A MARINE RECREATIONAL VESSEL RECONNAISSANCE SYSTEM UTILIZING IKONOS IMAGERY

KEVIN PEGLER

October 2004

GEODESY AND GEOMATICS ENGINEERING
UNB

TECHNICAL REPORT
NO. 226

A MARINE RECREATIONAL VESSEL RECONNAISSANCE SYSTEM UTILIZING IKONOS IMAGERY

Kevin H. Pegler

Department of Geodesy and Geomatics Engineering
University of New Brunswick
P.O. Box 4400
Fredericton, N.B.
Canada
E3B 5A3

October 2004

© Kevin Pegler 2004

PREFACE

This technical report is a reproduction of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Geodesy and Geomatics Engineering, October 2004. The research was supervised by Dr. David Coleman, and funding was provided by the GEOIDE (Geomatics for Informed Decisions) Network of Centres of Excellence.

As with any copyrighted material, permission to reprint or quote extensively from this report must be received from the author. The citation to this work should appear as follows:

Pegler, K. (2004). *A Marine Recreational Vessel Reconnaissance System Utilizing IKONOS Imagery.* Ph.D. dissertation, Department of Geodesy and Geomatics Engineering Technical Report No. 226, University of New Brunswick, Fredericton, New Brunswick, Canada, 205 pp.

ABSTRACT

This dissertation investigates the ability of IKONOS imagery to detect small recreational boats. To accomplish this, automatic target detection software called MRV Recon has been developed which makes use of a weighted Euclidean distance metric.

To test the detection accuracy of MRV Recon, a dataset was created by gathering position and attribute data for 53 recreation vessel targets within Cadboro Bay, British Columbia, Canada. IKONOS imagery was collected in May 2003.

The overall detection accuracy was 77 %. The targets were broken down into two categories: A) less than 6 m in length, and B) greater than 6 m long. The detection rate for the category B targets was 100%, while the detection rate for the category A targets was 61%. It is important to note that some category A targets were selected specifically to test the detection limits of MRV Recon. The smallest target detected was 2.2 m long and 1.1 m wide. The analysis also revealed that the ability to detect targets between 2.2 m and 6 m long was diminished if the target was a dark colour.

It has been demonstrated that MRV Recon will provide the Canadian Coast Guard with a unique and effective tool for gathering crucial data on recreational vessels.

ACKNOWLEDGEMENTS

First and foremost, I would like to give thanks to my supervisor and friend, Dr. David J. Coleman, P.Eng. Dr. Coleman's boundless energy, enthusiasm, and dedication to Geomatics Engineering set a very high standard for me. I know I am a better man now than I was when I began this journey largely because of his tutelage. I am deeply indebted to him.

I would also like to thank the support of the Canadian Coast Guard and the federally supported Network of Centres of Excellence, GEOIDE (Geomatics for Informed Decisions). The work is part of GEOIDE project #ENV 60 – Marine Geomatics and Risk Analysis.

I believe that the Department of Geodesy and Geomatics Engineering is arguably one of the greatest departments of its kind in the world. Of course, a department is only as great as the people in it and GGE has some of the best. I would like to show my appreciation to my departmental colleagues (and others) whose advice, knowledge, or council I sought: Dr Yun Zhang, Dr. Peter Dare, Dr. Ron Pelot (Dalh), Dr. Peter Keller (UVic), Dr. Don Kim, Dr. Sunil Bisnath, Mr. Jonathon Beaudoin, Mr. Tomas Beran, Mr. George Dias, Mr. David Fraser, Captain Michael Hogan, Captain Travis Wert, and Ms. Chantelle Delorme-Lafontaine (RSI). Thank you all so very much.

I couldn't have done this degree without the sacrifice of my family. I would like to acknowledge my parents for supporting the dreams of their only child. Thank you.

For my children, Stephanie and Drew, let this accomplishment serve as an example to you that no matter what the goal and the difficulties in achieving it, you *can* achieve it

by remaining focused, working hard, and maintaining a balanced approach to life. Reach for the stars, kids!

Finally, I want to thank my wife Shirley, for keeping the household running, bearing and raising the children, putting up with me, maintaining her own career, and being so supportive to me that mere words cannot describe. Thank you, with all my heart.

~

TABLE OF CONTENTS

Page

Abstract ii
Acknowledgements iii
Table of Contents v
List of Figures viii
List of Tables x
List of Abbreviations xi

1. INTRODUCTION 1
    1.1 Selection of Topic 1
    1.2 Articles Format Thesis 3
    1.3 Outline of Chapters 4

2. THE POTENTIAL FOR USING VERY HIGH SPATIAL RESOLUTION IMAGERY FOR MARINE SEARCH AND RESCUE 7
    2.1 Introduction 7
    2.2 Background Research 8
    2.3 Methodology for Developing a Marine Recreational Reconnaissance System 14
    2.4 Results 15
    2.5 Further Work 18
    2.6 Validation of the Reconnaissance System 18
    2.7 Conclusion 18
    2.8 Acknowledgements 20
    2.8 References 20

3. EXPERIMENTAL DESIGN 22
    3.1 Introduction 22
        3.1.1 Previous Target Detection Experiment 23
    3.2 Location 26
    3.3 Imagery Acquisition 27
    3.4 Initial Experimental Design 29
    3.5 Design Limitations 34
    3.6 Final Experimental Design 34
    3.7 Data Collection 37
    3.8 Target/Image Position Error Budget 39
    3.9 Recommendations 42
    3.10 Summary 43

4. IMAGE PREPROCESSING 44
    4.1 Introduction 44
    4.2 Pre-processing 46
        4.2.1 Previous Target Detection Experiment 47

4.2.2 Evaluation of Results 49
4.2.3 Preprocessing Details 50
4.3 Procedure 50
4.3.1 Masking Procedure 53
4.4 Preprocessing Results 55
4.5 Summary 56

5. COMPARISON OF MAXIMUM DISTANCE METRICS FOR USE IN THE REMOTE SENSING OF SMALL TARGETS 57
5.1 Introduction 57
5.2 Previous Research 59
5.3 Methodology 61
5.3.1 Data 62
5.3.2 Preprocessing 62
5.3.3 Target Selection 64
5.4 Results 67
5.4.1 3x3 Kernel Results 67
5.4.2 5x5 Kernel Results 72
5.4.3 Overall Detection Results 74
5.5 Weighted Euclidean Distance 74
5.5.1 WED 3x3 Kernel Results 75
5.5.2 Detection Results for WED 76
5.6 Proximity Analysis 78
5.6.1 Analysis on Re-sampled Data 80
5.6.2 Proximity Analysis on Re-sampled Data 83
5.7 Conclusions 84
5.8 Acknowledgements 86
5.8 References 86

6. AUTOMATIC SMALL RECREATIONAL VESSEL DETECTION USING IKONOS DATA 89
6.1 Introduction 89
6.2 Objective 90
6.3 Background 91
6.4 Study Site 94
6.5 Data and Preprocessing 95
6.6 Methodology 97
6.7 Results of the Blind Test 101
6.8 Software Enhancements 102
6.9 Final Results and Analysis 102
6.10 Conclusions 113
6.11 References 115

7. RESULTS, CONCLUSIONS, AND FUTURE RESEARCH 116
7.1 Results 116
7.2 Conclusions 118

7.3 Future Research 119
7.4 Accomplishments and Contributions 121

REFERENCES 123

APPENDICES

I. MRV Recon C CODE 126
II. SOFTWARE DESIGN 1 175
III. SOFTWARE DESIGN 2 186

Vita

LIST OF FIGURES

Page

1.1 Organizational chart for this dissertation 5

2.1 Outlier Detection Method – for 3 band imagery 11
2.2 IKONOS Imagery, Fredericton Yacht Club, Fredericton, New Brunswick, Canada. 12
2.3 Proposed Methodology for the Development of a Recreational Vessel Reconnaissance System 14
2.4 Left: Resampled, pseudo-coloured panchromatic band. Right: Resulting "target" channel 16
2.5 Resampled, pseudo-coloured panchromatic band merged with the "target" channel 17
2.6 Digital numbers surrounding a single targeted vessel 17

3.1 Cadboro Bay and RVYC, Vancouver Island, British Columbia, Canada 27
3.2 Mooring a Category A Target, Cadboro Bay, B.C. 36
3.3 Category B targets at RVYC 37
3.4 IKONOS data with associated ground truth 38
3.5 Anchor Geometry 40
3.6 Circular anchor error 40

4.1 IKONOS imagery for testing pre-processing 47
4.2 The manner in which distances were measured off of the SNB reference line .. 48
4.3 Model that performs the unsupervised classification 51
4.4 Unsupervised classification results using the ISODATA classifier 52
4.5 The classified image after being processed by the SIEVE module 53
4.6 The image resulting after running the "Unsuper1" model and "MODEL"........ 54
4.7 Results of preprocessing, IKONOS imagery, St. John River 55
4.8 Results Pre-processing on IKONOS imagery, Cadboro Bay 56

5.1 IKONOS imagery, Fredericton New Brunswick, Canada 62
5.2 PCI Modeler preprocessing application 63
5.3 Preprocessing results and test targets 64
5.4 Spectral response curve for target *Alpha* 65
5.5 Spectral response curve for target *Bravo* 65
5.6 Spectral response curve for target *Charlie* 66
5.7 Spectral response, target *Charlie*, 3x3 kernel 69
5.8 Spectral response, target *Alpha*, 3x3 kernel 70
5.9 Spectral response target *Alpha* 5x5 kernel 73
5.10 Detection results using WED with a 3x3 kernel 78
5.11 Detection results using WED with a 5x5 79
5.12 Proximity Analysis 80
5.13 Proximity Analysis: Re-sampled data 83
5.14 WED, 3x3, re-sampled data 84

6.1 MRV Recon GUI 91
6.2 The study site: Cadboro Bay, Vancouver Island, British Columbia, Canada 95
6.3 IKONOS Panchromatic Image of Cadboro Bay 96
6.4 Flowchart for MRV Recon 98
6.5 Accuracy Assessment Blind Test 101
6.6 Positive Detections vs. Length 104
6.7 Positive Detection vs. Area 105
6.8 Reflectance Cross-sections for Targets Golf and Bravo 106
6.9 False Positives Response Curves 108
6.10 False Negatives Response Curves 108
6.11 Background Response Curves 109
6.12 Comparison of Cross-section for Target Alpha and India 110
6.13 Cross-section through target Charlie and its wake 110
6.14 Comparison Between Measured and True Length for Targets 112
6.15 Comparison Between Measured and True Width for Targets 113

7.1 Detection results 117

LIST OF TABLES

Page

3.1 Binomial Distribution 32
3.2 Initial Budget 33
3.3 Target attribute data collected by volunteers 35

4.1 Evaluation of Results 49

5.1 Outlier frequency and distance from mean for 3x3 kernel 68
5.2 Covariance target *Charlie*, 3x3 kernel 68
5.3 Covariance, target *Alpha*, 3x3 kernel 71
5.4 Covariance target *Bravo*, 3x3 kernel 71
5.5 Inverted covariance (weight) matrix used in Mahalanobis target *Bravo* 72
5.6 Outlier frequency and distance from mean for 5x5 kernel 73
5.7 Detection Results 74
5.8 Distance from mean using Weighted Euclidean Distance 76
5.9 Distance from mean using Weighted Euclidean Distance 76
5.10 WED Detection Results 77
5.11 Detection Results for re-sampled data 81
5.12 WED results for re-sampled data 82

6.1 Output from MRV Recon 100
6.2 MRV Recon Accuracy Assessment 103
6.3 Categorized Accuracy Assessment 103
6.4 Reflectance Values and Covariance Matrix for Target Golf 106
6.5 Reflectance Values and Covariance Matrix for Target Bravo 107

7.1 Binomial distribution for determination of the true error rate 117
7.2 Research Objectives 121

LIST OF ABBREVIATIONS

GEG	Geographical Engineering Group
GIS	Geographic Information System
NCE	National Centre of Excellence
RADAR	Radio Detection and Ranging
MRV Recon	Marine Recreational Vessel Reconnaissance
RCC	Rescue Coordination Centre
CCG	Canadian Coast Guard
RCC	Rescue Coordination Centre
DND	Department of National Defense
SAR	Search and Rescue
MARIS	Maritime Activity and Risk Investigation System
PWC	Personal Watercraft
VHSR	Very High Spatial Resolution
HIS	Hyperspectral Imaging
CCGA	Canadian Coast Guard Auxiliary
GEOIDE	Geomatics for Informed Decisions
DN	Digital Number
I/O	Input /Output
RVYC	Royal Victoria Yacht Club
AOI	Area of Interest
TLE	Two Line Element
CASARA	Civil Air Search and Rescue Association
GPS	Global Positioning System
BSF	Background Suppression Filtering
NIR	Near-Infrared
WED	Weighted Euclidean Distance
TR	Threshold Ratio
PAN	Panchromatic

CHAPTER 1
INTRODUCTION

> *Something appears to be new only because it was previously unknown. I have found that the more I learn the more I realize how little I know.*
>
> \- Cornelius Shields

The thesis of this dissertation is that IKONOS satellite imagery can be used to reliably detect and characterize small recreational vessels. This thesis is part of a larger project objective being to develop a reconnaissance system for small recreational vessels in support of activity assessment for search and rescue.

