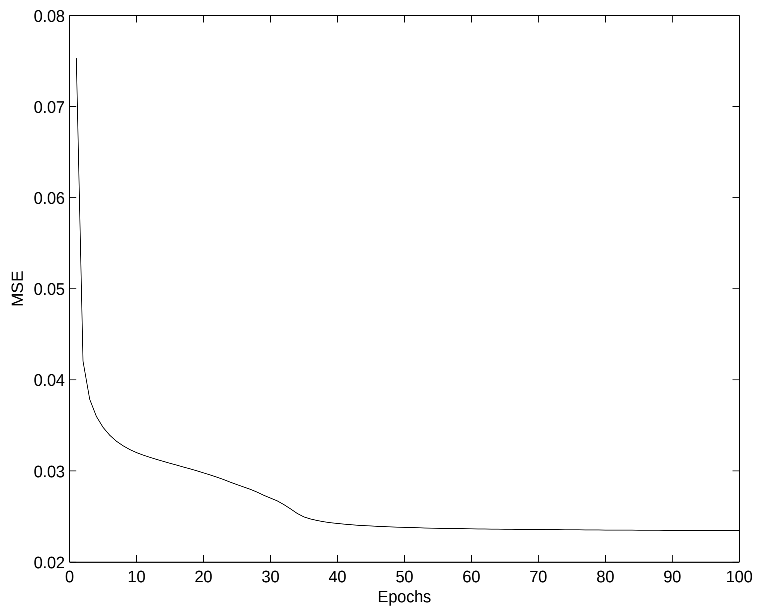


Figure 3.18 Training epochs vs. MSE performance

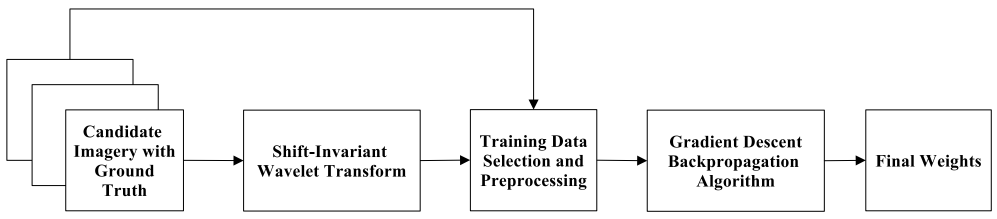
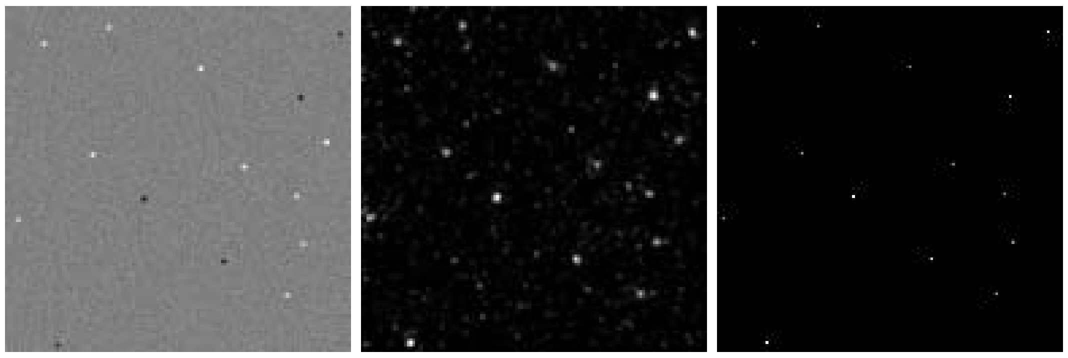

Candidate Imagery with Ground Truth
Shift-Invariant Wavelet Transform
Training Data Selection and Preprocessing
Gradient Descent Backpropagation Algorithm
Final Weights

Figure 3.19 Training paradigm

testing, the synthetic images are generated similarly to the image shown in Figure 3.14(a), each populated with 15 targets. After decomposing an image with the wavelet transform, a 4×4 moving window is utilized to gather data over all of the wavelet subbands to form the input to the network. Using a moving window gives more accurate output data as compared to simply splitting the subbands up into one set of 4×4 blocks. Target signatures can fall at the border of any given block of subband data, so a moving window approach minimizes the chance that a target is missed by the network.

Unfortunately, this method makes the presentation of the inputs to the network, along with the reconstruction of the output data, more computationally expensive than using a single set of 4×4 blocks from the subbands. For any 2×2 block of output data, the result is upsampled and appropriately translated to match the corresponding location in the input imagery. These output responses are then summed, resulting in much better detection performance when compared to not using a moving window approach.

After the response has been computed for a particular image, a probabilistic "likelihood" dataset now exists. The resulting image from the network is then subjected to non-maximal suppression. This results in a detection list with probable target coordinates and likelihood values. A test image, its neural network output, and the results from non-maximal suppression are shown in Figure 3.20. A simple block diagram of the network testing methodology is shown in Figure 3.21.



(a) Test Image (b) Nerual Network Output (c) Non-maximal Supression

Figure 3.20 Detection results for a test image

The detection list resulting from non-maximal suppression is compared against the ground truth generated with the original image, and the receiver operating characteristic (ROC) is computed. The ROC can be used to measure the detection and clustering performance of the network. ROC curves were computed for three datasets, each containing 50 images, but with a different noise variance per dataset. These curves are shown in Figure 3.22.

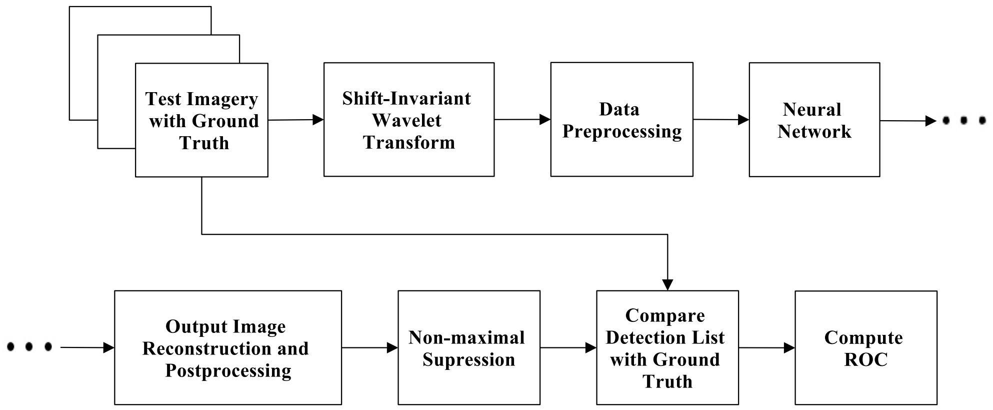

Test Imagery with Ground Truth
Shift-Invariant Wavelet Transform
Data Preprocessing
Neural Network
Output Image Reconstruction and Postprocessing
Non-maximal Supression
Compare Detection List with Ground Truth
Compute ROC

Figure 3.21 Network testing and performance evaluation

As can be seen from the ROC curves, the detection performance of the network for the given test imagery is fairly good, with a relatively low probability of false alarm vs. probability of detection. The performance of the network is degraded as the noise variance increases, as expected.

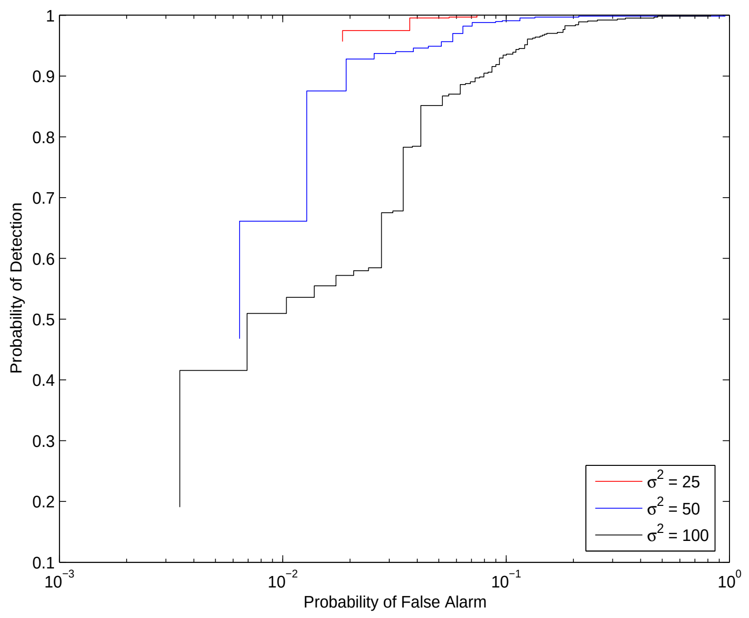


Figure 3.22 ROC curves for datasets with $\sigma^2 = 25$, $\sigma^2 = 50$, and $\sigma^2 = 100$

4. IMAGE REGISTRATION ALGORITHMS

4.1. OVERVIEW OF IMAGE REGISTRATION

Automatic image registration is the process establishing point-by-point correspondence between two images of a scene, with little or no human interaction. This is a necessary step for visualizing a single scene composed of many images as one image mosaic. The basic details of image registration are presented. Also outlined here are a baseline image registration algorithm utilizing spatial correlations between two images, and the results of a scale-space method, called the shift-invariant feature transform (SIFT). These algorithms and their results are discussed in detail.

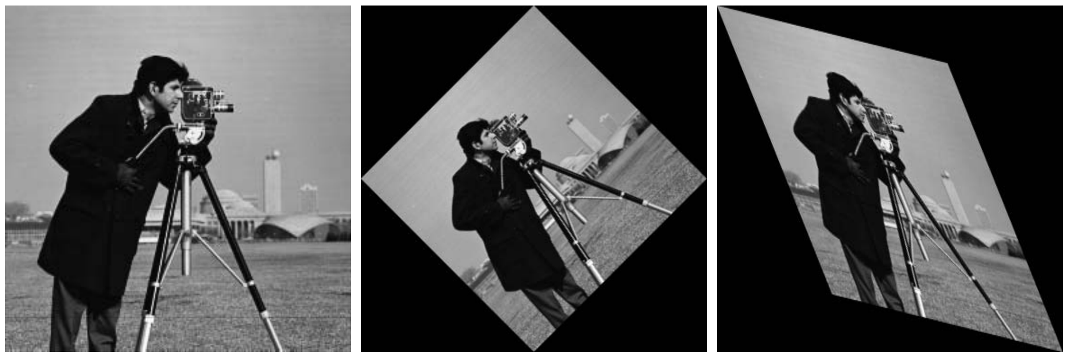
4.1.1. Terminology. When registering two images together, one image is called the *reference image*, and the other the *sensed image*. The point-to-point correspondences found between the two scenes are called *control points*. The reference image remains stationary, while the sensed image is subjected to some transformation or warping which puts it into the same coordinate system as the reference image. When registering multiple images, after initial registration of the first two images, the previous sensed image becomes the reference image and a new sensed image is registered with this reference image, and so on. There are essentially five steps in performing image registration regardless of which technique may be employed at a certain step of a registration algorithm. The steps of basic automated image registration are listed below.

1. **Preprocessing:** Image smoothing, deblurring, removal of local mean, edge detection, etc.
2. **Feature selection:** Extraction of points, lines, regions, templates, etc. from an image.
3. **Feature correspondance:** Determination of the correspondence between features in two images and selection of control points.
4. **Transformation function estimation:** A linear-conformal, affine, projective or possibly higher order (polynomial) transformation determined from the coordinates of the control points.

5. **Resampling:** Interpolation and resampling of a sensed image to the coordinate system of a reference image given the transformation function.

Studied in this thesis are the first three steps of the registration process. The last steps – transformation function estimation and resampling – are not trivial, but remain the same for most registration algorithms.

4.1.2. Types of Transformations. The most commonly used transformation functions used in the baseline and subsequent algorithms are the linear-conformal and affine transformations. A linear-conformal mapping is a subset of the affine mappings. It is composed of rotation, scaling, and translation. The general affine transformation is composed of theses three operations plus a shear operation. Only two control point pairs are needed to define a linear-conformal transformation, and this is the mapping that is used when the automatic registration algorithm only finds two control point pairs. Three control point pairs are needed to define a general affine transformation, and this type of transformation is used when three or more control points are found. The following discussion of transformation types follows the notation in [34]. Shown in Figure 4.1 are an image (a) and its mapping under a linear-conformal (b) and affine (c) transformation.



(a) (b) (c)

Figure 4.1 An example image and the image under a linear-conformal and affine tranformation

4.1.2.1 Linear-conformal mappings. The linear-conformal mapping is performed by composing a rotation $\mathbf{R}$, scaling $\mathbf{S}$, and translation (or displacement) $\mathbf{D}$, and can be denoted mathematically as

$$\mathbf{P}'_j = \mathbf{D}_{x_0,y_0}\mathbf{S}_s\mathbf{R}_\theta\mathbf{P}_j, \tag{4.1}$$

where $\mathbf{P}'_j$ and $\mathbf{P}_j$ are the jth pair of coordinates in the sensed and reference images respectively. This is expressed explicitly as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.2}$$

or equivalently

$$\begin{aligned} x' &= xs\cos\theta - ys\sin\theta + x_0 \\ y' &= xs\sin\theta + ys\cos\theta + y_0, \end{aligned}$$

where four parameters define the transformation: x_0 and y_0 are the displacements in the x and y directions, s the scaling parameter, and θ the angle of rotation. The angle θ is independent of the other parameters and is calculated from the two control point pairs with

$$\theta = \theta_2 - \theta_1 = \arctan\left(\frac{y'_2 - y'_1}{x'_2 - x'_2}\right) - \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right). \tag{4.3}$$

Once θ is known, the other three parameters can be solved with the system of three equations in Eq. 4.2.

4.1.2.2 General affine mappings. The general affine transformation includes the component of shear. This transformation has six parameters which can be computed from at least three non-colinear control point pairs. The transformation matrix is given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.4}$$

where (x, y) and (x', y') are the coordinates before and after the transformation. If only a few control point pairs are found with the registration algorithm, any error in the coordinates usually causes inaccurate registration. When many control point pairs are provided or found by an automatic registration algorithm, they can be used to determine a least-squares estimate of the six parameters in Eq. 4.4. An error criteria function can be defined as

$$\begin{aligned}\varepsilon(a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}) = \sum_{j=1}^{n} [(&a_{11}x_j + a_{12}y_j + a_{13} - x'_j)^2 \\ &+ (a_{21}x_j + a_{22}y_j + a_{23} - y'_j)^2].\end{aligned}$$

To minimize the error each in parameter, the six partial derivates $\partial\varepsilon/\partial a_{ij}$ are evaluated and the expressions are set equal to zero which yields a system of six equations represented in the following matrix

$$\begin{bmatrix} \sum x_j^2 & \sum x_j y_j & \sum x_j & 0 & 0 & 0 \\ \sum x_j y_j & \sum y_j^2 & \sum y_j & 0 & 0 & 0 \\ \sum x_j & \sum y_j & n & 0 & 0 & 0 \\ 0 & 0 & 0 & \sum x_j^2 & \sum x_j y_j & \sum x_j \\ 0 & 0 & 0 & \sum x_j y_j & \sum y_j^2 & \sum y_j \\ 0 & 0 & 0 & \sum x_j & \sum y_j & n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} \sum x'_j x_j \\ \sum x'_j y_j \\ \sum x'_j \\ \sum y'_j x_j \\ \sum y'_j y_j \\ \sum y'_j \end{bmatrix}, \tag{4.5}$$

where the summations are over j from 1 to n. n is the number of control point pairs provided to estimate the transformation parameters. Solving the above matrix equation for $a_{11}, \ldots, a_{23}$ gives the least-squares estimate of the parameters.

4.1.3. Registration Example. Shown in Figure 4.2 is an example of registration performed on typical LAMD imagery. Four frames have been selected and registered together using general affine transformations with the correlation-based technique for control point selection. The frames shown in the figure have a large amount of overlap and have been stitched together with little error. Correlations were performed using subimages of size 32×32 taken from the reference images.

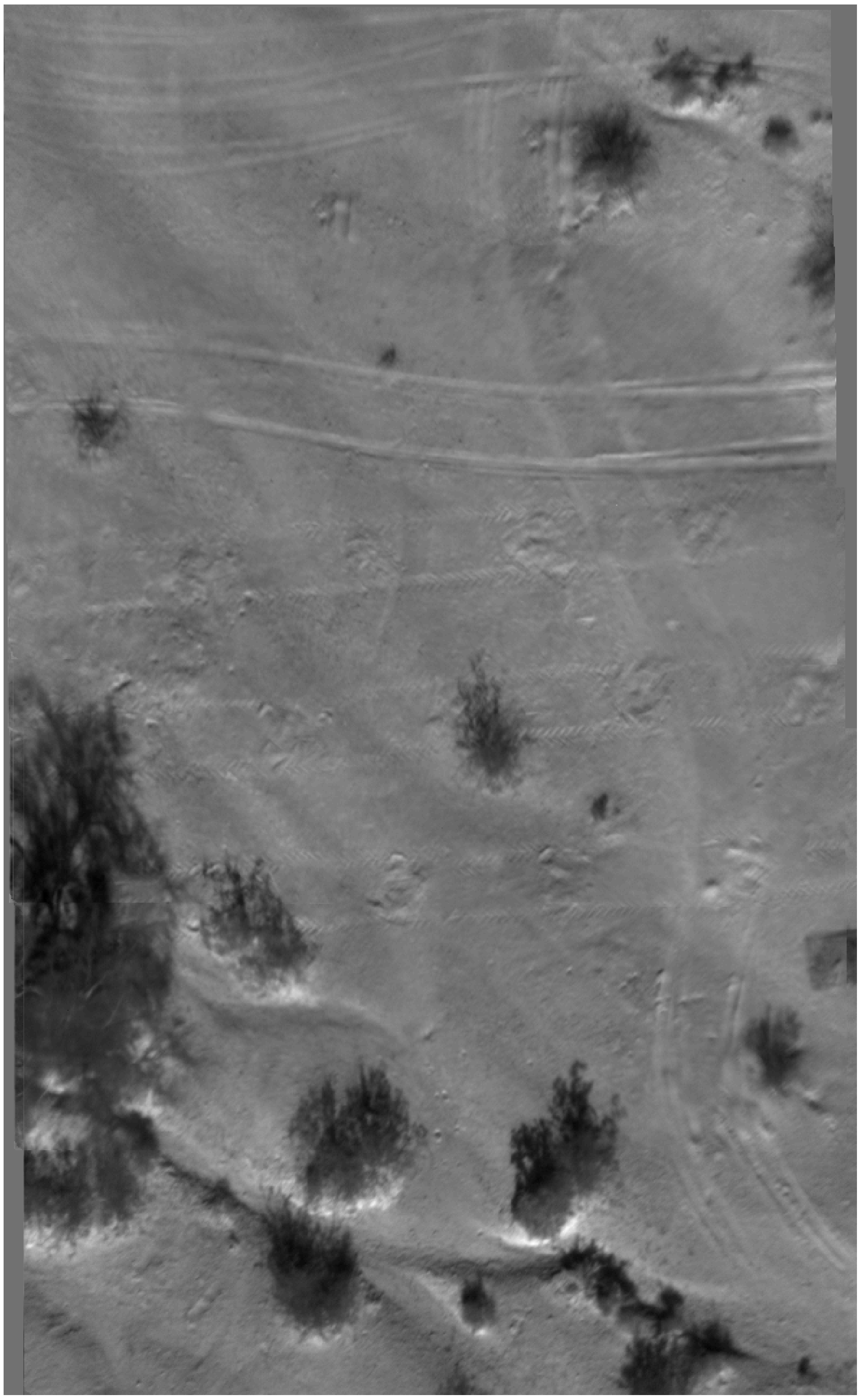


Figure 4.2 Four LAMD frames registered with the correlation-based method and affine transformations

4.2. CORRELATION-BASED METHOD

The current baseline registration algorithm used to register a set of image frames is based on a correlation technique. Given two images, a subimage of the reference image is selected by some desired criterion, and then a correlation of the subimage is performed with the sensed image. If the maximum value of correlation at some location in the sensed image exceeds a threshold, the corresponding points are recorded and used to estimate the transformation function.

4.2.1. Reference Image Feature Selection.

In this algorithm, the feature extracted from the reference image is a small region of the image, possibly with a size of 32×32 or 64×64 pixels. Smaller regions, such as 16×16 pixels, tend to contribute to higher numbers of inaccurate correlations, which can cause registration errors. Larger regions can cause problems when there is a small amount of overlap between the referenced and sensed images. A larger region may also result in poor correlation if rotation or warping is significant. If the region is too large, it is possible that no maximum correlation value will reach the minimum threshold and the number of control points the algorithm is able to find might be small.

These subimages are selected based a maximum energy criterion. Over any given $M \times N$ region of the image, the energy is calculated. Here M and N are the dimensions of the subimages used for calculating the correlations. A flat mask $h(x, y)$ of size M by N with values $1/MN$ is convolved with the squared pixel values of the reference image, resulting in a local mean-squared image. Let the reference image be denoted by $I(x, y)$. Then the local mean-squared image is

$$I_{ms}(x,y) = \mathcal{F}^{-1}\{\mathcal{F}\{I_r^2(x,y)\}\hat{h}(u,v)\}, \tag{4.6}$$

where the spatial convolution has been performed in the frequency domain and $\hat{h}(u, v)$ is the Fourier transform of the mask $h(x, y)$. A local mean image $I_m(x, y)$ is then computed with

$$I_m(x,y) = \mathcal{F}^{-1}\{\mathcal{F}\{I_r(x,y)\}\hat{h}(u,v)\}. \tag{4.7}$$

The local energy at each location in the image can then be computed in the root-mean-square (RMS) sense, i.e.

$$I_{rms}(x,y) = \sqrt{\left|[I_{ms}(x,y) - I_m(x,y)]^2\right|}. \tag{4.8}$$

The RMS image is then subjected to non-maximal suppression, and a list of high-energy locations is created. The list can be sorted from the highest energy locations descending or it can be randomized. The current baseline algorithm randomizes the locations to be chosen as features, since simply choosing the highest-energy locations might result in clustering of features around a single high-energy locality in the image, such as a single bush or tree in a highly correlated background. Randomizing the list helps to spread the chosen features more uniformly across the image, which results in more accurate estimation of the transformation function once the control points have been computed.