1.1 Selection of Topic

How was such a topic selected? The Geographical Engineering Group (GEG) was approached by a group of researchers from Dalhousie University who were using risk modeling and Geographic Information Systems (GIS) for aiding the Canadian Coast Guard (CCG) in marine search and rescue (SAR). The principal researcher in a National Centre of Excellence (NCE), Dr. Ronald Pelot, GEOIDE research project asked Dr. David Coleman of the GEG, to join the research team. Dr. Pelot wanted the partnership because of a need for expertise in Geomatics Engineering.

Meetings were held in the summer of 2000 at the Department of Industrial Engineering, Dalhousie University to discuss possible veins of research that could be explored by GEG. Industrial engineers are experts, among other things, in risk modeling and Dr. Ronald Pelot and his team had already done a significant amount of work in developing the risk model used to predict where future marine SAR incidents would occur. The discussions revealed many possible topic areas from which any number of Ph.D. dissertations could be developed.

However, one topic was particularly intriguing. Dr. Pelot and his team completed that previous March a study on “Recreational/Tourism Marine Activity Assessment in the Bay of Fundy” [Pelot et al., 2000]. It outlined the need for accurate information on the number, locations, and kinds of recreational vessel activities in Canadian waters. It looked at different methods for gathering this information including: “satellite image tracking, directed aerial observation, fishery surveillance, land-based RADAR tracking, directed surface-level observation, and surveys” [Pelot et al., 2000].

In discussing the potential for *satellite image tracking* to gather information on recreational vessels the study stated, “… the resolution of available satellite images is inadequate for this purpose. “ Further, it went on to say:

> “Canadian satellite images cannot detect anything less than 7-9 metres in length; this is unacceptable since a large fraction of the recreational vessel population is less than nine metres. Certain American satellite images, however, are more powerful (i.e. resolution of several feet or less) although the quality is unknown. Many boating characteristics may be unobtainable unless the resolution and quality is excellent. Finally, obtaining radar images would be very expensive, and some may be inaccessible for national security purposes” [Pelot et al., 2000].

The statements above were the genesis of this research into the development of a marine recreational vessel reconnaissance system (MRV Recon). It was felt that the assertion, “the resolution of available satellite images is inadequate for this purpose” could be challenged. In particular, the new commercially available IKONOS imagery could be used to accurately detect vessels 7-9 metres long. Further, it was believed that civilian technology could be developed to challenge the notion that this kind of reconnaissance technology mostly exists within the military.

Just how challenging the reconnaissance would be wasn’t fully realized until after a meeting with the Director of the Victoria Rescue Coordination Centre (RCC). Mr. John Palliser was the director of the joint CCG and Department of National Defense (DND) centre that coordinates all the SAR platforms (boats, aircraft, hovercraft etc.) for the west coast of Canada. In that interview, Mr. Palliser confirmed the need for better information on recreational boating. However, he emphasized the importance of being able to detect very small vessels such as personal watercraft and sea kayaks. He indicated there was increasing trend in the number of SAR incidents surrounding these vessels and the cost associated with this trend [Palliser, 2000].

Automatically detecting vessels 6-9 metres in length using IKONOS imagery was difficult. However, finding a vessel a few metres long that is approaching the best resolution of IKONOS (1m panchromatic, 4m multispectral) was a much greater challenge. Recognizing the importance of this problem, a proposal was written and defended. As was stated at the outset, the thesis was that IKONOS satellite imagery can be used to reliably detect and characterize small recreational vessels. In addition, it was proposed that a reconnaissance system for small recreational vessels in support of activity assessment for search and rescue could be developed.

1.2 Articles Format Dissertation

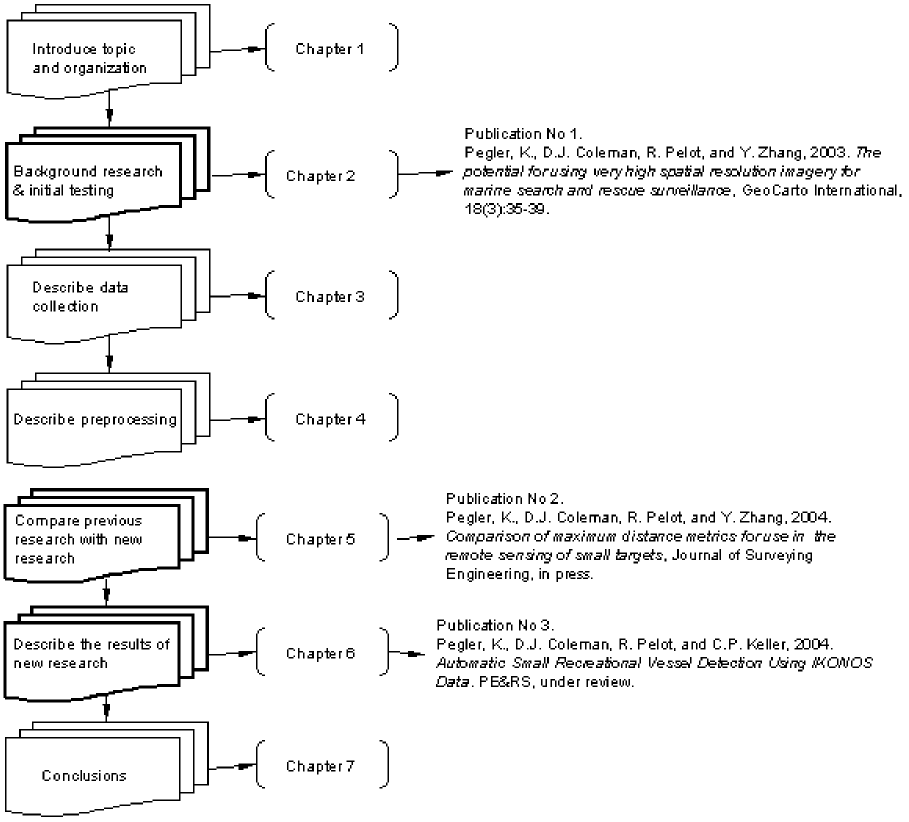
This dissertation subscribes to an articles format as approved by the School of Graduate Studies, University of New Brunswick in 2004.

1.3 Outline of Chapters

Figure 1.1 illustrates the organization of this dissertation. There are three chapters containing the required journal publications. These chapters are prefaced with a small introduction and are reformatted to meet the guidelines for thesis preparation by the School of Graduate Studies.

Chapter 2 is a journal chapter, originally published in the journal GeoCarto International [Pegler et al., 2003] describing the potential for using IKONOS imagery for marine search and rescue. It contains the background research. Specifically, it looks at the body of research containing vessel detection. Further, it investigates associated research in remote sensing as applied to search and rescue, and small target detection. The various themes are compared and contrasted. Some particularly interesting and applicable research in small target detection using hyperspectral imagery is discussed in detail. It forms the foundation for where this research began. Finally, Chapter 2 describes the duplication of the above research and tests its suitability to this problem.

Chapter 3 is a bridging chapter describing the experiment design required for the data collection. It precedes Chapter 4 because it is important to know how the data was collected before describing the necessary preprocessing. For this research it was decided not to use synthetic data, as is common in target detection and computer vision work, in favour of creating a dataset containing real targets. The chapter describes the reasons for choosing the study site. It lays out the initial design of the experiment and the statistics behind that design. It goes on to describe limitations and the subsequent final experiment design. Chapter 3 concludes with a summary and recommendations to others who might undertake such a project.


Introduce topic and organization
Chapter 1
Background research & initial testing
Chapter 2
Describe data collection
Chapter 3
Describe preprocessing
Chapter 4
Compare previous research with new research
Chapter 5
Describe the results of new research
Chapter 6
Conclusions
Chapter 7

Publication No 1.
Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang, 2003. *The potential for using very high spatial resolution imagery for marine search and rescue surveillance*, GeoCarto International, 18(3):35-39.

Publication No 2.
Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang, 2004. *Comparison of maximum distance metrics for use in the remote sensing of small targets*, Journal of Surveying Engineering, in press.

Publication No 3.
Pegler, K., D.J. Coleman, R. Pelot, and C.P. Keller, 2004. *Automatic Small Recreational Vessel Detection Using IKONOS Data*. PE&RS, under review.

Figure 1.1 Organizational chart for this dissertation.

The work performed in preprocessing the imagery is described in Chapter 4 – another bridging chapter. The major preprocessing step is the masking out of the upland portions of the imagery. The preprocessing methodology was developed by a student as part of her undergraduate Technical Report [Munroe, 2003]. The author supervised the student.

Chapter 5, “Comparing of Maximum Distance Metrics for use in the Remote Sensing of Small Targets”, is the second journal chapter. It contains the principal new engineering research for this thesis. A new distance metric is tested and compared with the distance metrics used in the original research (as described in Chapter 2).

Chapter 6, Automatic Small Recreational Vessel Detection Using IKONOS Data, is a journal chapter describing the results of the blind test. It describes the findings and then some minor enhancements to the software. The test is run again, and the results are compared with the blind test. Several targets are investigated in depth. The chapter ends with conclusions regarding the results.

The final chapter summarizes the entire work, makes recommendations for future research and then ends by drawing conclusions.

This dissertation contains three appendices. Appendix I, contains the C code comprising MRV Recon.

Appendix II, "Software Design 1" outlines the re-creation and implementation in the C language of the foundation research. It is the first of two appendices that describes the design, constraints, hurdles, and internal criticism surrounding the over 2500 lines of code that comprise MRV Recon.

Appendix III is entitled "Software Design II". It is the second appendix describing development of the most time-consuming portion of the research, the programming of MRV Recon. The appendix outlines the design of the software. The major modules and functions are described. It outlines the design, constraints, hurdles, and internal criticism and problems encountered in the programming. Finally, the appendix concludes with recommendations, and future improvements to MRV Recon.

CHAPTER 2

THE POTENTIAL FOR USING VERY HIGH SPATIAL RESOLUTION IMAGERY FOR MARINE SEARCH AND RESCUE RECONNAISSANCE[1]

This chapter presents the first journal paper for this research and was published in GeoCarto International, by Pegler et al. [2003]. This paper describes the background research performed and in particular research into a “spatio-spectral” template by Subramanian and Gat [1998] that could potentially provide the foundation for this research. Further, the duplication and testing of the spatio-spectral template is described. The senior author performed the research and writing for this article. The remaining authors generously supplied both advice and editing.

The paper makes reference to SARMAP. SARMAP (as it was then called), is now known as a Maritime Activity and Risk Investigation System (MARIS). Additionally, readers should be aware that the acronymn SAR stands for *search and rescue* (commonly used by most used by most civilian and military agencies performing rescue operations) and is not to be confused for meaning synthetic aperature RADAR.

2.1 Introduction

Of the many activities taking place on Canadian waters, recreational boating represents one of the highest incident groups [Pelot, 2000]. Moreover, Canadian Coast Guard (CCG) personnel have expressed concern over the recent trend in increasing search and rescue incidents of personal watercraft (PWC) and so-called adventure tourism [Palliser, 2000]. Adventure tourism can include: sea kayaking, whale watching,

---

[1] Reprinted by permission of *GeoCarto International*, Vol. 18, No. 3, 2003.

open dinghy cruising, and single handed yacht racing. Researchers at Dalhousie undertook an assessment of recreational marine activity in a pilot study for the Bay of Fundy [Pelot, 2000]. The report stated:

> '*Current information on recreational boating is very sparse, so additional data are required to develop a robust model'* and further, *'…due to the lack of readily available information and data on recreational boating exposure, there is a need to define a method to amass a substantial amount of knowledge on all aspects of recreational marine activity, particularly private boating*' [Pelot, 2000].

As a result of the lack of information on recreational boating activities, the objectives of this research are threefold:

1) To review the existing research on the use of remote sensing in marine recreational vessel reconnaissance and search and rescue.
2) To explore the potential for a Marine Recreational Vessel Reconnaissance system.
3) To implement and test a preliminary system, based on the background research, to ascertain the feasibility of the project.

2.2 Background Research

There is a significant overlap between research regarding marine search and rescue and "target" vessel detection. However, most of the search and rescue research is focused on detection of lost vessels as opposed to military applications [Kruzins et al., 1998 ; Hongyan and Shiyi, 1995].

This application is distinct from the current general body of research involving remote sensing in search and rescue because it is not reactive in nature. Remote sensing is mostly used in marine search and rescue to assist in finding lost vessels in an emergency scenario. This is often described as ‘beaconless remote sensing’ [Wallace et al., 1998]. Beaconless remote sensing is utilized when a locator beacon, such as an Emergency Position Indicating Radio Beacon (EPIRB), has failed to function in a crisis.