4.2.2. Correlation Calculations. Once the list of feature locations from the reference image has been created, the subimages surrounding those locations are chosen and then a correlation calculation is made with the sensed image. The two-dimensional correlation coefficient r for two matrices $\mathbf{A}$ and $\mathbf{B}$ both of size $M \times N$ is given by

$$r = \frac{\displaystyle\sum_m \sum_n (A_{mn} - \bar{\mathbf{A}})(B_{mn} - \bar{\mathbf{B}})}{\left[\left(\displaystyle\sum_m \sum_n (A_{mn} - \bar{\mathbf{A}})^2\right)\left(\displaystyle\sum_m \sum_n (B_{mn} - \bar{\mathbf{B}})^2\right)\right]^{1/2}}, \tag{4.9}$$

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are the means of the matrices $\mathbf{A}$ and $\mathbf{B}$. It is quite expensive to compute this correlation coefficient at every pixel location as the reference subimage moves across the sensed image, so the computation is performed in the frequency domain.

First, the RMS values $I'_{rms}(x,y)$ of the sensed image $I'(x,y)$ are obtained in the same manner as the reference image, using the same flat mask $h(x,y)$. The Fourier transform of the sensed image is also obtained, denoted by $\hat{I}'(u,v)$. Then the mean of the first subimage chosen from the reference image feature list is subtracted and then this result is normalized to have unit energy. Let this preprocessed "chip" be denoted $g(x, y)$. This terminology is borrowed from the impulse response of the matched filter in a digital communications system. The following correlation calculation is performed in the frequency domain

$$\hat{r}_1(u, v) = \hat{I}'(u, v)\hat{g}^*(u, v), \tag{4.10}$$

where * denotes the complex conjugate. The final correlation coefficients of the chip and sensed image are given by

$$r(x, y) = \frac{\mathcal{F}^{-1}\{\hat{r}_1(u, v)\}}{\sqrt{MN}I'_{rms}(x, y)} = \frac{r_1(x, y)}{\sqrt{MN}I'_{rms}(x, y)}. \tag{4.11}$$

If the maximum of the correlation values $r(x, y)$ exceeds a threshold, then the location of the peak is found and the coordinates of the subimage taken from the reference image and the location of the correlation peak in the sensed image are put into the list of control point pairs. This process is then repeated for the next subimage in the reference image feature list.

4.3. SCALE-INVARIANT FEATURE TRANSFORM

The scale-invariant feature transform (SIFT) is a novel algorithm developed by David Lowe [20] which is used to extract invariant features that can be used to reliably match different views of an object or scene. The features extracted from a given image are relatively invariant to significant amounts of scaling and affine warp. Presented in this thesis is how SIFT features can be used to aid in image registration. The algorithm is able to successfully find control points without resorting to a correlation based method by matching features in a high-dimensional feature-space. The current implementation of the SIFT algorithm uses a brute-force search to find the minimum distance between reference image features and sensed image features, which is computationally expensive. Other matching algorithms such as the Best-Bin-First (BBF) algorithm [35] can be used in future implementations to search through the feature space relatively quickly as compared to brute-force algorithms.

4.3.1. Overview of SIFT. The SIFT algorithm can efficiently extract many features over many scales and location of a single image. A typical image of size

500×500 gives rise to approximately 2000 stable features. There are five important steps to extracting these invariant features from an image. These steps are listed below:

1. **Scale-space construction:** A scale space is constructed for the given image by repeatedly convolving the image with a Guassian kernel. Each plane of the scale space is then subtracted from its adjacent plane to obtain a difference-of-Gaussian image. The set of difference-of-Gaussian images is used to perform extrema detection.
2. **Scale-space extrema detection:** A search over all possible scales and locations in the difference-of-Gaussian images is performed to find candidate points which are invariant to scale and orientation.
3. **Keypoint localization:** Interpolation is performed to accurately fit a candidate feature location (keypoint) in location and scale. Keypoints are then refined based on measures of their stability.
4. **Orientation assignment:** One or more orientations are assigned to each keypoint based on the local image gradient directions at the keypoint's scale in scale space. Any further operations are performed relative to the this orientation to provide invariance to any affine transformation.
5. **Keypoint descriptor construction:** At the selected scale and location of each keypoint, the surrounding image gradients are used to construct a keypoint descriptor. The descriptor is essentially the location of each keypoint in the feature space.

The keypoint descriptors are very distinctive, allowing a single feature in the reference image to be matched to its corresponding feature in the sensed image with high probability. The figures presented in the following discussion have been adapted from the original paper by Lowe [20].

4.3.2. Scale-space Construction. The first step of the SIFT algorithm is the construction of scale space for a given image. The scale space is defined as a function $L(x, y, \sigma)$ that is produced from a convolution of a variable-scale Gaussian function $G(x, y, \sigma)$ with an image $I(x, y)$. Specifically,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{4.12}$$

where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left[\frac{-(x^2 + y^2)}{2\sigma^2}\right]. \tag{4.13}$$

Keypoints are located not in this scale space, but rather in the difference-of-Gaussian images that result from the difference of adjacent planes in scale space separated by a multiplicative constant k:

$$\begin{aligned} D(x, y, \sigma) &= [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

The reasoning behind looking for keypoints in these difference-of-Gaussian images is twofold. First, D is relatively easy to compute with a point-by-point subtraction of the two planes of scale space. Second, the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian-of-Gaussian $\sigma^2\nabla^2 G$. It has been found that the extrema of $\sigma^2\nabla^2 G$ produce the most stable image features as compared to other possible operations, such as the gradient and Harris corner function [36]. In [20] Lowe describes the relationship between D and $\sigma^2\nabla^2 G$ via the heat diffusion equation

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G, \tag{4.14}$$

here parameterized in terms of σ rather than $t = \sigma^2$. An approximation of $\sigma\nabla^2 G$ is provided from the finite difference approximation of $\partial G/\partial\sigma$ using the difference of adjacent scales $k\sigma$ and σ:

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}, \tag{4.15}$$

or equivalently

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G. \tag{4.16}$$

Thus the difference-of-Gaussian D is an approximation of the Laplacian-of-Gaussian $\sigma^2\nabla^2 G$ multiplied by the constant factor $k-1$. The approximation error approaches zero as $k \to 1$, but the error has little impact on extrema detection even when there are large differences in scale.

The construction of the scale space $L(x, y, \sigma)$ and difference-of-Gaussian images $D(x, y, \sigma)$ is depicted in Figure 4.3. The original image $I(x, y)$ is first expanded via bilinear interpolation, doubling its dimensions and quadrupling its area. This image expansion yields more keypoints from the SIFT algorithm. This base image is then incrementally convolved with Gaussian kernels $G(x, y, k^i\sigma)$ at a scale k^i, where $0 \leq i \leq s + 2$, for integer i. s is the number of intervals that an octave, which represents a doubling of σ, is divided into. k is chosen such that $k = 2^{1/s}$. For each octave, $s + 3$ blurred images are created so that the detection of extrema covers the complete octave. Once a complete octave has been created, the image in scale space which has twice the initial value of σ is downsampled by a factor of two and this result is used to construct the next octave of the scale space. The current implementation of the SIFT algorithm uses $s = 3$ with four octaves, so $k = \sqrt[3]{2}$. Shown in Figure 4.3 are two octaves with $s = 2$.

4.3.3. Reduction of Computational Complexity. To compute the scale space, the convolution operation is not performed as explicitly stated above. One feature of the two-dimensional Gaussian kernel is that it is separable. This is exploited in the current implementation. Separability of the kernel arises since it can be written as

$$\frac{1}{2\pi\sigma^2}\exp\left[\frac{-(x^2+y^2)}{2\sigma^2}\right] = \left[\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\right]\left[\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}}\right]. \tag{4.17}$$

This allows a one-dimensional convolution to be performed along rows. This intermediate result is then convolved with the one-dimensional kernel along the columns. This method dramatically reduces the computational complexity required to perform the convolution from $O(n^2)$ to $O(2n)$.

The convolution operation obeys the associative property and the convolution of two Gaussian functions is also a Gaussian function. Using these two facts allows computation to be further reduced when computing the scale space. Progressively

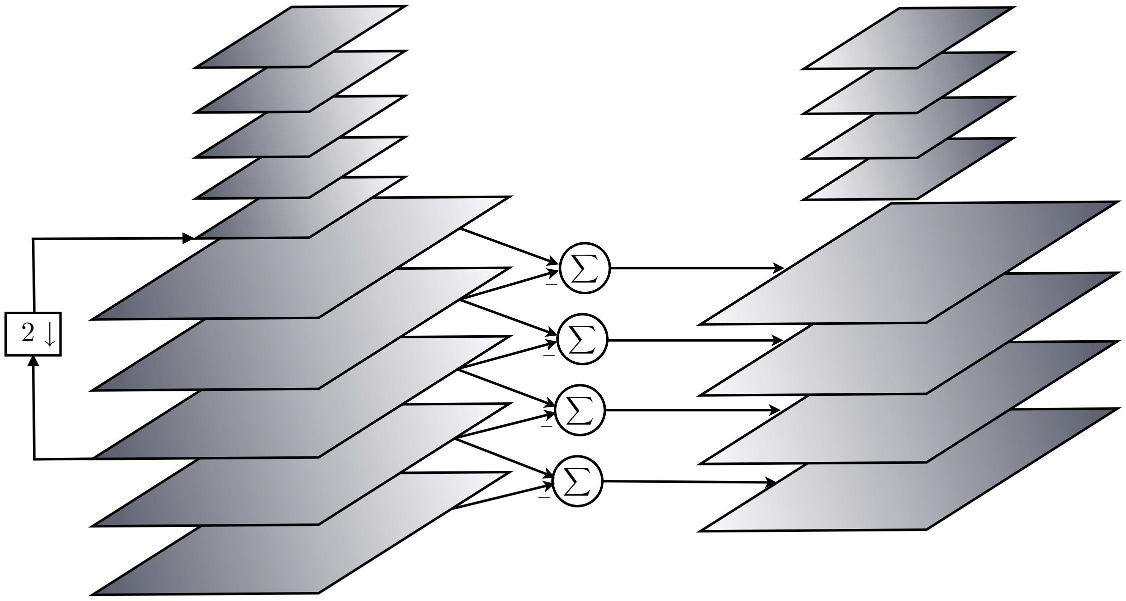


Figure 4.3 Construction of scale space and difference-of-Gaussian images

larger Gaussian kernels need not be convolved with the base image to obtain the scale space images at higher scales, but instead kernels of the same size can be convolved with the image at a lower scale adjacent to the scale being computed. These operations help make the SIFT algorithm very fast since no complex convolutions or FFTs need to be performed.