This application is proactive in nature. In this model, the remotely sensed data is being used to predict and map the likely locations of search and rescue incidents. There is no requirement to have near real-time data because there is not an immediate risk to human lives. All that is required is the ability to passively monitor recreational boating activities to help in the prediction of future SAR incidents to ensure better strategic and tactical planning by CCG personnel on base location and resource allocation.

Currently, much of the research focus in vessel detection is on the use of Synthetic Aperture RADAR in search and rescue [Gilliam et al., 1999; Kruzins et al., 1998 ; Wallace et al., 1998]. It is relied heavily upon in search and rescue due to its all-weather capabilities. Quite often marine SAR incidents occur during poor weather conditions, and in those scenarios all-weather imaging is a significant benefit.

Until recently, RADAR remote sensing provided the best balance between resolution and all-weather capability. However, in the fall of 1999, the first Very High Spatial Resolution (VHSR) commercial optical remote sensing satellite was launched – IKONOS. Researchers suggest that the use of the new high resolution remote sensing satellites show great promise for SAR applications [Wallace et al., 1998]. Moreover, the use of high resolution optical imagery in the proposed monitoring application reduces the significance of its inherent weakness - its lack of all-weather capability. If an area cannot be monitored for recreational vessel activity due to poor weather, the data can be captured on a subsequent satellite pass. There is little need of a near real-time object detection capability in this application as exists in other SAR applications where there is an immediate risk to human life.

There is some interesting research being focused on the use of high resolution hyperspectral imaging (HIS) for search and rescue [Subramanian and Gat, 1998]. This methodology seems applicable to the problem at hand. This system uses a single sensor that can simultaneously gather data in several hundred very narrow bands of the EM spectrum, as opposed to other systems that gather data from different portions of the EM spectrum using separate sensors.

The advantage of producing an extremely detailed spectral signature for each pixel using HIS is that sub-pixel size objects that are otherwise unresolved in conventional imagery can be detected [Subramanian and Gat, 1998]. In fact, the researchers demonstrated that objects as small as 1/10 of an image pixel, having a resolution of three metres, can be detected. The researchers also suggested that the techniques could be employed with VHSR imagery. However, the reduction in spectral resolution by moving from hyperspectral to multispectral detectors likely means a corresponding decrease in ability to detect smaller objects. If, however, the above techniques are tailored for use in conjunction with one metre resolution VHSR imagery in a 'simplified' marine environment, perhaps small recreational vessels could be detected.

The spatio-spectral template developed by Subramanian and Gat [1998] for sub-pixel object detection is a simple nonparametric solution using 'local image statistics based on spatio-spectral considerations'. It is a two stage process: (1) examination of local statistics and outlier selection and (2) object (or vessel) selection.

In the first stage, the image cube, comprising the separate images of the various bands recorded by the sensor, is scanned by a 3x3 kernel. Of course, the kernel size can be adjusted for different scenarios. The mean vector for all layers of the image cube is calculated. As shown in Figure 2.1, the distance from the pixel currently in the centre of the kernel to the mean is also calculated. This process is repeated for the entire image (shown in Figure 2.2). Using this method, individual outliers (possible target vessels) are selected if they exceed a preset threshold. This threshold ratio is defined as, 'the ratio of the distance from the individual pixel to the mean, divided by the standard deviation of the distances for all nine pixels', within the 3x3 kernel [Subramanian and Gat, 1998].

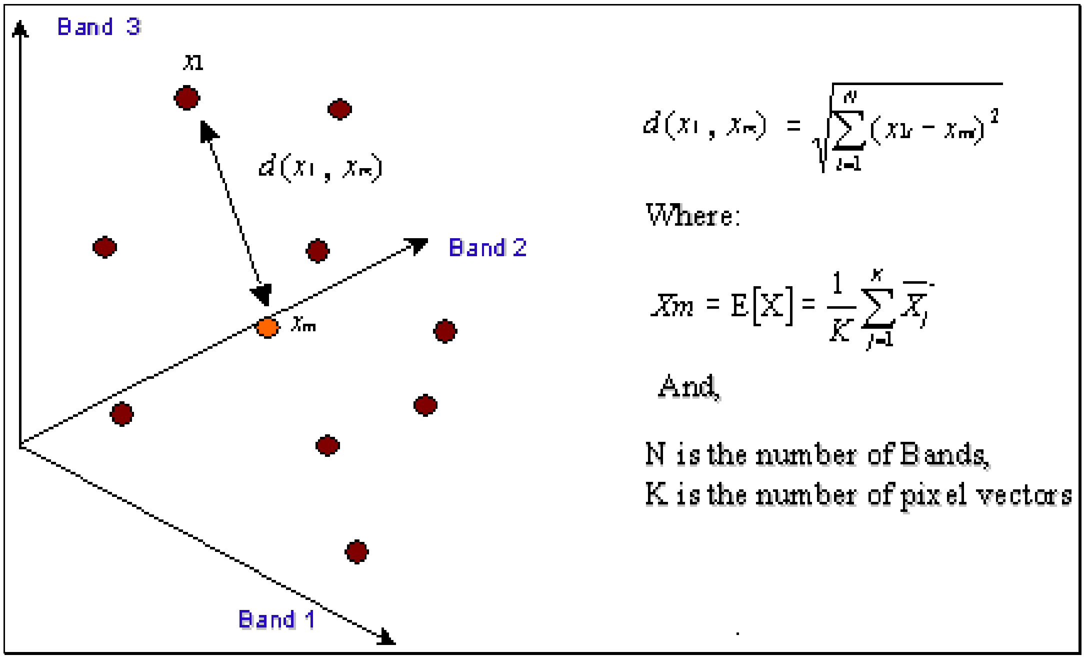

Band 3
$x1$
$d(x_1, x_m)$
Band 2
x_m
Band 1
$d(x_1, x_m) = \sqrt{\sum_{i=1}^{N} (x_{1i} - x_m)^2}$
Where:
$Xm = \mathrm{E}[X] = \frac{1}{K}\sum_{j=1}^{K} \overline{X_j}$
And,
N is the number of Bands,
K is the number of pixel vectors

Figure 2.1 Outlier Detection Method – for 3 band imagery.

The second stage of the sub-pixel detection is called Target Detection. All the outliers selected in the first stage are ranked based on the histogram of the frequency of selection for each of the outliers. For example, if a 3x3 kernel systematically passes over the image cube, an individual pixel can be selected as an outlier a maximum of nine times. Put another way, the centre pixel in a 3x3 kernel can be included in nine different kernels of the same dimension – unless the pixel is at the edge of the image. According to Subramanian and Gat [1998], pixels having the highest frequency of selections as an outlier are then most likely to be actual targets.

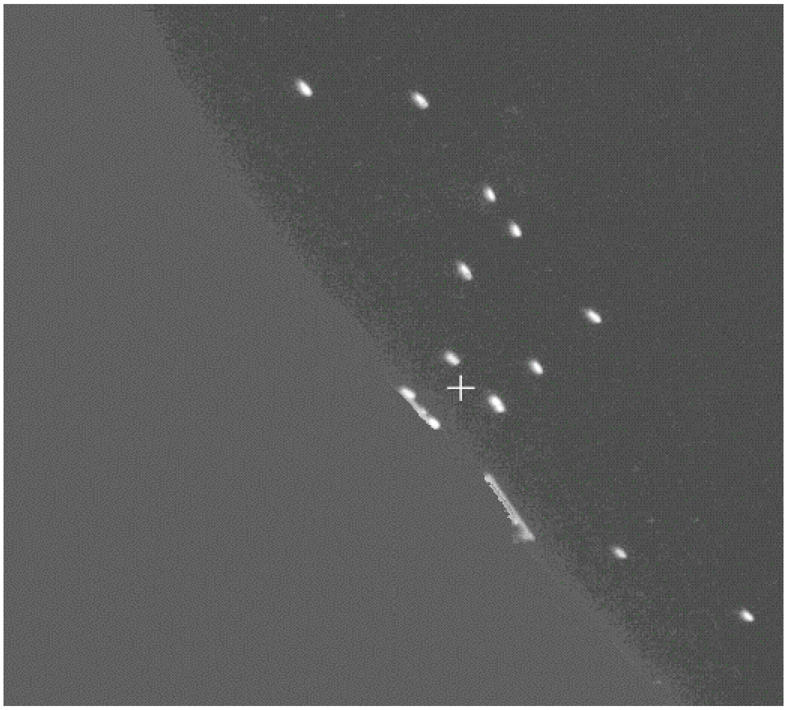


Figure 2.2 IKONOS Imagery, Fredericton Yacht Club, Fredericton, New Brunswick, Canada.

After a vessel has been detected, data must be gathered regarding its characteristics. Shortis et al. [1994] developed a method within their work on target recognition which will be referred to as 'target characterization'. The name is given herein, to clarify the distinction between this method and traditional spectral based classification techniques. This method of generating information for the vessel length and width for example, could easily be adapted to provide a heading for the vessel [Zhang, 2001]. Further, the construction material of the vessel can be determined by using the spectral response of each target and comparing this with known spectral libraries.

The method developed by Shortis et al. [1994] is as follows:

- Target objects are determined by scanning each line in the image from top to bottom.
- An eight-way search is initiated around each candidate pixel that exceeds a predetermined threshold to find other candidate pixels.
- All pixels in the target object are given a unique label.
- The length and width of the target are measured using the $\Delta x,\Delta y$ dimensions (at this point, the azimuth could be determined) of the target object.
- The line-by-line search for new targets continues until all the targets are characterized.

Finally, to complete the characterization and classification of the vessel, the spectral signature under each uniquely labeled target object is generated. This signature is compared with a reference library of known signatures to determine the material of construction of the vessel. If the material of construction of the vessel isn’t found in the spectral library (most vessels are constructed of the following: fiberglass, steel, aluminum, wood, or concrete) the target could be labeled as a false alarm and discarded. Research suggests that the ISODATA classification scheme is best suited to high-resolution imagery (Zhang, 2000).

2.3 Methodology for Developing a Marine Recreation Reconnaissance System

It is believed, based on the background research, that it is possible to create a working recreational vessel reconnaissance system using civilian technology in support of Search and Rescue marine activity assessment. The purpose of the system would be to gather and classify high quality data on marine recreational vessel inventories for input into SARMAP.

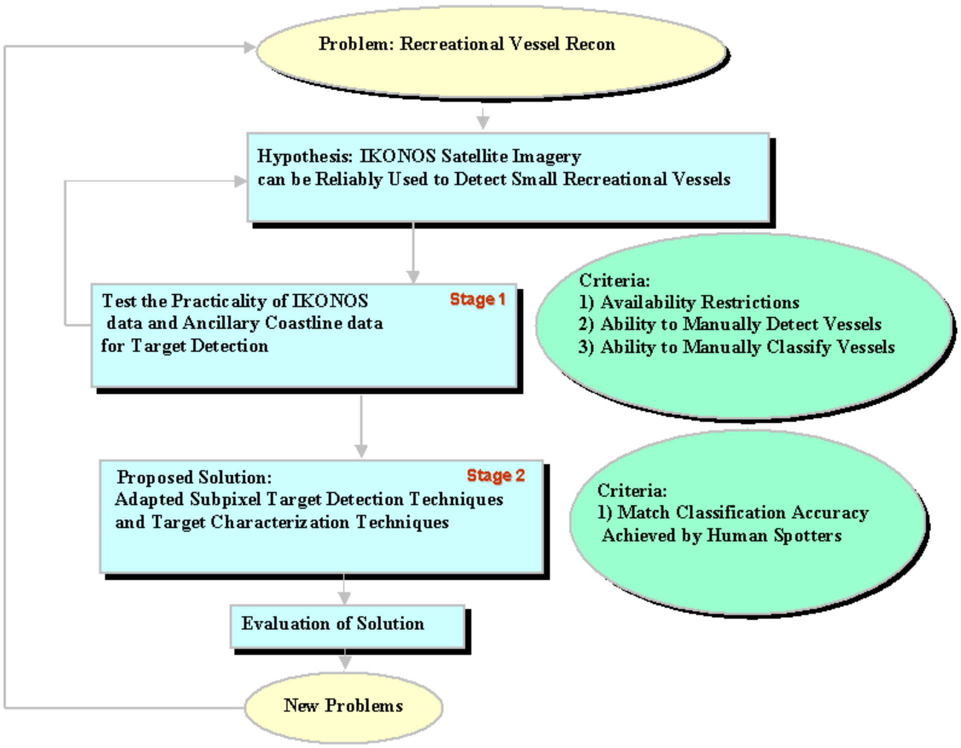

Problem: Recreational Vessel Recon
Hypothesis: IKONOS Satellite Imagery can be Reliably Used to Detect Small Recreational Vessels
Test the Practicality of IKONOS data and Ancillary Coastline data for Target Detection
Stage 1
Criteria:
1) Availability Restrictions
2) Ability to Manually Detect Vessels
3) Ability to Manually Classify Vessels
Proposed Solution: Adapted Subpixel Target Detection Techniques and Target Characterization Techniques
Stage 2
Criteria:
1) Match Classification Accuracy Achieved by Human Spotters
Evaluation of Solution
New Problems

Figure 2.3 Proposed Methodology for the Development of a Recreational Vessel Reconnaissance System.

Figure 2.3 illustrates the proposed methodology initially being adopted for this work. The first stage of the work has entailed a detailed exploration of VHSR imagery: its relative strengths and weaknesses making use of the imagery shown in Figure 2.2.

The algorithm developed by Subramanian and Gat [1998] has been implemented using C/C++ code in conjunction with the PCI C/C++ developer's tool kit. The feasibility of using this work was then tested on IKONOS imagery containing several yachts resting on their moorings in the Saint John River, New Brunswick. It is important to recognize that this imagery was not collected for testing this work nor was any exhaustive ground truth data gathered. The intention of the first phase of this project is not to provide rigorous scientific results. As was stated above, the first phase is to be an exploration into the potential of using the spatio-spectral template for this purpose. Further, this exploration is being done without spending large sums of money on ground truthed VHSR imagery. Once the researchers are satisfied that a viable system is possible, then our partners at CCG will be approached for approval to move onto the second phase of the research.

2.4 Results

The image on the left side of Figure 2.4 is a resampled and pseudo coloured portion of the imagery found in Figure 2.2 shown above. The original imagery was not suitable for even preliminary testing of a sub-pixel target detection algorithm. The "target" vessels found in the image are on the order of 8m or more, in length. As a result, the image was resampled to a lower resolution in an attempt to mimic having VHSR imagery containing small recreational vessels. Given the limited portion of the imagery found in Figure 2.2 that contains recreational vessels, the resampling to a coarser resolution was limited before the image (in terms of total number of pixels) became so small as to be useless.

The panchromatic band was resampled to 6m resolution using bilinear interpolation. Correspondingly, the multispectral bands were resampled to a 24m resolution also using a bilinear interpolation. However, the multispectral image was then resampled back to 6m resolution using a nearest neighbour interpolation to preserve the data values. Finally, having both the panchromatic and multispectral band with the same spatial resolution, they were merged into one image cube. Yachts that had a dimension of eight or more pixels long now are only one or two pixels long. It is still not a sub-pixel target but much improved for evaluation purposes without incurring great cost.

The image on the right side of Figure 2.4 shows the results from the implemented spatio-spectral target detection algorithm. Figure 2.5 is a merger of both the images found in Figure 2.4 to better illustrate the preliminary results.

Figure 2.6 illustrates the output from the target detection algorithm for a specific vessel that corresponds to the "+" cursor location in Figure 2.5. It is important to note that only one of the pixels in the vessel was selected as a target. The rest of the pixels are given a value of zero.

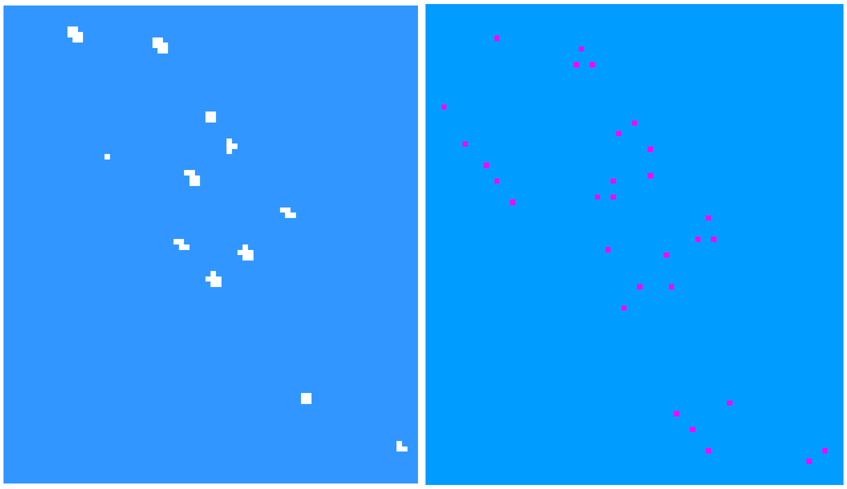


Figure 2.4 Left: Resampled, pseudo-coloured panchromatic band. Right: Resulting "target" channel.

Each of the 11 vessels was "targeted" by the algorithm. Nine targets were detected incorrectly right along the shoreline and there were 11 unexplained false targets. The false targets along the shoreline can be attributed to a poor job of masking out the upland areas by hand. The remaining false targets are unexplained, so some further development work must be done to enhance the spatio-spectral template and reduce the number of false targets. We are however encouraged that even with no enhancement of the original algorithm which was designed for a hyperspectral environment, all the vessels were targeted.

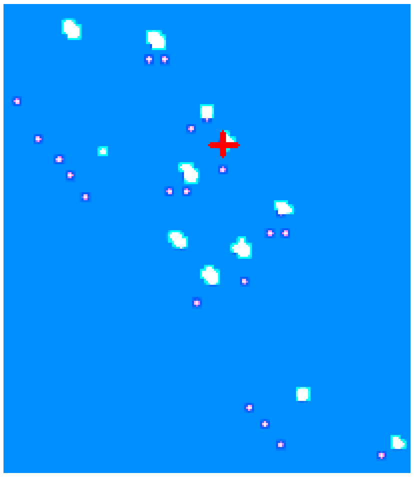
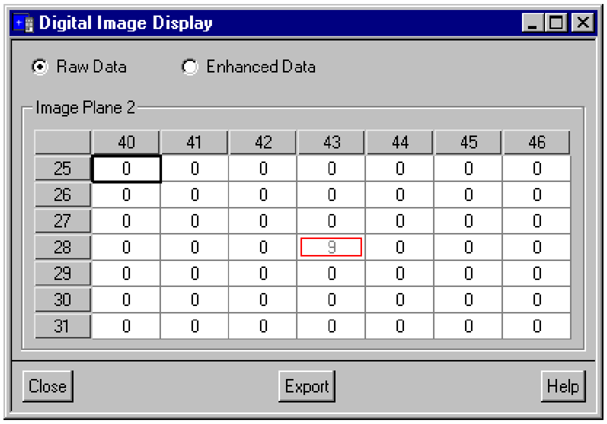


Figure 2.5 Resampled, pseudo coloured panchromatic band merged with the "target"channel.



● Raw Data ○ Enhanced Data

Image Plane 2

	40	41	42	43	44	45	46
25	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0
28	0	0	0	9	0	0	0
29	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0

Figure 2.6 Digital numbers surrounding a single targeted vessel.

2.5 Further Work

The second stage in the development of a recreational vessel reconnaissance system would be to adapt and improve the spatio-spectral template developed by Subramanian and Gat (1998). Specifically, the number of false targets must be reduced.

Future investigations include: a “region-growing” capability must be introduced to generate all the pixels corresponding to a vessel using each of the “target” pixels as shown in Figure 2.5, use Mahalanobis distance metric for outlier selection, and incorporating the response in the NIR band as a test for false targets – one would assume a low response for a false target in the NIR band if the target was just water.

2.6 Validation of the Reconnaissance System

In the previous research on recreational/tourism marine activity assessment [Pelot, 2000], a single calibration exercise was performed testing the ability of spotters to identify and classify recreational vessels. The Canadian Coast Guard Auxiliary (CCGA) provided various vessels of known type and length for the spotters in aircraft to detect and classify. Eighty-two percent of the vessels were correctly classified. The methodology developed within *MRV Recon* may be deemed acceptable if it meets or exceeds the eighty-two percent classification accuracy achieved by the spotters.

2.7 Conclusion

The research described in this paper is important as it will attempt to adapt existing investigations in the use of spatio-spectral template for use with hyperspectral imagery to provide critical reconnaissance data for marine search and rescue. Recreational boaters, particularly sport fisherman/hunters, PWC operators, and those involved in adventure tourism, represent one of the highest risk groups on Canadian waters. The proposed *MRV Recon* system coupled with the SARMAP software are part of a proactive program to ensure that Canadian Coast Guard bases have the best possible spatial distribution. In addition, the technology will ensure the most appropriate SAR resources are used for responding to marine incidents.

The purpose of the proposed *MRV Recon* software is to provide the SARMAP software with the ability to gather data regarding the *spatial distribution and characterization* of recreational vessel activities. Preliminary investigations confirm that commercially available high resolution VHSR satellite imagery may be used to reliably detect small recreational vessels. Imagery from the VHSR satellite would provide the primary data to be used in MRV Recon.

The first phase of the research has clearly demonstrated that an operational marine recreational vessel reconnaissance system is feasible. It also clearly illustrated the necessity of having VHSR imagery of small recreational vessels such as kayaks and PWCs with associated ground truth.

Much work remains in this important research to determine whether of not the proposed approach will meet the 82% classification accuracy achieved by other means. That being said, the technology exists to construct an excellent tool to help in the prediction of future marine incidents and thus allow the Canadian Coast Guard to make better strategic use of their resources and ultimately save lives.

2.8 Acknowledgements

The authors would like to thank the support of the Canadian Coast Guard. This research is being done as part of a federally supported Network Centre of Excellence called GEOIDE (Geomatics for Informed Decisions). Specifically the work is part of GEOIDE project #ENV 60 – Marine Geomatics and Risk Analysis.

2.9 References

Gilliam B., S.W. McCandless Jr., L. Reeves, and B. Huxtable, 1999, "RADARSAT-2 for Search and Rescue". *SPIE Conference on Automatic Target Recognition IX*, Orlando, Florida, April SPIE Volume. 3718 pp.189-194.

Hongyan S. and M. Shiyi , 1995, "Multisensor Data Fusion for Target Identification". *Chinese Journal of Electronics.* Vol. 4, No.3, July 1995. pp 78-84.

Kruzins E., Y.Dong, and B.C. Forster, 1998, "Detection of Vessels Using SAR for Support to Search and Rescue". *Proceeding of the 9th Australian Remote Sensing and Photogrammetry Conference*. Vol. 1, p.7304. July 20-24.

Palliser, J. 2000 . Personal Communication. Superintendent Victoria Rescue Coordination Centre, Victoria, British Columbia. June.

Pelot R., 2000, "Recreational/Tourism Marine Activity Assessment in the Bay of Fundy, Nova Scotia ". *Unpublished Report of the Department of Industrial Engineering, Dalhousie University*, Halifax, Nova Scotia. March.

Shortis, M.R., T.A. Clarke, and T. Short, 1994, "A Comparison of Some Techniques for the Sub-pixel Location of Discrete Target Images". *SPIE Conference on Videometrics III*, Boston, SPIE Vol. 2350. pp 239-250.

Subramanian S., and N. Gat, 1998, "Sub-pixel Object Detection Using Hyperspectral Imaging for Search and Rescue Operations". *Proceeding of the SPIE Conference on Automatic Target Recognition VIII*, Orlando, Florida. April. Vol. 3371. pp. 216-225.

Wallace R., D. Affens, and S. McCandless, 1998, "Search and Rescue from Space". *SPIE Conference on Automatic Target Recognition VIII*, Orlando Florida, April. Vol.3371 pp174-184.

Zhang, Y., 2000, “A Method for Continuous Extraction of Multispectrally Classified Urban Rivers”. *Photogrammetric Engineering and Remote Sensing*. Vol. 66. No. 8. August. pp.991-999.

Zhang, Y., 2001, Personal Communication. Associate Professor, Department of Geodesy and Geomatics Engineering, University of New Brunswick, Fredericton, New Brunswick. January.

CHAPTER 3
EXPERIMENTAL DESIGN

The previous chapters described the implementation and testing of the spatio-spectral template. This chapter outlines the design of an experiment that creates an imagery dataset used to test the detection accuracy of MRV Recon. Once the creation of the dataset has been described herein, the following chapter describes the preprocessing necessary to allow the dataset to be used in MRV Recon.

In addition to the logical position of this chapter within the dissertation described above, it also represents its chronological position. It was recognized that once the initial research and testing was completed, eventually a dataset that reflected the problem of detecting small recreational boats had to be created. Although this dataset was not used until the testing performed in the final journal publication (Chapter 6), work began on it at about this stage of the research.

The dataset described here contains a variety of small recreational vessels. The initial design, its limitations, and the resulting final design are outlined. Particular attention is paid to the statistical determination of the number of targets to be used in the experiment. The chapter concludes with recommendations to others who may undertake a similar experiment.

3.1 Introduction

Ground truth, in the context of vessel detection, requires simultaneously collecting the imagery, position, and attributes of the all vessels within the study area. Iverson

[1997] in his work in hyperspectral imagery makes three recommendations for the meaningful collection of imagery for the detection of targets:

> 1. The imagery must “represent realistic operational conditions”.
> 2. “…Requires realistic spectral content and phenomologies for the objects of interest and for the interfering background”.
> 3. “…The imagery must have *accurate ground truth* with regard to the objects of interest and information should be available on other subpixel objects or features that can give rise to false alarms (emphasis added).”