4.3.4. Scale-space Extrema Detection. Once the difference-of-Gaussian images have been computed, the next step of SIFT is to locate the extrema within these images. This is performed by searching for either a maximum or minimum at each location and scale within these images. Each sample point is compared to its eight neighbors within its scale, and its nine neighbors at both the adjacent higher and lower scales. This is depicted in Figure 4.4, where the sample point is denoted with an "X". If the point is found to be an extremum, its location and scale are recorded. Since most points will be discarded as extrema after the first few checks of its adjacent neighbors, this search requires very little computation and is performed very rapidly.

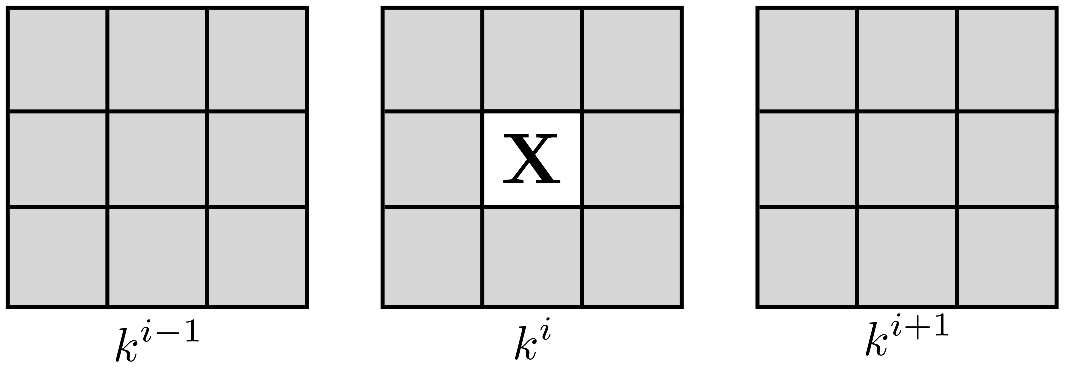


Figure 4.4 A sample point compared to its nearest neighbors in both location and scale

4.3.5. Keypoint Localization and Refinement. After finding the extrema within the difference-of-Gaussian images, the values surrounding an extremum are used to accurately interpolate its location in both the x and y coordinates and in scale σ. This is performed by fitting a 3D quadratic function to the points surrounding the extremum. The approach suggested by Lowe uses a Taylor series expansion up to the quadratic terms of the function $D(x, y, \sigma)$ where the origin is at the sample point:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}, \tag{4.18}$$

where $\mathbf{x} = (x, y, \sigma)^T$ is the offset from the sample point, $\partial D/\partial \mathbf{x}$ is a row vector of derivatives with respect to x, y, and σ, and $\partial^2 D/\partial \mathbf{x}^2$ is the 3×3 Hessian matrix. Evaluating $\partial D(\mathbf{x})/\partial \mathbf{x}$, setting the expression to zero and solving gives the location of the extremum as

$$\hat{\mathbf{x}} = -\left(\frac{\partial^2 D}{\partial \mathbf{x}^2}\right)^{-1} \frac{\partial D}{\partial \mathbf{x}}. \tag{4.19}$$

Both the derivative of D and its Hessian are approximated using differences of neighboring sample points. If the value of the offset is larger than 0.5 in any direction, the extremum lies closer to a different sample point. This point is then found and the process is repeated about this new point. The interpolated value of the extremum $D(\hat{\mathbf{x}})$ is used to reject any unstable extrema with a low contrast. Upon substituting

Eq. 4.19 into Eq. 4.18, the interpolated value becomes

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}. \tag{4.20}$$

The current SIFT implementation discards any extrema where $|D(\hat{\mathbf{x}})| < 0.04$, where the input image was first normalized to range from $[0, 1]$.

Once the low contrast extrema have been discarded, any keypoints arising from edges in the difference-of-Gaussian function need to be eliminated as well. The difference-of-Gaussian function has a strong response along any edges in the image and these are unstable to small amounts of noise. To determine any poorly defined peaks in the difference-of-Gaussian images, the principal curvatures are examined. A poorly defined peak will have a large principal curvature along the edge, but a small curvature in the perpendicular direction. The principal curvatures are computed with the Hessian matrix $\mathbf{H}$ at the location and scale of the keypoint, where

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{4.21}$$

The eigenvalues of the Hessian $\mathbf{H}$ are proportional to the principal curvatures of D. Instead of explicitly computing the eigenvalues of $\mathbf{H}$, the ratio of principal curvatures can be computed by evaluating only the ratio of the eigenvalues. This approach is used by Lowe and is attributed to Harris and Stephens [37]. Suppose α and β are the larger and smaller eigenvalues respectively. Then the sum and product of the eigenvalues can be computed with the trace and the determinant of $\mathbf{H}$:

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta, \tag{4.22}$$

$$\det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \tag{4.23}$$

If the determinant is negative, then the point is discarded as being an extremum. Let r be the ratio of the principal curvatures (ratio of the two eigenvalues) so that $\alpha = r\beta$, then

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{4.24}$$

So for a given ratio r, if the inequality

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{4.25}$$

is not satisfied, the point is discarded as being a poorly defined peak. The current implementation eliminates points where $r > 8$.

4.3.6. Orientation Assignment. To achieve invariance to any possible rotation of a feature, an orientation is assigned to all candidate points which have survived the process of localization and refinement. Rather than working with difference-of-Gaussian images, further calculations are all performed with the Gaussian-smoothed images $L(x, y, \sigma)$. Prior to computing the orientation for a single keypoint, the gradient magnitudes $m(x, y, \sigma)$ and orientations $\theta(x, y, \sigma)$ of all points in L are precomputed by using pixel differences with

$$m(x, y, \sigma) = \sqrt{[L(x+1, y, \sigma) - L(x-1, y, \sigma)]^2 + [L(x, y+1, \sigma) - L(x, y-1, \sigma)]^2} \tag{4.26}$$

and

$$\theta(x, y, \sigma) = \arctan\left[\frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)}\right]. \tag{4.27}$$

Once the gradient magnitudes and orientations have been computed, an orientation histogram is formed by using an area of pixel values (of magnitude and orientation) surrounding the location at the nearest scale to that of the keypoint. Currently, a 15×15 region around the keypoint location is used to construct the orientation histogram. The magnitudes in this region are weighted by a circular gaussian kernel with a standard deviation 1.5 times that of the scale σ. This keeps strong gradient magnitudes which might lie several pixels away from the keypoint location from contributing to an incorrect orientation. Each magnitude is then placed into the histogram bin corresponding to the gradient orientation. The histogram has 36 bins each representing 10° to cover the 360° range of orientations. The maximum peak of the histogram is found, and then a parabola is fit to it and the points on each side of the peak to better interpolate the location of the maximum. From this peak, an orientation is assigned to the keypoint. If another peak in the histogram has a height 80% that of the maximum, the interpolation process is repeated and another keypoint is created with this orientation. As stated by Lowe, and from our own observations, approximately 15% of keypoints are assigned two orientations. These additional orientations aid in increasing the stability of matching.

4.3.7. Keypoint Descriptor Construction. The keypoint descriptor is essentially a point in a 128-dimensional feature space which can be matched with high probability against another descriptor representing the same feature in a different image. The descriptor is constructed by using a method similar to that used to find the keypoint orientation. A 16×16 region of the magnitudes and orientations around the keypoint and nearest its scale is extracted and the magnitudes subjected to a Gaussian window with standard deviation half that of the window size ($\sigma = 8$). Again, the Gaussian is used to attenuate any large gradient magnitudes which may lie several pixels away from the keypoint location. All orientations in this region are then rotated relative to the orientation of the keypoint to provide for rotation invariance of the descriptor.

Once the preprocessing of the magnitudes and orientations has been performed, the 16×16 region is split into 16 separate 4×4 regions. For each subregion, the magnitude of the gradients are then placed into histogram with 8 bins each covering a 45° range of orientation. This operation is depicted in Figure 4.5.

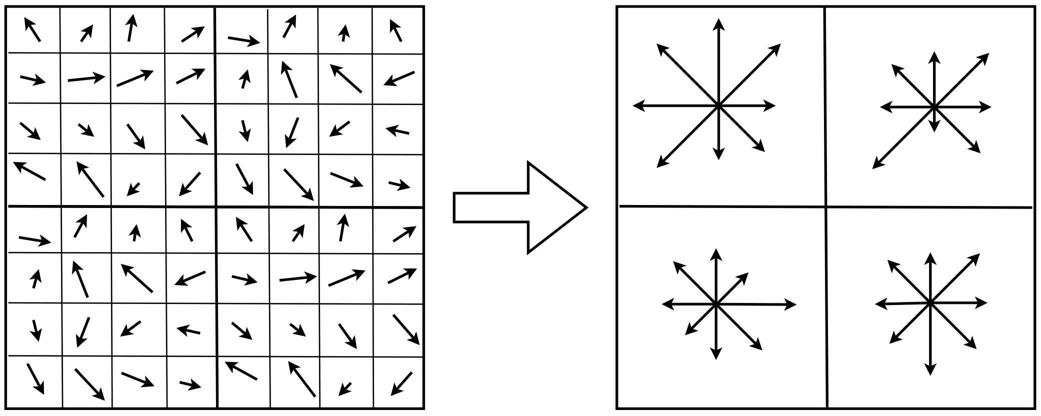


Figure 4.5 Construction of a descriptor from gradient magnitudes and orientations surrounding a keypoint

Rather than the 2×2 descriptor array shown above, the current SIFT implementation uses a 4×4 descriptor array where each element of the array contains an eight-bin histogram. This results in a 128-dimensional feature vector. This vector is normalized to unit length before being stored in a database of keypoints and their descriptors for a given image. Normalization of the vector helps to reduce the effects of illumination change from image to image. Once the feature vectors have been computed for two adjacent images, the current implementation of SIFT employs a brute-force search to find the minimum distance between the features in the reference and sensed images. This is currently the most expensive portion of the algorithm.

4.3.8. Results. The results of the SIFT algorithm are shown graphically in Figures 4.6 and 4.7. The 30 points with smallest minimum distance matched in the feature space of both images are shown. The numbers in the images correspond to the distance, with 1 being the smallest distance and 30 being the largest. Note that points 17, 19, and 29 are erroneous. The registration of two different segments of LAMD images is shown in Figure 4.8. The first segment (a) is a collection of 22 images. The second (b) is a collection of 10 images. As can be seen in the last figures, the overall registration result is relatively good and compares well with the correlation-based algorithm.

4.3.9. Further Applications of SIFT. The SIFT algorithm has been implemented in this work to aid in the task of automatic control point selection specifically for image registration. SIFT features can be readily adapted to other applications, such as object recognition and change detection. In [20], Lowe discusses the use of the Hough transform to identify clusters which belong to a single object and then performing verification of object determination via the least-squares solution for consistent pose parameters. This allows SIFT features to be used to easily and consistently identify objects among clutter and occlusion with relatively little computational complexity as compared to other algorithms performing this task.