Iverson recognized the importance of using “real” data instead of “synthetic” datasets. Certainly synthetic datasets are easier and cheaper to acquire. However, “real” datasets more closely mimic the actual operational environment and provide a truer test for detecting targets. He also recognizes that any meaningful test must be made with “accurate ground truth” for without knowing what actually exists within an image it is impossible to prove how well the targets are detected. With these recommendations in mind, a program for data collection for recreational vessel target detection was investigated.

3.1.1 Previous Target Detection Experiments

The data collection programs of Iverson [1997] and Subramanian and Gat [1998] were “land-based”. Land based targets can be set out for an extended period of time until the imagery is collected. Ground truth for land-based programs is a matter of driving to the test site and gathering the necessary data.

The design of a data collection exercise for targets on the water is a more difficult proposition. Given the coarseness of the temporal resolution associated with spaceborne remote sensing, targets must either remain on the water or be replaced several times.

Additionally, the weather in the marine environment is often harsh. Working from open boats, exposed to the elements, the task of setting out targets and collecting data is challenging in the best conditions. Poor weather prohibits the work of collecting ground truth and further complicates matters.

Lengthening the time it takes to simultaneously collect the imagery and ground truth increases cost and complexity of a vessel detection project. Not surprisingly, previous efforts in designing a program for vessel detection research that includes ground truth of the test data are few.

There are two categories of datasets used in vessel detection research. In the first category, no ground truth data is collected and the detection of target vessels in the imagery is compared with vessels detected by a human interpreter [Eldhuset, 1996]. This type of experiment would not be appropriate for testing MRV Recon because the size of the targets are at or near the resolution of the imagery and thus difficult to detect visually.

The second category of vessel detection research includes the collection of ground truth. Vachon et al. [1996] studied ship detection using RADARSAT. The “field program” for this work had three targets participate in the experiment. The targets ranged in size from 63m to 83m. Further, a Canadian Department of National Defence Aurora maritime surveillance aircraft was flown over the target area to provide redundant positioning of the above three targets and, “identify additional ship targets on an opportunity basis [Vachon et al., 1996]. In this experiment all three target vessels were detected.

Another investigation was performed using RADARSAT with associated ground truth of the targets [Randall et. al, 1999]. In this work, which was focused on the detection of icebergs, vessel detection was a by-product of the iceberg detection. Using several images, researchers compared the detection of various sizes of icebergs with the vessels found in the images. The vessels consisted of a moderately sized CCG vessel, an offshore supply vessel, a barge, a small fishing boat, a wooden schooner used for harbor tours, and a container ship. Randall reported that all the target vessels were detected by RADARSAT.

Also exploring the potential of RADAR in vessel detection, Kruzins et al. [1998] performed experiments that included the collection of ground truth data regarding the size, type, position, and orientation of 65 vessels ranging in length from 5 m to 100 m. Kruzins reported that all 65 vessels “used as ground truth have all been captured by the NASA/JPL AirSAR systems at P, L, and C-Bands”. Perhaps most relevant to this project, the experiment specifically dealt with recreational vessels as point targets (no wakes) suggesting that using steep incidence angles for airborne C-Band RADAR can “detect small pleasure vessels with length ranging from 5-20 m …[Kruzins et. al, 1998]”.

After reviewing the previous target detection experiments, it was decided to create an experiment similar to that developed by Kruzins albeit for optical imagery rather than RADAR. Ideally there would be more vessels and in particular vessels between 1 –5 m in length to test the combination of IKONOS and MRV Recon to detect very small targets.

3.2 Location

Pelot [2000] investigated the detection accuracy of trained spotters in aircraft searching for marine vessels in the Bay of Fundy. Initially, the location of the study site for this experiment was also to be in the Bay of Fundy to allow for integration with Pelot's study. However, after a preliminary needs analysis, it became apparent that the Bay of Fundy wasn't an ideal location to hold a recreational vessel detection study using satellite imagery. Poor weather, a short recreational boating season, and a relatively small number of recreational vessels diminished the utility of this site for the study.

In testimony to the strength of research networks, the location of our GEOIDE partners in Victoria, British Columbia provided us with an excellent alternative study site location. During an early meeting between the GEOIDE partners, casual observation revealed a year round recreational boating season and an extremely large number of recreational vessels. Additionally, our GEOIDE research partner from the University of Victoria, Dr. Peter Keller, is an accomplished local yachtsman.

The year-long season, large boating community, and local inroads into that boating community were overwhelming reasons to select Victoria over the Bay of Fundy as the study site for the testing of MRV Recon even at the expense of losing continuity with the previous research.

Cadboro Bay, adjacent to the suburb of Victoria known as Oak Bay, is a small bay on the southern tip of Vancouver Island. Figure 3.1 illustrates the bay that is approximately 1 km wide and 2 km long.

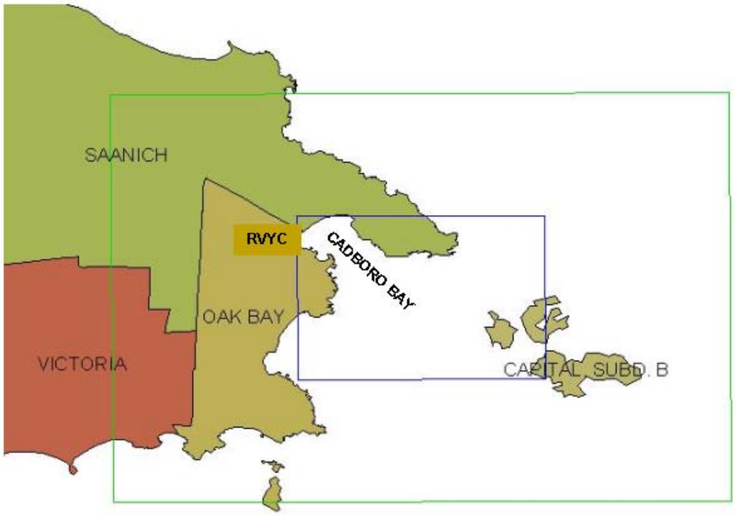


Figure 3.1 Cadboro Bay and RVYC, Vancouver Island, British Columbia, Canada.

Since 1913 it has been the home waters of the Royal Victoria Yacht Club (RVYC). The RVYC has over 1000 members with an active racing, cruising, and junior program. Dr. Keller is an active senior member of the RVYC. Outside of the working relationship with Dr. Keller and his team, potentially of most benefit to the data collection experiment is all the infrastructure that comes with a large yacht club: launching facilities, work boats, mooring tackle, and a large number and variety of small recreational boats.

3.3 Imagery Acquisition

Having now selected Cadboro Bay as the study site or area of interest (AOI), the next step involved selecting an imagery vendor. With the cost of airborne imagery out of reach of the project budget, the remaining options rest with spaceborne high resolution commercial satellite imagery vendors (or their resellers): Space Imaging or Digital Globe.

Initial discussions with various sales staff regarding the need for knowing when an AOI could be imaged met with little enthusiasm. Vendors suggested that, for security reasons, the collect dates on which a specific AOI could be imaged are not normally disclosed. The best option was to pay a “priority upgrade surcharge” to ensure that an AOI would be imaged within an approximately two-week window. This is called priority tasking. In addition to requesting a priority tasking, a client can specify near nadir (within 15°) viewing geometry. By constraining the collection of the imagery in this manner the number of feasible collect dates in the two-week window is reduced. Since we did not know the specific collect dates, the drawback of priority tasking for this experiment is that it would require the setting out of the target vessels and collecting the position attributes every day for two weeks. This would greatly increase the effort and costs surrounding the collection of ground truth data for this project.

In an attempt to get around this problem, Rancourt [2002] in his Masters research attempted to answer the question: if we did purchase a priority upgrade, could the position of the satellite be calculated within that window of time and then know at what days and times the AOI could be imaged? Using the Two Line Element (TLE) sets published by NORAD for all orbiting satellites, Rancourt developed software to calculate when a specified AOI could be imaged.

Nevertheless, discussions continued with the vendors and one individual, Ms. Chantelle Delorme Lafontaine of RADARSat International, recognized the potential of the project and championed our desire for having the vendor provide dates and times the imagery could be collected. Having the vendor provide the dates and time negated any uncertainty using Rancourt's software. After some negotiations, and recognition by the vendors of the necessity of ground truth in this experiment, a deal was struck whereby SpaceImaging, through their reseller RADARSat International, would provide the necessary scheduling data if we paid an additional surcharge on top of the priority upgrade.

In addition to being provided the collect date, it was agreed that they would lift the usual minimal percent cloud cover restrictions except for the AOI. The inner polygon shown in Figure 3.1 must remain cloud free so as not to obscure the small targets. In other words, if on a specific collect date the weather over the region was cloud covered but the weather over Cadboro Bay was clear, then the vendor would release the imagery. It would be disheartening and expensive to set out all the targets and to know that at the collect time the necessary patch of sky was clear, but because of the minimal percent cloud cover restrictions, the vendor would not release the imagery. The securing of this agreement alleviated this restriction.

3.4 Initial Experimental Design

The primary consideration in designing the experiment was establishing the number of targets to include. This consideration was extremely important for two reasons: 1) the number of targets drives both the cost and complexity of the work associated with the ground truth and 2) the number of targets included must be great enough to statistically prove that MRV Recon can accurately detect recreational vessels.

In order to prove the accuracy of MRV Recon it is necessary to find the upper bound for the following statement: “I am 95% confident that the probability of failing to detect a target is no more than … (the unknown upper bound) ”. The quality of the detection accuracy would be reflected in a very small value for the upper bound [Tingley, 2003]. A binomial distribution was suited for designing the experiment. The binomial distribution is quite often used to model random variables in many kinds of experimental and sample survey situations. The equation for the binomial distribution [Scheaffer and McClave, 1995] is:

$$p(y) = \left(\frac{n}{y} \right) p^{y} (1 - p)^{n-y} \tag{3.1}$$

where:

p(y) = the probability of success of a trial
n = number of trials
y = the number of successes in the n trials

Schaeffer and McClave [1995] suggest that a random variable Y is binomially distributed if the following conditions exist:

1) The experiment consists of a fixed number of n identical trials.
2) Each trial can result in one of only two possible outcomes, “success” or “failure”.
3) The probability of “success”, p, is constant from trial to trial.
4) The trials are independent.
5) y is defined to be the number of successes among the n trials.

Specifically for this experiment, the number of trials n corresponds to the number of targets. The observed error rate p is:

$$p = \frac{N - k}{N} \qquad (3.2)$$

where:

p = observed error rate;
k = the number of targets correctly detected;
N = the total number of targets.

For example, if N = 50 targets set out and MRV Recon correctly detected 47 of them then the observed error rate is:

$$p = \frac{50 - 47}{50} = 0.06 \qquad (3.3)$$

An iterative approach was used to answer the question, "I am 95% confident that the true error rate (of MRV Recon) is no more than *X*%". Using the Binomial distribution function within the Minitab statistical software, various "candidate" true error rates are generated for the following equation:

Pr(3 or more errors; N = 50, a *candidate* true error rate) = 0.95 (3.4)

After a few iterations for estimates of the true error rate, it was found that an underlying true error rate (detection rate) would be no more than 12%, as shown in equation (3.5) below:

Pr(3 or more errors; N = 50, 12% true error rate) = 0.95 (3.5)

What equation (3.5) suggests is that, if it is estimated *a priori* that MRV Recon would fail to detect 3 targets of a total of 50 targets set out in the experiment, then at 95% confidence, the true error rate (failure to detect) would be no more than 12%. If more targets were set out then, for the same observed error rate, the true error rate would be reduced. In the context of designing an experiment, a balance is struck between a reasonable guess of the observed error rate, the number of targets used in the experiment, and accepting a reasonable true error rate.

Table 3.1 contains various iterations and combinations of numbers of targets, observed error rates, and true error rates generated using the binomial function. As stated above, for given “observed” and “true” error rates, the confidence increases with a greater number of targets. Note that the confidence of success *q* is measured as *1-p* (known as a one sided test) where *p* is the lower bound listed in the table entries. Finally, when designing the experiment, you must give it every chance to fail in order to truly test its accuracy. If the experiment is designed with so few targets that there are no false negatives, then you have an observed error rate of 0%. However, in such a scenario there will be no reasonable bound established for the true error rate.

Table 3.1 Cumulative Binomial Distribution.