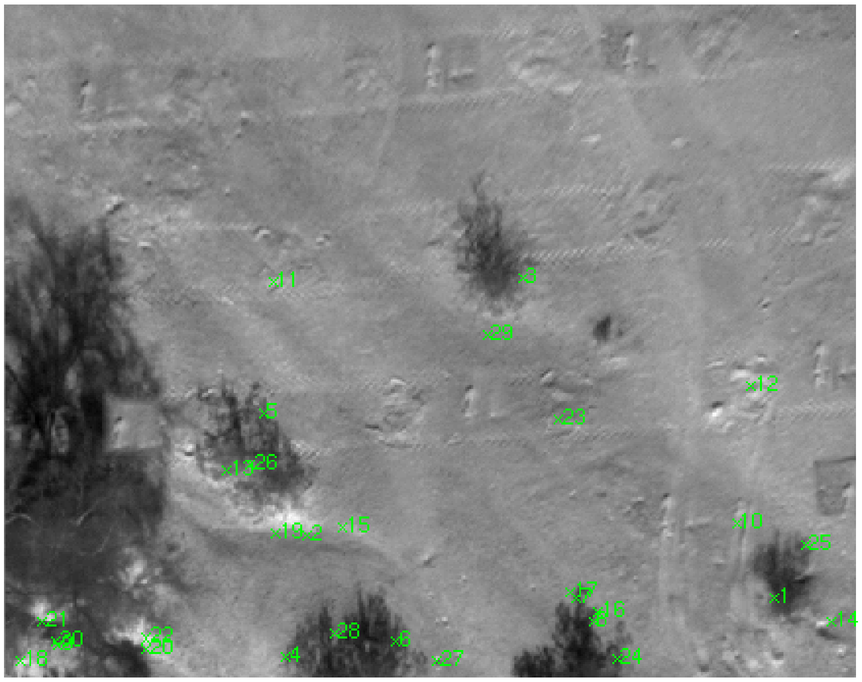


Figure 4.6 SIFT control-points for a LAMD sensed image

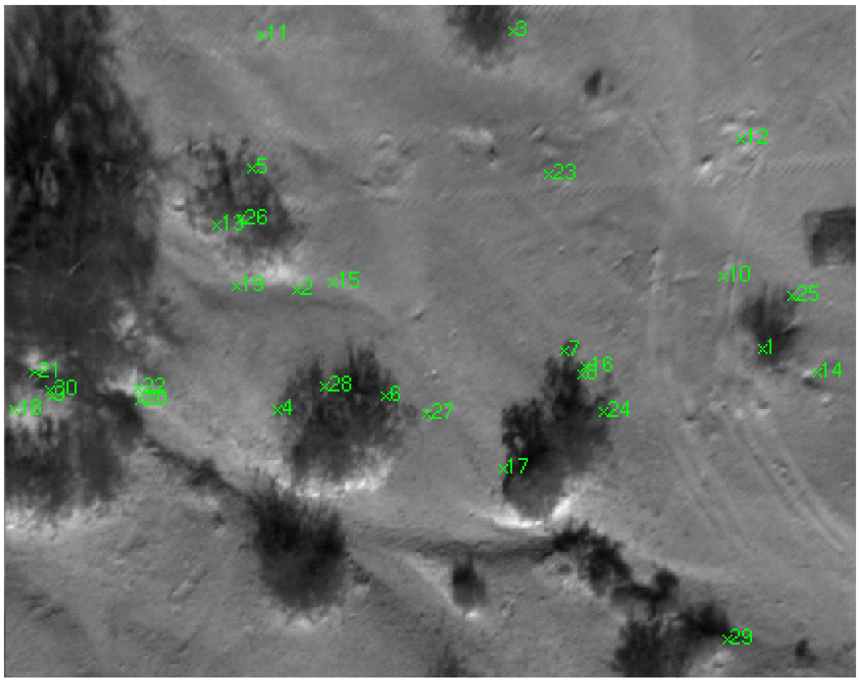
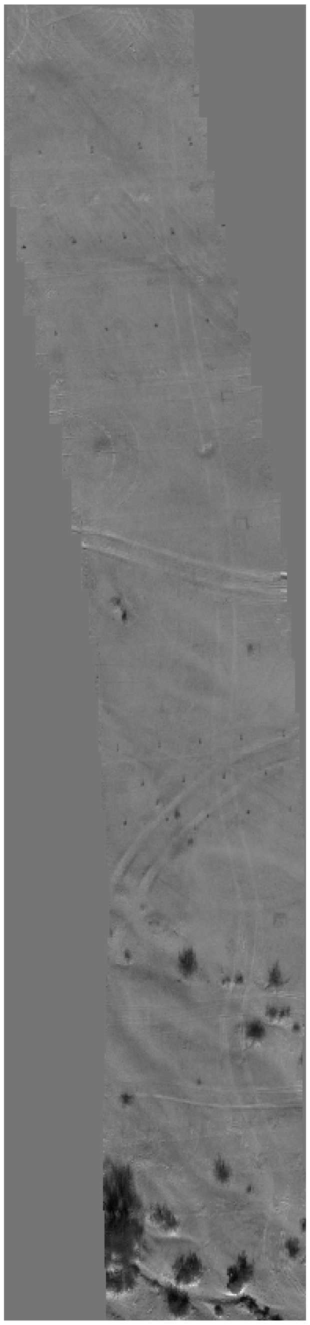


Figure 4.7 SIFT control-points for a LAMD reference image


(a)

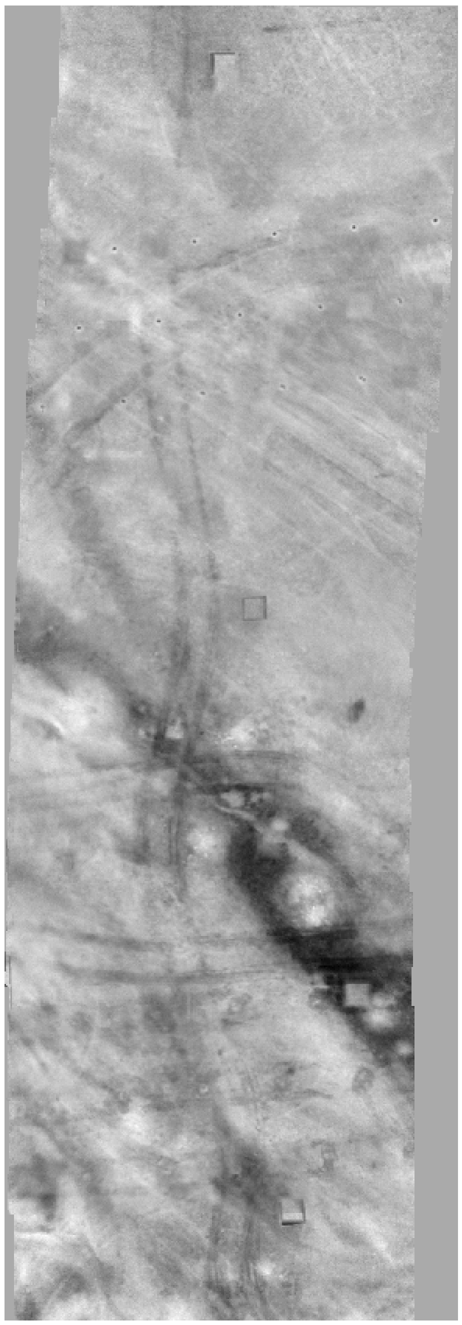

(b)

Figure 4.8 Two LAMD segments after registration with SIFT

5. CONCLUSIONS AND FUTURE WORK

In this thesis several multiscale methods applied to the important processes of anomaly detection and image registration have been presented and discussed. This work has shown that either wavelet domain or scale-space methods can be used for these applications and can perform as well or better than the baseline algorithms, albeit with greater computational complexity.

The wavelet transform and its shift-invariant counterpart have been presented and discussed. The use of the shift-invariant wavelet transform for target detection has been explored using two different architectures: one based on a statistical test, and the other a multilayer perceptron neural network. Results in the form of ROC curves have been use to characterize performance of these algorithms, and this performance has been compared to the baseline detection algorithm. Future investigations which assess the performance of other wavelet-based detection architectures presented in the literature could possibly validate the wavelet transform as an invaluable tool for anomaly detection. Further studies into the computational complexity of such architectures are needed.

The use of scale-space theory has been applied to the problem of image registration through the scale-invariant feature transform. The steps of this novel algorithm have been outlined in detail, and results of the application of the algorithm to image registration have been presented. The use of SIFT features for change detection and object recognition is very likely in the near future.

APPENDIX A.

C++ FUNCTION PROTOTYPES FOR CORRELATION-BASED REGISTRATION AND RX DETECTION

Included in this appendix are C++ interface files containing function prototypes for correlation-based registration and RX single and multi-band anomaly detection. This code was developed for implementing baseline anomaly detection and image registration algorithms on groundstation hardware provided by the Countermine Division of the Night Vision and Electronic Services Directorate of the U.S. Army. There are three interface files: `imageUtils.h`, providing basic image processing utilities, `detectionTools.h`, providing RX processing capability, and `registrationTools.h`, providing correlation-based control point selection. Fast Fourier transforms were performed using the open source FFTW library.

1. imageUtils.h

```
#ifndef IMAGEUTILS_H
#define IMAGEUTILS_H

/*
 *  imageUtils.h
 *
 *  Created by jbarnes on 9/25/05.
 *  Jeff Barnes
 *  jbarnes@umr.edu
 *  Copyright 2005
 *  Airborne Reconnaissance and Image Analysis Laboratory
 *  University of Missouri - Rolla
 *  All rights reserved.
 *
 */

#include <iostream>
#include <cmath>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;
#include "Image.h"
#include "fftw3.h"


void fft2(int numRows,
          int numCols,
          double *inData_space,
          fftw_complex *outData_freq);
```

```
/*  Description:  Forward 2D Fast Fourier Transform
 *
 *  Inputs:  numRows - number of rows
 *           numCols - number of columns
 *           *inData_space - pointer to spatial domain data
 *
 *  Output:  *outData_freq - pointer to complex FFT data
 */

void ifft2(int fftRows,
           int fftCols,
           fftw_complex *inData_freq,
           double *outData_space);
/*  Description:  Inverse (backward) 2D Fast Fourier Transform
 *
 *  Inputs:  fftRows - number of rows of FFT data
 *                     (same as spatial domain)
 *           fftCols - number of columns of FFT data
 *                     (floor(columns/2) + 1 of spatial
 *                     domain columns)
 *           *inData_freq - pointer to FFT data
 *
 *   Output: *outData_space - pointer to spatial data
 */
void createBoxMask(int height,
                   int width,
                   double *boxMask);
/*  Description:  Create a flat rectangular mask where
 *                all pixel values sum to unity
 *
 *  Inputs:  height - number of rows in mask
 *           width - number of columns in mask
 *
 *  Output: *boxMask - pointer to mask data
 */

void createBoxMaskFFT(int height,
                      int width,
                      int numRows,
                      int numCols,
                      fftw_complex *boxMaskFFT,
                      int rowCen=-1,
                      int ColCen=-1);
/*  Description:  Create FFT spectrum of rectangular mask
 *                where the mask has been zero padded
 *
 *  Inputs:  height - number of rows in spatial mask
 *           width - number of columns in spatial mask
 *           numRows - total number of rows in image
 *           numCols - total number of columns in image
 *
 *  Output:  *boxMaskFFT - pointer to spectrum of mask
 */

void createCircMask(int radius,
                    double *circMask);
/*  Description:  Create a circular mask of a certain radius where
 *                all values sum to unity
 *
 *  Input:  radius - radius of the mask
 *
 *  Output:  *circMask - pointer to mask data
 */

void myfft2(int height,
            int width,
            int numRows,
            int numCols,
            double *image,
            fftw_complex *imageFFT,
            int rowCen=-1,
            int colCen=-1);
/*  Description:  Forward 2D FFT with shifted spectrum
 *                (the first and third quadrants and
 *                second and fourth quadrants are swapped)
 *
 *  Inputs:  height - rows of original data
 *           width - columns of original data
 *           numRows - rows of data to be transformed
 *           numCols - columns of data to be transformed
 *           *image - pointer to image data
 *
 *  Output:  *imageFFT - shifted spectrum of input image
 */

 void complexMult(int arraySize,
                 fftw_complex *inData1,
                 fftw_complex *inData2,
                 fftw_complex *outData);
/*  Description:  Multiplies complex data of type "fftw_complex"
 *
 *  Inputs:  arraySize - size of the input arrays to be multiplied
 *           *inData1 - pointer to first complex array
 *           *inData2 - pointer to second complex array
 *
 *  Output:  *outData - pointer to result of multiplied arrays
 */

void getRMSImage(int numRows,
                 int numCols,
                 double *image,
                 fftw_complex *maskFFT,
                 double *RMSImage);
/*  Description:  Calculates RMS data for a given image
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns
 *           *image - pointer to image data
 *           *maskFFT - pointer to spectrum of square
 *                      (or circular) flat mask
 *
 *  Output:  *RMSImage - pointer to RMS image data
 */

double complexMag(double realNum,
                  double imagNum);
/*  Description:  Returns the magnitude of complex number
 *
 *  Inputs:  realNum - real part of complex number
 *           imagNum - imaginary part of complex number
 *
 *  Output:  returns magnitude of type double
 */

template<class T>
void findMax(T *arrayPtr,
             int arraySize,
             T& maxVal);
/*  Description:  Finds the maximum value of an array
 *
 *  Inputs:  *arrayPtr - pointer to input array
 *           arraySize - size of the array
 *
 *  Output:  maxVal - the maximum value of the array
 */

template<class T>
void findMin(T *arrayPtr,
             int arraySize,
             T& minVal);
/*  Description:  Finds the minimum value of an array
```

```
 *
 *  Inputs:  *arrayPtr - pointer to input array
 *           arraySize - size of the array
 *
 *  Output:  minVal - the minimum value of the array
 */

 void fftWrite(int fftRows,
              int fftCols,
              fftw_complex *fftData,
              string dirPath,
              string fileName);
/*  Description:  Write FFT magnitude data to file
 *                (currently for debugging purposes)
 *
 *  Inputs:  fftRows - number of rows in FFT data
 *           fftCols - number of columns in FFT data
 *           *fftData - pointer to FFT spectrum
 *           dirPath - path of directory where file is written
 *           fileName - filename of the file written
 *
 *  Output:  File with FFT spectrum magnitudes written
 *           as unsigned integers from [0,65535]
 */

void spaceWrite(int numRows,
                int numCols,
                double *data,
                string dirPath,
                string fileName);
/*  Description:  Write spatial domain data to file
 *
 *      Inputs:  numRows - number of rows in image
 *               numCols - number of columns in image
 *               *data - pointer to image data
 *               dirPath - path of directory where file is written
 *               fileName - filename of the file written
 *
 *      Output:  File with spatial data written as unsigned
 *               integers ranging from [0,65535]
 */

void nonmax(int numRows,
            int numCols,
            double *imgData,
            double *nonMaxImg,
            int minDistX=16,
            int minDistY=16);
/*  Description:  Non-maximal suppression of an image
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           *imgData - pointer to image data
 *
 *  Output:  *nonMaxImg - pointer to image under
 *                        non-maximal suppression
 */

void randperm(const int arraySize,
              int *randArray);
/*  Description:  Random permutation of integers
 *                (Similar functionality to MATLAB's "randperm")
 *
 *  Input:  arraySize - size of the random array
 *
 *  Output:  *randArray - pointer to random array
 */


void randn(int arraySize,
           double *normalRandArray);
/*  Description:  Generates array of Gaussian random
 *                numbers with mean zero and variance one
 *                using the Box-Mueller transformation method
 *                (Similar to MATLAB's "randn")
 *
 *  Input:  arraySize - size of random array
 *
 *  Output:  *normalRandArray - pointer to random array
 */

 void mysort(vector<double> doublevecin,
            vector<double> &doublevec,
            vector<int> &idxvec);
/*  Description:  Sorts data in a vector by smallest value ascending
 *                (double precision data)
 *
 *      Input:  doublevecin - input vector (unsorted)
 *
 *  Outputs:  doublevec - sorted vector
 *             idxvec - indices of original data in sorted vector
 */

template<class T>
```

```
void swap_vals(T& v1,
               T& v2);
/*  Description:  Swap the values of two arguments
 *
 *  Inputs:  v1 - first value
 *           v2 - second value
 *
 *  Outputs:  v1 - second value
 *            v2 - first value
 */

int index_of_smallest(vector<double> invec,
                      int start_index,
                      int vecsize);
/*  Description:  Finds the index of the smallest value of array
 *
 *  Inputs:  invec - input vector
 *           start_index - starting index from which to
 *                         find the smallest value
 *           vecsize - size of input vector
 *
 *  Output:  integer representing index of smallest value
 */
```

2. detectionTools.h

```
#ifndef DETECTIONTOOLS_H
#define DETECTIONTOOLS_H
/*
 *  detectionTools.h
 *
 *  Created by jbarnes on 9/25/05.
 *  Jeff Barnes
 *  jbarnes@umr.edu
 *  Copyright 2005
 *  Airborne Reconnaissance and Image Analysis Laboratory
 *  University of Missouri - Rolla
 *  All rights reserved.
 *
 */
#include<iostream>
#include<string>
#include<cmath>

using namespace std;
#include "fftw3.h"
#include "imageUtils.h"
#include "rx_statistics.h"
```

```
struct targetData
{
        vector<int> rowLocs;
        vector<int> colLocs;
        vector<double> vals;
};
/*  Description:  Structure containing pixel locations
 *                of anomalies and their likelihood values
 */

void singleBandRX(int numRows,
                  int numCols,
                  fftw_complex *tMaskFFT,
                  fftw_complex *mMaskFFT,
                  fftw_complex *cMaskFFT,
                  double *inputImg, double *rxImg);
/*  Description:  RX algorithm for single-band imagery
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           *tMaskFFT - pointer to target mask spectrum
 *           *mMaskFFT - pointer to demeaning mask spectrum
 *           *cMaskFFT - pointer to covariance mask spectrum
 *           *inputImg - pointer to image data
 *
 *  Output:  *rxImg - pointer to RX algorithm output image
 */

 void multiBandRX(int numRows,
                 int numCols,
                 int numBands,
                 fftw_complex *tMaskFFT,
                 fftw_complex *mMaskFFT,
                 fftw_complex *cMaskFFT,
                 double *multiBandImg,
                 double *rxImg);
/*  Description:  RX algorithm for multispectral imagery
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           numBands - number of bands in image
 *           *tMaskFFT - pointer to target mask spectrum
 *           *mMaskFFT - pointer to demeaning mask spectrum
 *           *cMaskFFT - pointer to covariance mask spectrum
 *           *multiBandImg - pointer to image data
 *
 *  Output:  *rxImg - pointer to RX algorithm output image
 */

void multiComplexMult(int inSize1,
                      int inSize2,
                      fftw_complex *in1,
                      fftw_complex *in2,
                      fftw_complex *out);
/*  Description:  Complex multiplication of a single array
 *                of size inSize1 by an array of size inSize2
 *                where inSize1 < inSize2 and  inSize1 evenly
 *                divides inSize2
 *
 *  Inputs:  inSize1 - size of the first array
 *           inSize2 - size of the second array
 *           *in1 - pointer to first array of complex data
 *           *in2 - pointer to second array of complex data
 *
 *  Output:  *out - pointer to output result
 */

 void createRXMasksFFT(int targetRadius,
                      int demeaningRadius,
                      int blankingRadius,
                      int numRows,
                      int numCols,
                      fftw_complex *tMaskFFT,
                      fftw_complex *mMaskFFT,
                      fftw_complex *cMaskFFT);
/*  Description:  Create RX masks in the frequency domain
 *
 *  Inputs:  targetRadius - radius of target mask
 *           demeaningRadius - radius of demeaning maskk
 *           blankingRadius - radial width of blanking region
 *           numRows - rows of data on which RX operates
 *           numCols - columns of data on which RX operates
 *
 *  Outputs:  *tMaskFFT - pointer to target mask spectrum
 *            *mMaskFFT - pointer to demeaning mask spectrum
 *            *cMaskFFT - pointer to covariance data
 */

void multiFFT2(int numRows,
               int numCols,
               int howmany,
               double *input_space,
               fftw_complex *output_freq);
```

```
/*  Description:  Performs forward FFTs on a batch of images
 *               (used to gather target and covariance
 *                data for multispectral RX)
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           howmany - number of images in batch
 *           *input_space - pointer to spatial domain
 *                          input imagery
 *
 *  Output:  *output_freq - pointer to output spectra
 */

 void multiIFFT2(int numRows,
                int numCols,
                int howmany,
                fftw_complex *input_freq,
                double *output_space);
/*  Description:  Performs inverse FFTs on a batch of spectra
 *               (used for multispectral RX)
 *
 *  Inputs:  numRows - number of rows of FFT data
 *           numCols - number of columns of FFT data
 *           howmany - number of spectra in batch
 *           *input_freq - pointer to spectra
 *
 *  Output:  *output_space - pointer to spatial domain data
 */

void calcRXstats(int numRows,
                 int numCols,
                 int numBands,
                 double *targData,
                 double *covData,
                 double *rxImg);
/*  Description:  Calculate RX statistics once the
 *               target and covariance data have been
 *               calculated (multispectral RX)
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           numBands - number of bands in image
 *           *targData - pointer to target-mean data
 *           *covData - pointer to covariance data
 *
 *  Output:  *rxImg - RX algorithm output
 */
```

```
void getTargetLocs(int numRows,
                   int numCols,
                   double *rxImg,
                   targetData &tdata,
                   double threshold = 2.0,
                   int numLocs = 20);
/*  Description:  Gather target locations from RX output
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           *rxImg - pointer to RX output data
 *
 *  Output:  tdata - structure containing row and
 *                   column locations and likelihood
 *                   values of anomalies
 */

#endif
```

3. registrationTools.h

```
#ifndef REGISTRATIONTOOLS_H
#define REGISTRATIONTOOLS_H

/*
 *  registrationTools_1.h
 *
 *  Created by jbarnes on 9/25/05.
 *  Jeff Barnes
 *  jbarnes@umr.edu
 *  Copyright 2005
 *  Airborne Reconnaissance and Image Analysis Laboratory
 *  University of Missouri - Rolla
 *  All rights reserved.
 *
 */

#include <iostream>
#include <cmath>
#include <vector>

using namespace std;
#include "Image.h"
#include "fftw3.h"

struct controlPointData
{
        vector<int> rowLocs1;
        vector<double> rowLocs2;
        vector<int> colLocs1;
        vector<double> colLocs2;
        vector<double> vals;
};
/*  Description:  Structure containing control points
 *                in both the reference and sensed images
 *                along with correlation values
 */

void findCorrsForImg(double *lastImg,
                     double *img,
                     controlPointData &cpdata,
                     int maskHeight=32,
                     int maskWidth=32);
/*  Description:  Find correspondences between a reference
 *                and sensed image using correlation-based
 *                method
 *
 *  Inputs:  *lastImg - pointer to reference image
 *           *img - pointer to sensed image
 *
 *  Output:  cpdata - control point data structure
 */

template<class T>
void cleanControlPointData(controlPointData &cpdata,
                           T minDist);
/*  Description:  If two control points are found in the control
 *                point data and are less than minDist
 *                pixel values in distance, then the control
 *                points with the lesser correlation values
 *                are removed from the list of control points.
 *                The control points are then sorted from
 *                maximum correlation value descending.
 *
 *  Inputs:  cpdata - structure containing control point data
 *           minDist - minimum distance between control points
 *
 *  Output:  cpdata - structure containing control points
 *                    sorted from maximum correlation descending
 */


#endif
```

APPENDIX B.