Number of Missed Targets	N=25 p = 0.15	N = 50 p = 0.12	N=50, p=0.15	N = 100 p=0.15
0	0.01720	0.001675	0.000296	0.000000
1	0.09307	0.013099	0.002905	0.000002
2	**0.25374**	**0.051264**	**0.014189**	0.000015
3	0.47112	0.134534	0.046047	0.000093
4	0.68211	0.267954	0.112105	0.000426
5	0.83848	0.435336	0.219353	0.001553
6	0.93047	0.606522	0.361299	0.004702
7	0.97453	0.753252	0.518752	0.012165
8	0.99203	0.860799	0.668101	0.027476
9	0.99786	0.929238	0.791094	**0.055095**
10	0.99951	0.967502	0.880083	0.099447
11	0.99990	0.986476	0.937188	0.163486
12	0.99998	0.994885	0.969939	0.247302
13	1.00000	0.998236	0.986834	0.347425
14	1.00000	0.999444	0.994713	0.457224
15	1.00000	0.999840	0.998050	0.568315

For the initial design of the Cadboro Bay experiment, it was suggested to set out 100 targets. Estimating a reasonable number of false negatives as being 10 (equivalent to a 10% error being observed) then, at 95% (1 - 0.0550946, from Table 3.1) confidence, this would lead to a true error rate of no more than 15%. A greater number of targets would lower the true error rate, but practicality was already in jeopardy as 100 targets are already quite a large number. However, it was felt that this number was manageable. A budget was developed based on 100 targets and is included as Table 3.2.

The budget reveals that quite a complex experiment was envisioned in the first design. It included participation of CASARA (Civil Air Search and Rescue Association) and the Canadian Coast Guard Auxiliary. Further, it was envisioned that we would have GPS and laser range finders on the support vessels. In addition, the targets and vehicles to transport them to the study site were to be rented. The total cost including imagery, communications, insurance, fuel, etc. was budgeted to be around $37,000.

Table 3.2 Initial Budget.

Proposed Budget		
DIRECT COSTS	$	**Comments**
Travel	5,964	Two trip Victoria & Two Ottawa
Imagery	7,200	IKONOS Priority Tasking
Boat Rentals	8,000	60 Kayaks - no discounts average price x 3 single days
Fuel CCGA	1,176	70 l x $0.70 * 8 boats for 3 days
Fuel CASARA	846	151L *4 tanks @ $0.70 per litre * two air craft
Communications	900	two way radios @ $20/day * 15 radios for 3 days
GPS	600	Rentals to put on moving vessels $10 per day * 20 units * 3 days
Ground Control	2,000	Static GPS survey for GCPs
Honorariums	500	RVYC etc
Other positioning	1,020	Leica Laser Locator - range finder @$85.00/day - four units for 3 days
GPS	1,200	GS50 for use with Leica Laser Locator @ $100.00 day
Transportation	960	4 vehicles for 4 days $60.00/day
Project Insurance	1,000	Liability
Food	1,000	Lunch on water
Other		
subtotal	$32,366	
tax	4,854.90	15%
TOTAL	**$37,221**	

3.5 Design Limitations

The budgeted cost of $37,000 was a limitation. However, complexity rather than cost was the biggest limitation. The initial experiment design was too complex. As a result, the local boating groups shied away from it when approached because of the large time commitment. Moreover, when it was explained that there was no control over which days the satellite would image the AOI - meaning that the volunteers would be committing to possibly several days of work that likely would not fall on a weekend - their interest understandably waned.

3.6 Final Experimental Design

The final experiment design was scaled back in a number of areas, but chiefly in the number of targets. By reducing the complexity of the work it was hoped that it would be more attractive to volunteers - and more cost effective.

The UNB Applied Statistics Centre was again consulted regarding the impact of reducing the number of targets. The response was that, if the number of targets was reduced, then there must be a willingness to accept a correspondingly lower confidence level. For example, (referring to Table 3.1) if we put out 25 targets and fail to detect 3 of them, then it could be concluded that, with 75% confidence (1 - 0.2537), that the underlying error rate for MRV Recon in this experiment was no more than 15% (0.15). In comparison, for the same number of failures and true error rates, but using 50 targets, it could be concluded that with 99% confidence the true error rate was no more than 15%. In other words, the greater the number of targets the more confidence is associated with the determination of the true error rate.

Immediately after scaling back the experiment, we were approached with a proposal from the members of the RVYC to undertake the data collection for us for a flat fee of $4750. The profits from this contract were to go to the club's junior sailing program. It was agreed that for this fee they would collect ground truth data for a minimum of 25 targets. Targets included both recreational boats and other objects, such as inflatable racing marks, moored in Cadboro Bay. Further, GPS positions and detailed descriptions for each target would be collected. Table 3.3, illustrates the attribute data that were collected for the targets by the RVYC volunteers.

Table 3.3 Target attribute data collected by volunteers.

Target ID	**Latitude**	**Long.**	**Heading**	**Length**	**Width**	**Material of Construction**	**Sail or Power**	**Remarks**

The RVYC team created two categories of targets: A and B. Category A targets were less than 6 m in length. They consisted of small sailing dinghies, prams (small, blunt nosed rowing dinghies), kayaks, inflatable race marks, and 1 m diameter, steel permanent mooring balls. With the exception of the mooring balls, all targets in Category A had to be moored using temporary moorings of polypropylene line and concrete blocks. Milk jugs were used as floats and were assigned a unique target code and their position was recorded when they were set out in the bay.

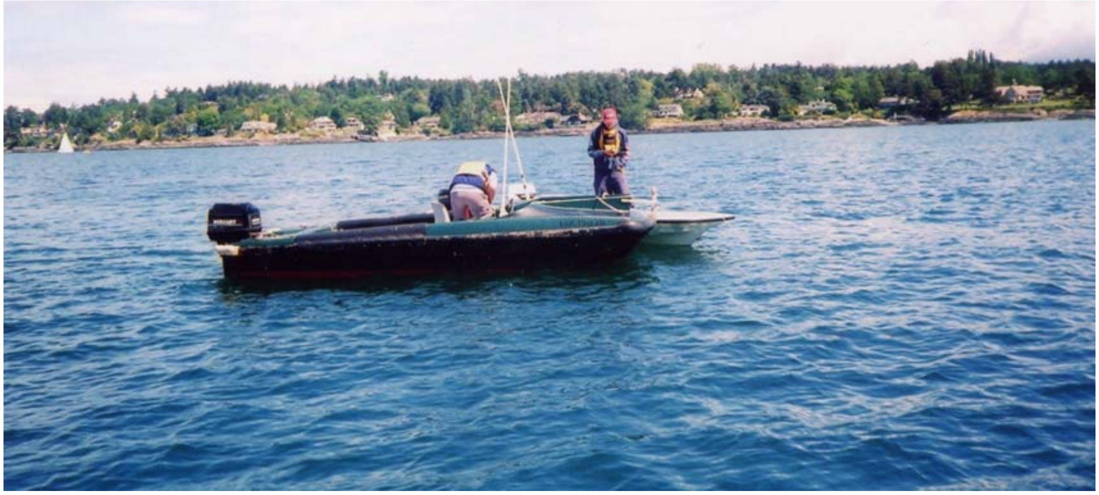


Figure 3.2 Mooring a Category A Target, Cadboro Bay, B.C.

The RVYC agreed to a maximum of four data collection attempts within the priority tasking window between May 15 and May 29. On the morning of each collection date, the targets were towed out in rafts and attached to their assigned mooring. Figure 3.2 shows two support vessels checking the position and correct mooring location for a small Category A target.

The temporary mooring system allowed for time-consuming setting and positioning of the moorings to be done over a few evenings prior to the first collection date. Further, all the attribute data for the small targets - such as length, width, colour, and material of construction - could be conveniently gathered on shore prior to the first collection date. This minimized the time impact of volunteers on the collect days because most of the work was already done. All that had to be done was tow out the targets to their moorings.

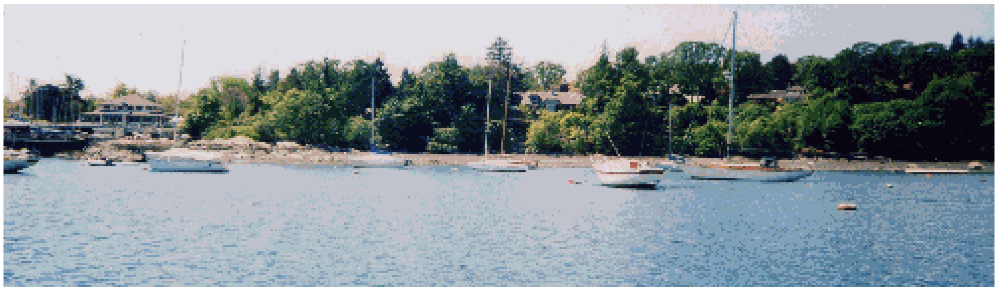


Figure 3.3 Category B targets at RVYC.

Figure 3.3 shows the Category B targets resting on their moorings. Category B targets are larger than 6 m in length. For the most part, these larger yachts were already sitting on permanent moorings in the bay just adjacent to RVYC. Teams of volunteers assigned unique numbers to these targets and collected, sometimes perilously balancing in small support vessels, attribute data for these larger targets. Again, most of this was done prior to the first collect date. On the morning of a collect date, a small team of volunteers would proceed out and check that no changes had occurred such as newly arrived or departed boats.

3.7 Data Collection

In the end, the RVYC volunteers were able to secure 28 Category A targets. The majority of these targets were a mixture of privately owned dinghies, kayaks and inflatable Zodiacs. In addition they borrowed small one-design racing dinghies in the Optimist, Byte, Laser, Flying Junior and 420 classes. The colours of these targets ranged from white to grey. The 25 Category B targets were comprised of larger sailing yachts ranging in size from 6 to 12.5 m in length.

The first collect date was May 15, 2003. In the morning, the Category A targets were ferried out to their assigned moorings and the Category B yachts were checked for any changes. The weather was however uncooperative. The skies remained clouded over for the entire day and the AOI could not be imaged.

The second collect date was May 18. The targets were set out and the weather was only marginally better. However about 30 minutes prior to the collect time of 11:24am, the skies cleared over the AOI while those over the straight remained cloudy.

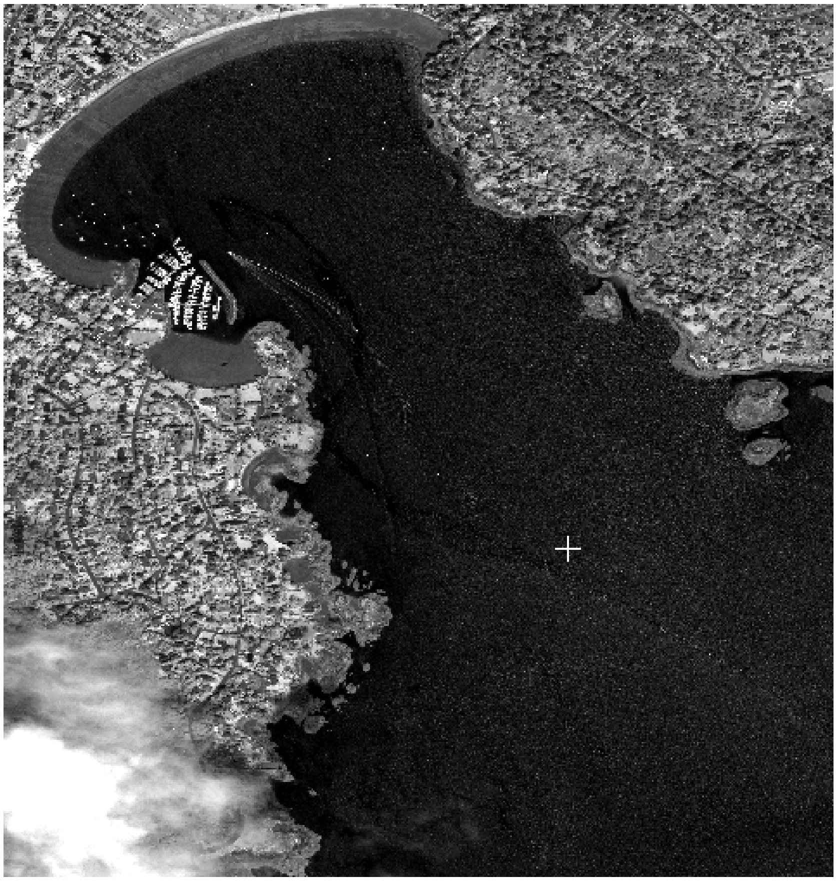


Figure 3.4 IKONOS data with associated ground truth.

Figure 3.4 is the resulting IKONOS imagery for the data collection experiment. The cloud cover can be seen just entering the AOI on the lower left corner of the image. Otherwise the portion of the bay containing the targets is completely clear.

3.8 Target/Image Position Error Budget

Having now collected the imagery, efforts must now be concentrated on the issues surrounding the detection accuracy of MRV Recon. Measuring the detection accuracy of MRV Recon requires comparing what is labeled as a target and what is known to be a target from the ground truth. In assessing the detection accuracy, the maximum difference in the position that is allowable to claim a positive detection must be determined.

There are three components to the total error in position: Global Positioning System (GPS) error, anchor rode error, and the IKONOS position error. Handheld GPS units were used to position the targets. Tiberius [2003] suggests that the expected position estimate using a handheld GPS is between 5 – 20 m of the true position at the 95% confidence level. A 20 m error is a worst-case scenario. However, Dare [2003] suggested that in this case, the handheld units are being used on the water and as there are few obstructions between the satellites and the handheld units, a value of 12 m would be more realistic to adopt in determining the total error in position.