C++ FUNCTION PROTOTYPES FOR SCALE-INVARIANT FEATURE TRANSFORM

This appendix contains the interface file `SIFTutils.h`. This file includes the function prototypes for the SIFT registration algorithm.

```
#ifndef SIFTUTILS_H
#define SIFTUTILS_H

/*
 *  SIFTutils.h
 *
 *  Created by jbarnes on 10/4/05.
 *  Jeff Barnes
 *  jbarnes@umr.edu
 *  Copyright 2005
 *  Airborne Reconnaissance and Image Analysis Laboratory
 *  University of Missouri - Rolla
 *  All rights reserved.
 *
 */
#include<iostream>
#include<vector>
#include "image.h"

using std::vector;

struct extrema
{
        vector<int> rows;
        vector<int> cols;
        vector<int> scale;
        vector<int> octave;
};
/*  Description:  Structure containing extrema locations
 */

struct peaks
{
        vector<int> rows;
        vector<int> cols;
        vector<double> vals;
};
/*  Description:  Structure containing peaks
 *                found through extrema detection
 */

struct keypoints
{
        vector<double> rows;
        vector<double> cols;
        vector<double> scale;
        vector<int> octave;
        vector<double> orientations;
};
/*  Description:  Structure of keypoint data containing
 *                row location, column location, scale,
 *                octave and orientation
 */

 struct controlPointData_SIFT
{
        vector<double> rowLocs1;
        vector<double> rowLocs2;
        vector<double> colLocs1;
        vector<double> colLocs2;
        vector<double> vals;
};
/*  Description:  Structure with control point data
 *                including row and column locations
 *                in the reference and sensed images
 *                and the value of the distance
 *                between their keypoint descriptors
 *                in feature space
 */

void imageExpand(int numRows_orig,
                 int numCols_orig,
                 double *img,
                 double *expandedImg);
/*  Description:  Image expansion via bilinear interpolation
 *
 *  Inputs:  numRows_orig - number of rows in original image
 *           numCols_orig - number of columns in original image
 *           *img - pointer to image data
 *
 *  Output:  *expandedImg - pointer to expanded image data
 */

void convGaussianFast(int numRows,
                      int numCols,
                      double *inputImg,
                      double *smoothedImg,
                      double sigma);
/*  Description:  Fast 2D convolution with separable
 *                Gaussian kernels
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           *inputImg - pointer to input image
 *           sigma - standard deviation of kernel
 *
 *  Output:  *smoothedImg - result from convolution
 */

 void buildFirstOctave(int numRows,
                       int numCols,
                       int s,
                       double *inputImg,
                       double *dogOctave,
                       double *nextOctaveStart,
                       double *gaussImages);
/*  Description:  Build the first octave of scale-space
 *                and difference-of-Gaussian images
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           s - Lowe's parameter "s"
 *           *inputImg - pointer to input image
 *
 *  Outputs:  *dogOctave - octave of DoG images
 *            *nextOctaveStart - pointer to data used to build next octave
 *            *gaussImages - octave of scale-space
 */

void buildOctave(int numRows,
                 int numCols,
                 int s,
                 int octaveNum,
                 double *prevOctaveStart,
                 double *dogOctave,
                 double *nextOctaveStart,
                 double *gaussImages);
/*  Description:  Build subsequent octaves after the first
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           s - Lowe's parameter "s"
 *           octaveNum - number of the octave to build
 *           *prevOctaveStart - pointer of data used to build octave
 *
 *   Outputs:  *dogOctave - difference-of-Gaussian images for octave
 *             *nextOctaveStart - pointer to data used to build next octave
 *             *gaussImages - pointer to scale-space for octave
 */

void buildPyramid(int numRows,
                  int numCols,
                  int s,
                  int totalOctaves,
                  double *inputImg,
                  double *pyramid,
                  double *gaussImages);
/*  Description:  Build complete scale-space and DoG pyramids
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           s - Lowe's parameter "s"
 *           totalOctaves - number of octaves in pyramid
 *           *inputImage - pointer to original input image
 *
 *  Outputs:  *pyramid - pointer to DoG pyramidd
 *            *gaussImages - pointer to scale-space images
 */

 void findExtrema(int numRows,
                  int numCols,
                  int s,
                  int totalOctaves,
                  double *pyramid,
                  extrema &dogExtrema);
/*  Description:  Find extrema in DoG images with check
 *                of nearest neighbors
 *
 *  Inputs:  numRows - number of rows in image
 *           numCols - number of columns in image
 *           s - scale parameter
 *           totalOctaves - total number of octaves in pyramid
 *           *pyramid - pointer to DoG pyramid
 *
 *  Output:  dogExtrema - structure with extrema in DoG images
 */

void findMaxima(int numRows,
                int numCols,
                double *img,
                peaks &peakList);
/*  Description:  Find maxima in DoG plane of pyramid
 *
 *  Inputs:  numRows - number of rows in octave
 *           numCols - number of columns in octave
 *           *img - pointer to plane in DoG images
 *
 *  Output:  peakList - structure containing peaks in
 *                      a plane of DoG images
 */

 void findMinima(int numRows,
                 int numCols,
                 double *img,
                 peaks &peakList);
/*  Description:  Find minima in DoG plane of pyramid
 *
 *  Inputs:  numRows - number of rows in octave
 *           numCols - number of columns in octave
 *           *img - pointer to plane in DoG images
 *
 *  Output:  peakList - structure containing peaks in
 *                      a plane of DoG images
 */

void refineMax(int rows,
               int cols,
               int octave,
               int scale,
               double *sclPtr,
               peaks &peakList, extrema &pyrExtrema);
/*  Descriptions:  Checks scales adjacent to the scale of the
 *                 extremum for values greater than the
 *                 values in peakList
 *
 *  Inputs:  rows - number of rows in octave
 *           cols - number of columns in octave
 *           octave - current octave
 *           scale - current scale
 *           *sclPtr - pointer to plane of scale space
 *           peakList - list of possible extrema
 *
 *  Output:  pyrExtrema - extrema that survive are placed
 *                        in this structure
 */

 void refineMin(int rows,
                int cols,
                int ocatve,
                int scale,
                double *sclPtr,
                peaks &peakList,
                extrema &pyrExtrema);
/*  Description:  Similar to "refineMax", except
 *                the check is for minima
 */

void refineExtrema(int nRows,
                   int nCols,
                   int s,
                   int numOctaves,
                   double *pyramid,
                   extrema &pyrExtrema,
                   keypoints &keyData);
/*  Description:  Refines the extrema by interpolating their
 *                values and checking if it is above a certain
 *                threshold in the DoG images.  Extrema which
 *                have a large ratio of principal curvatures
 *                are also discarded.
 *
 *  Inputs:  nRows - number of rows in image
 *           nCols - number of columns in image
 *           s - scale parameter
 *           numOctaves - total number of octaves
 *           *pyramid - pointer to DoG pyramid data
 *           pyrExtrema - structure containing list of extrema
 *
 *  Output:  keyData - structure containing keypoints for image
 */

 void getHessianData(int sclRows,
                     int sclCols,
                     double *ptr,
                     double xy[3][3],
                     double xSigma[3][3],
                     double ySigma[3][3]);
/*  Description:  Gather data in DoG pyramid needed
 *                calculate the 3 x 3 Hessian matrix
 *
 *  Inputs:  sclRows - number of rows in octave
 *           sclCols - number of columns in octave
 *           *ptr - pointer to singular location in DoG pyramid
 *
 *  Outputs:  xy - 3 x 3 array of data in x-y direction
 *                (x-y ~ column-row)
 *            xSigma - 3 x 3 array of data in x-sigma direction
 *            ySigma - 3 x 3 array of data in y-sigma direction
 */
```

```
void calcHessian(double xy[3][3],
                 double xSigma[3][3],
                 double ySigma[3][3],
                 double hessian[3][3],
                 double derivative[3]);
/*  Description:  Calculates the Hessian data from
 *                the data gathered in getHessianData
 *
 *  Inputs:  xy - 3 x 3 array of data in x-y direction
 *           xSigma - 3 x 3 array of data in x-sigma direction
 *           ySigma - 3 x 3 array of data in y-sigma direction
 *
 *  Outputs:  hessian - 3 x 3 array representing Hessian
 *                      matrix at specific location
 *            derivate - length 3 row vector of derivatives
 *                       of DoG data w.r.t. x, y, and sigma
 */

void calcOffset(double hessian[3][3],
                double derivatve[3],
                double offset[3]);
/*  Description:  Calculate the interpolated offset from
 *                the Hessian and derivative data
 *
 *  Inputs:  hessian - 3 x 3 array representing matrix of
 *                     Hessian data
 *           derivative - length 3 row vector f derivatives
 *
 *  Output:  offset - length 3 row vector of offsets in the
 *                    x, y, and sigma directions
 */

 void compMagAndOrient(int numRows,
                       int numCols,
                       int s,
                       int totalOctaves,
                       double *gaussImages,
                       double *magnitudes,
                       double *orientations);
/*  Description:  Precompute magnitudes and orientations
 *                over all planes of scale-space
 *
 *  Inputs:  numRows - number of rows in original image
 *           numCols - number of columns in original image
 *           s - scale parameter
 *           totalOctaves - total number of octaves in scale-space
 *           *gaussImages - pointer to scale-space
 *
 *  Outputs:  *magnitudes - pointer to magnitude data
 *            *orientations -pointer to orientation data
 */

void compMagnitudes(int numRows,
                    int numCols,
                    double *sclPtr,
                    double *magPtr);
/*  Description:  Compute magnitude data from single
 *                plane of scale-space
 *
 *  Inputs:  numRows - number of rows in octave
 *           numCols - number of columns in octave
 *           *sclPtr - pointer to plane of scale-space
 *
 *  Output:  *magPtr - pointer to plane of magnitudes
 */

void compOrientations(int numRows,
                      int numCols,
                      double *sclPtr,
                      double *orPtr);
/*  Description:  Similar to "compMagnitudes" except
 *                calculation of orientations
 */

 void assignOrientations(int numRows,
                         int numCols,
                         int s,
                         int totalOctaves,
                         keypoints &keyData,
                         keypoints &keyData2,
                         double *magnitudes,
                         double *orientations);
/*  Description:  Assigns an orientation to a keypoint
 *
 *  Inputs:  numRows - number of rows in original image
 *           numCols - number of columns in original image
 *           s - scale parameter
 *           totalOctaves - total number of octaves
 *           keyData - structure with keypoints
 *           *magnitudes - pointer to magnitude data
 *           *orientations - pointer to orientation data
 *
 *  Output:  keyData - orientation vector of keyData structure
 *                     is computed and filled
 *           keyData2 - any keypoints with two orientations
 *                      have another keypoint created within
 *                      this structure
 */

void fitPolynomialMax(double abscissa[3],
                      double poly[3],
                      double &maxAbscissa);
/*  Description:  Interpolates the maximum location
 *                of the histogram during orientation
 *                assignment
 *
 *  Inputs:  abscissa - size 3 array of abscissas
 *           poly - size 3 array of values at the abscissas
 *
 *  Output:  maxAbscissa - interpolated peak abscissa
 */

 void mergeKeypoints(keypoints &keyData1,
                    keypoints keyData2);
/*  Description:  Merges the keypoints in keyData2
 *                with the keypoints in keyData1.
 *                All keypoints are kept in keyData1.
 */

void buildDescriptors(int numRows,
                      int numCols,
                      int s,
                      int totalOctaves,
                      keypoints &keyData,
                      double *magnitudes,
                      double *orientations,
                      double *descriptors);
/*  Description:  Builds the feature-space descriptors
 *                from the keypoint data and magnitudes
 *                and orientation data
 *
 *  Inputs:  numRows - number of rows in original image
 *           numCols - number of columns in original image
 *           s - scale parameter
 *           totalOctaves - total number of octaves
 *           keyData - structure containing keypoint data
 *           *magnitudes - pointer to magnitude data
 *           *orientations - pointer to orientation data
 *
 *  Output:  *descriptors - pointer to array of keypoint descriptors
 */

 void getDescriptor(double *chipMag,
                    double *chipOr,
                    double *descriptor);
/*  Description:  Construct the descriptor from chips
 *                extracted from the magnitude and
 *                orientation data
 *
 *  Inputs:  *chipMag - pointer to magnitude chip (16 x 16)
 *           *chipOr - pointer to orientation chip (16 x 16)
 *
 *  Output:  *descriptor - pointer to descriptor (1 x 128)
 */

void findMatchingPoints(keypoints keyData_1,
                        keypoints keyData_2,
                        double *descriptors_1,
                        double *descriptors_2,
                        controlPointData_SIFT &cpData);
/*  Description:  Finds matching points in feature space via
 *                brute force search
 *
 *  Inputs:  keyData_1 - keypoints from the reference image
 *           keyData_2 - keypoints from the sensed image
 *           *descriptors_1 - reference image descriptors
 *           *descriptors_2 - sensed image descriptors
 *
 *  Output:  cpData - structure containing control point data
 */

 void SIFT(int nRows,
           int nCols,
           double *currData,
           double *lastData,
           controlPointData_SIFT &cpData);
/*  Description:  SIFT algorithm for registration
 *
 *  Inputs:  nRows - number of rows in input images
 *           nCols - number of columns in imput images
 *           *currData - pointer to sensed image
 *           *lastData - pointer to reference image
 *
 *  Output:  cpData - structure containing control point data
 */

#endif
```