Figure 3.5 illustrates the geometry of a target (the sailing dinghy) resting at anchor. From the records kept during the collection of the ground truth, it is known that most of the pre-made polypropylene anchor lines were 15 m is length. In addition, the shallowest water depth was 5 m. A boat resting on anchor in a gentle wind and/or current has an anchor line with a parabolic curve, as shown in Figure 3.5. However, when the winds and/or currents increase, the anchor line stretches taut. When the line is taut, the boat is the farthest way from the point directly over the anchor. Using Pythagoras’s theorem, then r = 14 m.

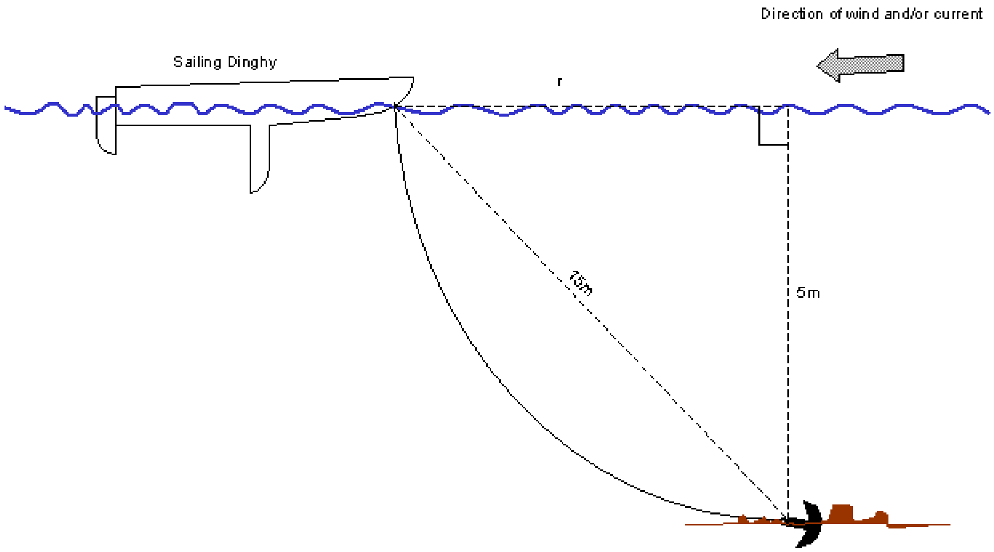


Figure 3.5 Anchor Geometry.

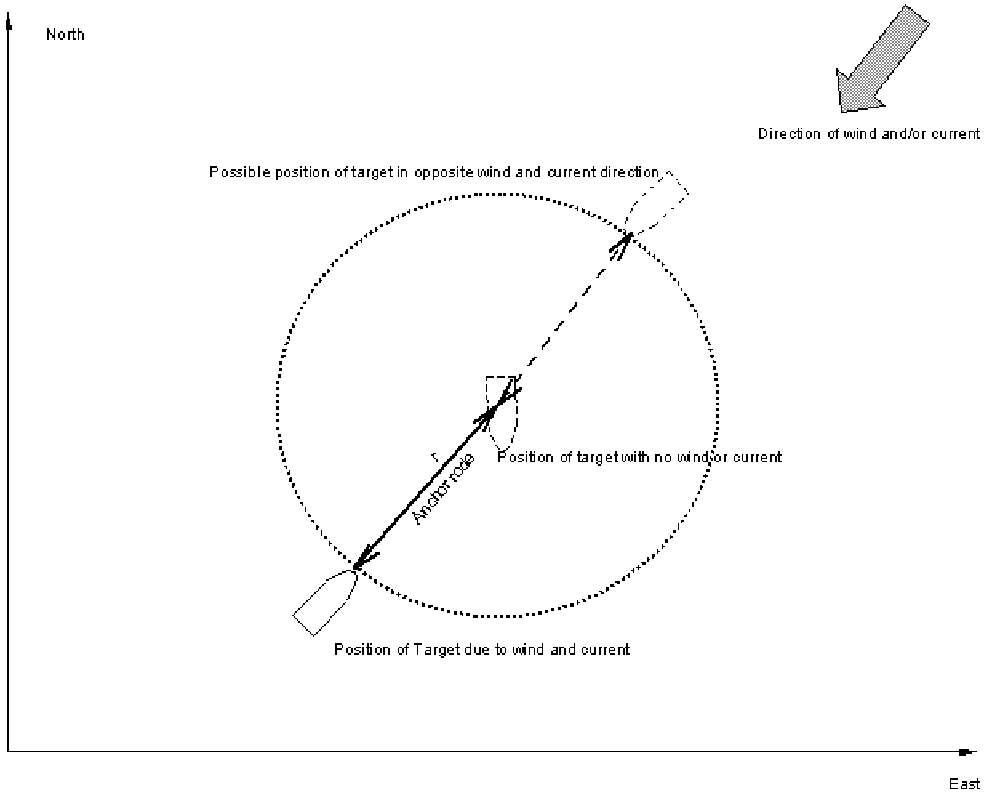

North
Direction of wind and/or current
Possible position of target in opposite wind and current direction
Position of target with no wind or current
r
Anchor rode
Position of Target due to wind and current
East

Figure 3.6 Circular anchor error.

Figure 3.6 illustrates the magnitude of the error due to the target swinging on its mooring. When the target is set out and positioned using the GPS, its position is at the centre of the circle. However, it can swing anywhere within the circle that we have calculated to have a 14 m radius. So the GPS position of the target is +/- 14 m due to anchor error.

Having dealt with the errors associated with the targets, next the error associated with the position of the target in the IKONOS image must be addressed. The published circular error of IKONOS image with the standard positioning package, is 15 m at 90% confidence level [SpaceImaging, 2003]. In order to match the rest of the errors this value was scaled to the 95% confidence level. The circular error at 95% confidence level is 18 m. This translates to the position of the target as measured in the image in +/- 9 m.

With the individual errors investigated, all that is left is to propagate these to determine a value for the total error in position. Equation (3.6) is the propagated total error.

$$\text{Error}_{\text{Total}} = \sqrt{\sigma_{\text{gps}}^2 + \sigma_{\text{anchor}}^2 + \sigma_{\text{IKONOS}}^2} \tag{3.6}$$

Substituting the values determined above yields:

$$\text{Error}_{\text{Total}} = \sqrt{12^2 + 14^2 + 9^2} \quad 21\,\text{m}$$

This total error calculation suggests that if the position of a labeled target as determined by MRV Recon is within 21 m of the recorded ground truth position, then it is recorded as a positive detection.

3.9 Recommendations

When undertaking a data collection mission such as this, two areas need special attention. The first is acquiring the imagery. A great deal of patience is required when finding a vendor (or re-seller) to champion the work. It was critical for us to be provided with the collect dates and times so that the targets could be placed just prior to imagery collection and removed immediately thereafter. Most of the sales people are frankly not that interested in a relatively low budget research project. Moreover, they are fairly resistant to a small imagery purchase that comes with a number of special requests. Thankfully personnel in the remote sensing imagery industry generally well recognize the importance of furthering the science and engineering surrounding remote sensing. After all, in the long term it's of benefit to the vendors of remote sensing imagery.

The second area is cost. Although in this case it was completely unsolicited, acquiring a contract with a stakeholder or interested party is very useful for keeping costs down. In order to facilitate interest by stakeholders it is important to keep the complexity of the data collection as low as possible. For this specific project, we would have ideally liked to have the multi-agency redundant data collection exercise as it was initially designed, however with only a limited budget, it was difficult to get volunteers interested. By redesigning the project, we were able to contract with a stakeholder group and that reduced our cost.

3.10 Summary

Previous research underscored the need for imagery with ground truth. The goal of this data collection exercise was to get as many small targets as possible in order to test the ability of MRV Recon and IKONOS to detect these small targets. The Cadboro Bay, British Columbia, study site was selected for its favourable weather and proximity to large numbers of recreational vessels.

The initial experiment design called for 100 small targets, with redundant positioning using GPS, laser range finders, and CASARA over flights. The experiment was redesigned to be simpler and thus more attractive to volunteer stakeholders. As a result, a contract was made with members of RVYC to set out and collect ground truth data for at least 25 suitably sized small targets. In May 2003, 53 targets (28 were 6 m or less in length) were imaged by IKONOS and position and attribute data were collected for each target.

A study of the error budget surrounding the targets in the imagery was undertaken. The study revealed that if the position of a labeled target as determined by MRV Recon is within 21 m of the recorded ground truth position, then it is recorded as a positive detection.

CHAPTER 4
IMAGE PREPROCESSING

This chapter outlines the development of the necessary image preprocessing for MRV Recon. It introduces the concept of preprocessing for target detection. It describes the development and testing of a preprocessing application. The results from the preprocessing are presented. The chapter concludes with a summary.

4.1 Introduction

Most computer vision applications make use of some preprocessing technique to make the job of processing the image simpler and ultimately reduce the number of false positives [Trivedi et al., 1989], [Shirvaikar and Trivedi, 1990], [Shortis et al., 1994]. Regardless of the environment surrounding a target and the preprocessing methodology used, the concept remains the same: remove as much of the data surrounding and obscuring the target as possible in order to improve the chances of positive detection.

In computer vision for robots or other industrial applications, a common preprocessing technique is background subtraction. Also called "image segmentation" or "gray-level segmentation", the idea is the "conversion between a gray-level image and a bilevel …image" [Parker, 1997 p. 116]. The result is a black and white image that contains all the information regarding the number, position and shape of targets with a lot less information.

Other research suggests that background suppression filtering (BSF) is the identification of "the spectral signature of the troublesome background element or elements … and then carry out a projection operation that reduces the dimensionality of the data set by one …[Ashton, 1999]". This notion of dimensionality of the data is compelling when developing a target detection system. Reduction in dimensionality effectively reduces the amount of data that the detection algorithm must sift through. If no targets exist in the background, as is the case when targeting marine vessels where the land is the background, then the chances of having a false positive are reduced. In non-marine environments the trick is not to have any valid targets hidden amongst the background data and then remove the opportunity of correctly identifying the targets when reducing the dimensionality of the dataset.

The environments for target detection and reconnaissance are decidedly more complex than is often the case for computer vision. In these environments the targets are greatly obscured, sometimes intentionally, by highly variable background [Shortis et al., 1994]. Although requiring a two-step process, target detection in a marine environment (also called vessel detection) may have a simpler background than the terrestrial counterpart.

The first step in vessel detection is to mask out the land portions of the image [Randell et al., 1999] [Eldhuset, 1996]. The result is an image only containing water and the target vessels and thus yielding a higher signal-to-noise ratio to be exploited in the subsequent detection phase (the second step in vessel detection).

Since the overall project goal is to develop a robust and automated reconnaissance system for the CCG, a student was engaged to develop an automated pre-processing method that effectively masks out the upland areas in an image. This work was envisioned and supervised by the principal author as part of the senior technical report course in the Department of Geodesy and Geomatics Engineering, UNB. This work was done by Ms. Katie Munroe, a senior undergraduate student, just prior to collection of the Cadboro Bay dataset. As a result, testing of the preprocessing was done making use of an existing IKONOS dataset over Fredericton imaged in July 2001.

As the student's supervisor, responsibility for any shortcomings of this work rests with this author. In spite of the author's lack of experience in supervision of students, Ms. Munroe's work [Munroe, 2003] was exemplary and her contribution to the development of MRV Recon is both significant and appreciated. The remainder of this chapter will outline the pre-processing steps and results.

4.2 Pre-processing

The IKONOS data cube consists of 5 channels of varying resolution. The panchromatic channel has a resolution of one-metre. The remaining four multispectral channels have a four-metre resolution. Prior to any further processing, the four-metre resolution channels were resampled to one-metre pixels using nearest neighbour sampling to preserve the original data values. The result is a 5 channel data cube: panchromatic, blue, green, red, and near infrared band each having one metre square pixel size. The resulting data cube is shown in Figure 4.1.