BIBLIOGRAPHY

[1] R. Siegel, "Landmine Detection," *IEEE Instrumentation and Measurement Magazine*, vol. 5, pp. 2805 – 2809, December 2002.

[2] S. B. Campana, *The Infrared and Electro-Optical Systems Handbook, Volume 5: Passive Electro-Optical Systems.* Bellingham, Washington: SPIE Optical Engineering Press, 1993.

[3] G. Maksymonko and K. Breiter, "ASTAMIDS Minefield Detection Performance at Aberdeen Proving Ground Test Site," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets*, vol. 3079, pp. 726 – 737, 1997.

[4] Department of Defense - ESTCP Cost and Performance Report, "Assessment of the Remote Minefield Detection System (REMIDS)," September 1999.

[5] N. H. Witherspoon, J. H. Holloway, K. S. Davis, R. W. Miller and A. C. Dubey, "The Coastal Battlefield Reconnaissance and Analysis (COBRA) Program for Minefield Detection," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets*, vol. 2496, pp. 500 – 508, 1995.

[6] H. H. Suzukawa and M. S. Farber, "Long-Range Airborne Detection of Small Floating Objects," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets*, vol. 2496, pp. 193 – 205, 1995.

[7] J. Simrad and P. Mathieu, "Airborne Far IR Minefield Imaging System (AFIRMIS): Description and Preliminary Results," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets III*, vol. 3392, pp. 84 – 95, 1998.

[8] H. T. Haskett and D. A. Reago, "Identification of Optimal Bands in the 3-5 μm Region for Lightweight Airborne Mine Detection System (LAMD)," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets IV*, vol. 4394, pp. 246 – 257, 2001.

[9] Q. A. Holmes, "Adaptive Multispectral CFAR Detection of Landmines," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets*, vol. 2496, pp. 421 – 432, 1995.

[10] X. Descombes, M. Sigelle and F. Preteux, "Estimating Gaussian Markov Random Field Parameters in a Nonstationary Framework: Application to Remote Sensing Imaging," *IEEE Transactions on Image Processing*, vol. 8, pp. 490 – 503, April 1999.

[11] M. H. C. Law, M. A. T. Figueiredo and A. K. Jain, "Simultaneous Feature Selection and Clustering Using Mixture Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1154 – 1166, September 2004.

[12] D. Menon, S. Agarwal, R. Ganju and C. W. Swonger, "False Alarm Mitigation and Feature Based Discrimination for Airborne Mine Detection," *Proceedings of the SPIE - Detection and Remediation of Mines and Minelike Targets IX*, vol. 5415, pp. 1163 – 1173, April 2004.

[13] S. L. Earp, "A performance model for the detection of patterned minefields," tech. rep., SLE Associates, Elliott City, MD, July 2002.

[14] N. Malloy, "A linear pattern detector," tech. rep., Multisensor Science, LLC, Elliott City, MD, June 2003.

[15] L. G. Brown, "A Survey of Image Registration Techniques," *ACM Computing Surveys*, vol. 24(4), pp. 325 – 376, 1992.

[16] S. Mallat, *A Wavelet Tour of Signal Processing.* San Diego: Academic Press, 2001.

[17] A. Skodras, C. Christopoulos and T. Ebrahimi, "The JPEG 2000 Still Image Compression Standard," *IEEE Signal Processing Magazine*, vol. 18, pp. 36 – 58, September 2001.

[18] A. Trang, "Comparative Mine Detection Performance Results for the RX Baseline Algorithm, the Wavelet-based JPEG 2000 Data Compression, and the Region-Of-Interest Wavelet Difference Reduction (WDR) Compression Schemes," tech. rep., Night Vision and Electronic Services Directorate, Countermine Division, Ft. Belvoir, VA, August 2004.

[19] T. Lindeberg, "Scale-space Theory: A Basic Tool for Analysing Structures at Different Scales," *Journal of Applied Statistics*, vol. 21(2), pp. 224 – 270, 1994.

[20] D. G. Lowe, "Distinctive Image Features from Scale-invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91 – 110, 2004.

[21] M. R. Rao and A. S. Bopardikar, *Wavelet Transforms: Introduction to Theory and Applications.* Reading, MA: Addison Wesley, 1998.

[22] J. C. Goswami and A. K. Chan, *Fundamentals of Wavelets: Theory, Algorithms, and Applications.* New York: John Wiley and Sons, 1999.

[23] X. P. Zhang and M. D. Desai, "Segmentation of Bright Targets Using Wavelets and Adaptive Thresholding," *IEEE Transactions on Image Processing*, vol. 38, pp. 1020 – 1030, July 2001.

[24] F. A. Sadjadi, "Infrared Target Detection with Probability Density Functions of Wavelet Transform Subbands," *Applied Optics*, vol. 43, pp. 315 – 323, January 2004.

[25] I. S. Reed and X. Yu, "Adaptive Multi-band CFAR Detection of an Optical Pattern with Unknown Spectral Distribution," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, pp. 1760 – 1170, October 1990.

[26] J. Y. Chen and I. S. Reed, "A Detection Algorithm for Optical Targets in Clutter," *IEEE Transactions on Aerospace Electronics Systems*, vol. AES-23, pp. 46 – 59, January 1987.

[27] A. Margalit, I. S. Reed and R. M. Gagliardi, "Adaptive Optical Target Detection Using Correlated Images," *IEEE Transactions on Aerospace Electronics Systems*, vol. AES-21, pp. 394 – 405, May 1985.

[28] S. Schweizer and J. M. F. Moura, "Efficient Detection in Hyperspectral Imagery," *IEEE Transactions on Image Processing*, vol. 10, pp. 584 – 597, April 2001.

[29] D. Menon, "A Knowledge Based Architecture for Airborne Minefield Detection," Master's thesis, University of Missouri - Rolla, August 2005.

[30] H. Ramachandran, "Background Modeling and Algorithm Fusion for Airborne Landmine Detection," Master's thesis, University of Missouri - Rolla, May 2004.

[31] X. Yu, I. S. Reed, W. Kraske and A. D. Stocker, "A Robust Adaptive Multi-Spectral Object Detection by Using Wavelet Transform," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 141 –144, March 1992.

[32] Q. Zhang and A. Beneviste, "Wavelet Networks," *IEEE Transactions on Neural Networks*, vol. 3, pp. 889–898, 1992.

[33] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for Science & Engineering.* New York: McGraw Hill, 2000.

[34] L. G. Shapiro and G. C. Stockman, *Computer Vision.* Upper Saddle River, New Jersey: Prentice Hall, 2001.

[35] J. Beis and D. G. Lowe, "Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces," *Conference on Computer Vision and Pattern Recognition, Puerto Rico*, pp. 1000 – 1006, 1997.

[36] K. Mikolajczyk and C. Schmid, "An Affine Invariant Interest Point Detector," *European Conference on Computer Vision, Copenhagen*, pp. 128 – 142, 2002.

[37] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Fourth Alvey Vision Conference, Manchester*, pp. 147 – 151, 1988.

VITA

Jeffrey D. Barnes [redacted] In December 2003, he received his Bachelor of Science in Electrical Engineering from the University of Missouri - Rolla. He joined the Master of Science program in Electrical Engineering at the University of Missouri - Rolla in the spring of 2004. While pursuing his graduate degree, he was supported by the Department of Electrical Engineering as a Graduate Research Assistant. He received his Master of Science in Electrical Engineering in May 2006. His technical interests are primarily in digital signal processing and image processing. His non-technical interests include studying current events, reading, and playing the piano. In January 2006 he began work as an electrical engineer in the Integrated Systems division of L-3 Communications, a defense contractor in Greenville, TX.