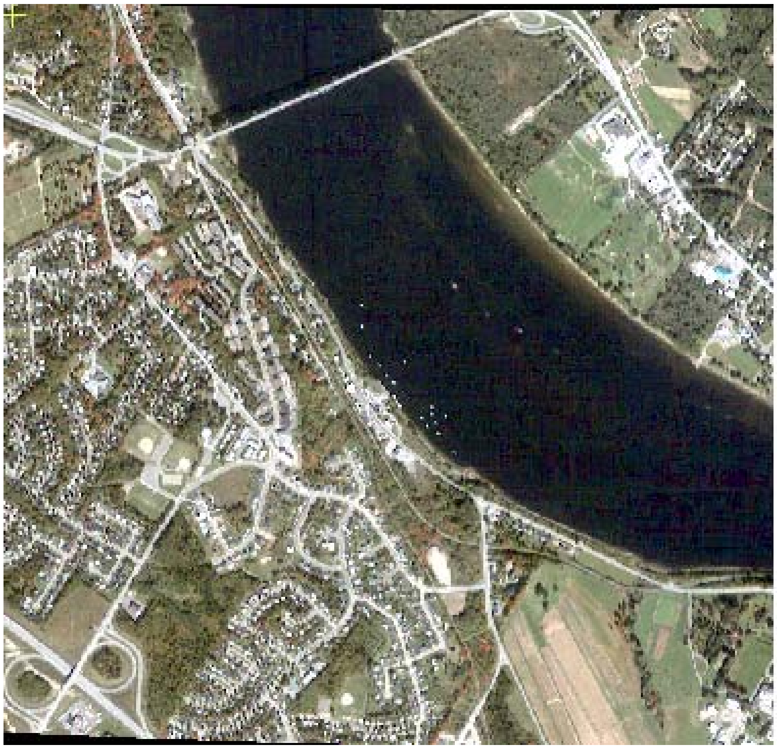


Figure 4.1 IKONOS imagery for testing pre-processing (Munroe, 2003).

4.2.1 Testing Masking Techniques

Unsupervised classification, polygon overlay, and a hybrid method combining unsupervised classification and polygon overlay techniques were tested and compared. Given the objective of developing an automated system, supervised classification techniques were not included in the testing. The following is the evaluation criterion used to select the most suitable method [Munroe 2003]:

1) The distance between the identified shoreline and a reference line.
2) The amount of processing time required.
3) The amount of user interaction required.
4) The potential for automation of the process.

Vector shoreline data from Service New Brunswick is used as the reference line in the evaluation process. McKay stated to Munroe [2003] that the data have a positional accuracy of +/- 2.5 m. This accuracy level is sufficient because (1) the resolution of the imagery being used is only 1 m, so a variation of 2.5 m is not excessive; and (2) boats are not typically found that close to shore unless alongside a jetty. Figure 4.2 illustrates how the distance from the reference line and the shoreline resulting from each preprocessing technique were measured.

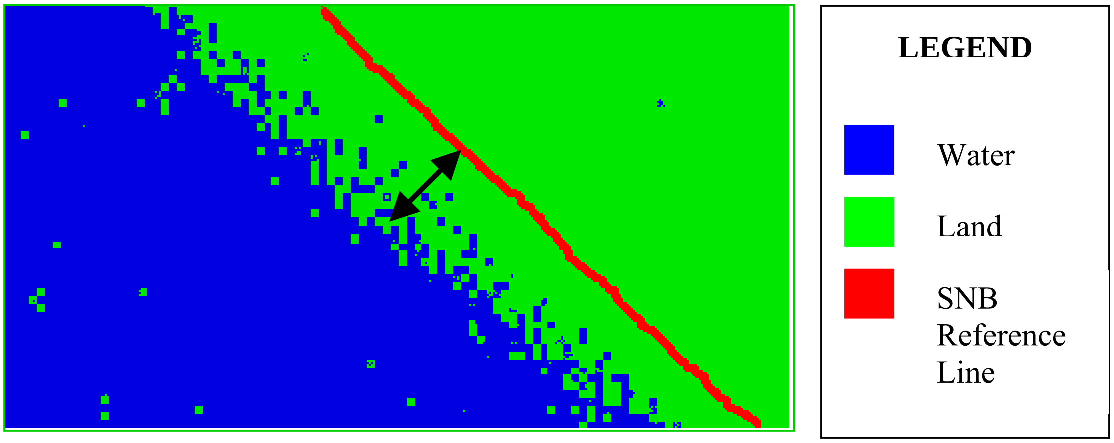


Figure 4.2 The manner in which distances were measured off of the SNB reference line [Munroe, 2003].

The second technique for masking upland portions of the imagery which was tested is unsupervised classification. Unsupervised classifications do not require training data to classify the image. The theory of unsupervised classification is that clustering procedures can be used to group pixels having close spectral characteristics. Following the classification, the user must be able to merge the clusters into classes and label them [Zhang, 2001]. Three different clustering algorithms are tested: ISODATA, K-means, and Fuzzy K-Means.

4.2.2 Evaluation of Results

Table 4.1 summarizes the results of comparing the three preprocessing methods. An extensive discussion can be found in the complete technical report (Munroe, 2003).

Table 4.1 Evaluation of Results [Munroe, 2003].

	UNSUPERVISED	**POLYGON OVERLAY**	**HYBRID**
USER INTERACTION?	**NO**	**YES**	**YES**
PROCESSING TIME	**8 min**	**8 min**	**11 min**
PROCESSING TIME BY USER	**N/A**	**5 min**	**5 min**
AUTOMATED?	**YES**	**NO**	**NO**
SEMI-AUTOMATED?	**N/A**	**YES**	**YES**
DISTANCE FROM REFERENCE LINE (MAX. VALUE)	**20 – 55 m**	**5 – 7 m**	**25 – 50 m**

The above demonstrates, particularly in light of the goal of an automated system, that a pre-processing system based on an unsupervised classification is the most appropriate for this application. The distance from the reference line is the poorest; however, this is somewhat confusing. Although the unsupervised method posted the greatest distance from the reference line it is the only method that masked the land without the SNB vector data used as a reference line. The other two methods made use of the SNB data to generate the mask and one would expect the results in the criteria as such to be better. At the outset of the project, it was hoped that an independent depiction of the shoreline to use as a reference for testing would be acquired. However, time and monetary constraints did not allow for this.

4.2.3 Preprocessing Details

The preprocessing is developed within the PCI Modeler™ application contained in PCI Geomatica™. PCI Modeler allows the user to visually "script" a process. Scripting allows a developer to visually link together different modules within PCI to form a "seamless" process. Once a script has been created, it can be used over and over again, making it a suitable way of automating the masking process [PCI Geomatics, 2002]. Further, once the script has been written, only minimal knowledge of the PCI software is required in order to perform the preprocessing,

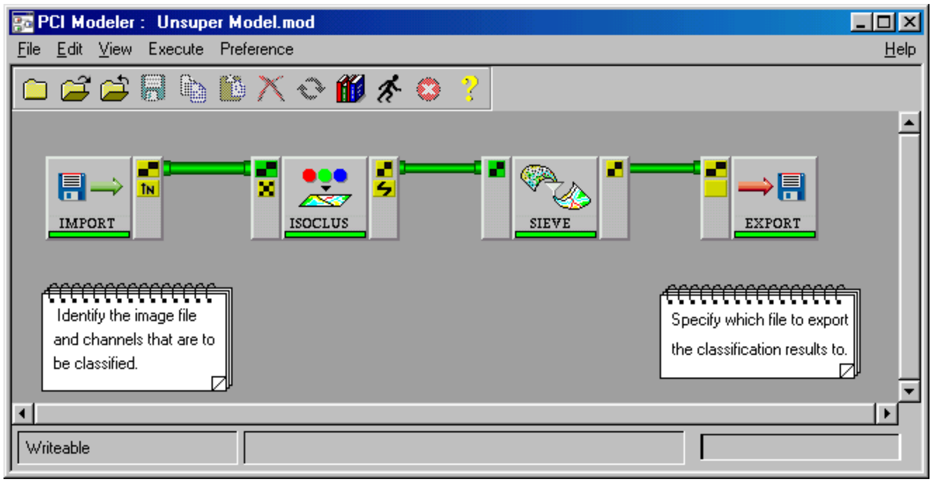
The strength of an unsupervised classification is that the user simply enters the number of classes they desire and the computer classifies the image. Following the classification, all pixels belonging to classes other than the water and boat class would then have to have their gray values set to zero.

4.3 Procedure

In order to mask the image by means of unsupervised classification, the following steps must be carried out [Munroe, 2003]:

1) Run the "Unsuper1" model in the Geomatica Modeler application.
2) Run the XPACE "Model" module.
3) Run the "Unsuper2" model in the Geomatica Modeler application.
4) Run the XPACE "Model" module.

The model shown in Figure 4.3 is used, with only minor alterations, twice in the preprocessing method. With the exception of importing and exporting the data in and out of the Module, the model contains two steps: (1) an unsupervised ISODATA classification; and (2) a SIEVE filter. The ISODATA was found to be the best performing of the various unsupervised classification algorithms tested [Munroe, 2003].



Identify the image file and channels that are to be classified.

Specify which file to export the classification results to.

Figure 4.3 Model that performs the unsupervised classification [Munroe, 2003].

The output of the ISOCLUS module, which is the ISODATA classifier, is shown in Figure 4.4. Munroe (2003) in her experiments discovered that no matter how the ISODATA classifier was manipulated, the target vessels were not classified into a single class. In fact, the boats were classified into various classes found on the land. As a result, if the land-based classes were reclassified to a single background class, which is the goal of the preprocessing, then the boats would be set to background. Since these are the targets of interest, a way of preserving the boats with the water class had to be found.

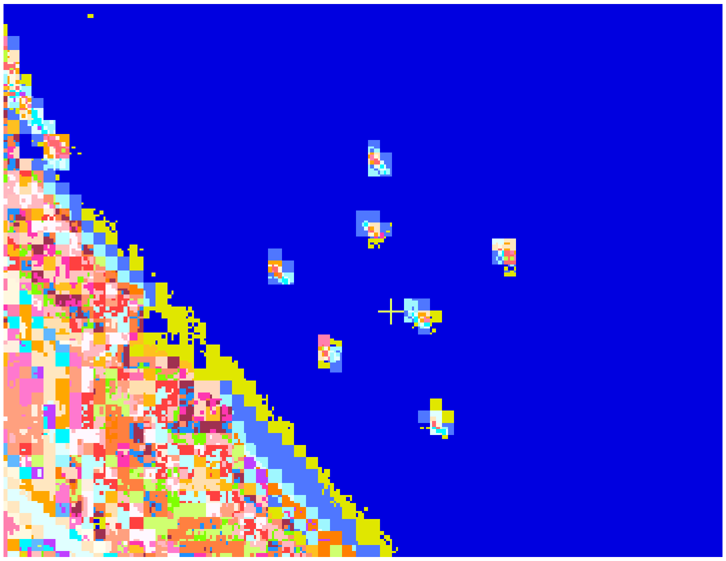


Figure 4.4 Unsupervised classification results using the ISODATA classifier [Munroe, 2003].

This problem is overcome by using a SIEVE filter. This filter is run over the classified image. It merges all small "island" pixels – below a specified size – into the largest neighboring class. Figure 4.5 illustrates that the boats below the minimum cluster size are assigned to the water class. The resulting image has a separate class for water and all the targets in the water. It can be seen in Figure 4.5 that, after the classification layer is run through the SIEVE module the boats are reclassified as belonging to the water class that is the black portion of the figure.

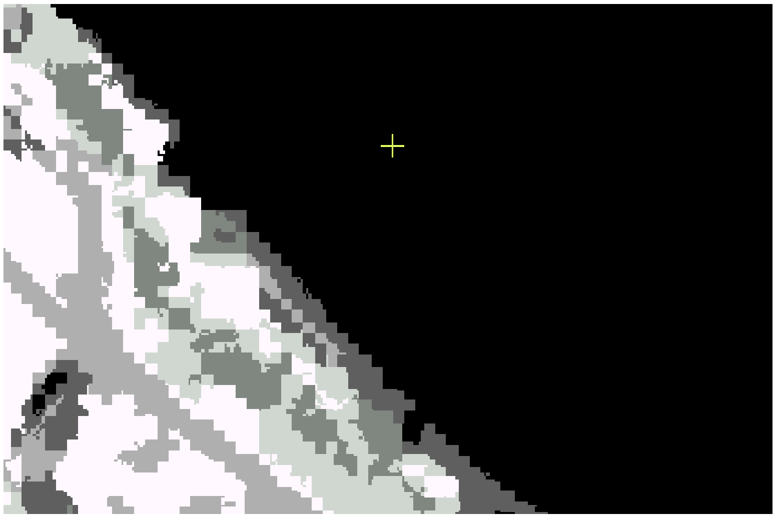


Figure 4.5 The classified image after being processed by the SIEVE module [Munroe, 2003].

4.3.1 Masking Procedure

Armed with an image containing a water/target class, the remaining upland portions of the image can be set to a background value that will not be processed in the target detection software because no targets vessels reside on land. The MODEL is an application that will alter the contents of image channels according to user-defined code. Code was written to perform a raster overlay operations where all pixels belonging to classes other than the water class will have their gray values set to zero. This code is applied to each channel of the image. Unfortunately, the “MODEL” task in XPACE has not been programmed for Modeler yet. Otherwise, this step could be integrated into the “Unsuper1” model [Munroe, 2003].

Figure 4.6 shows that many small land “islands” are classed into the water/target class and are not removed from the imagery by the overlay procedure. Most likely, the islands would be identified as false positives by a target detection algorithm. The solution to remove these small islands requires two steps virtually the same as the first two steps of the preprocessing method. To address this challenge further processing is required.

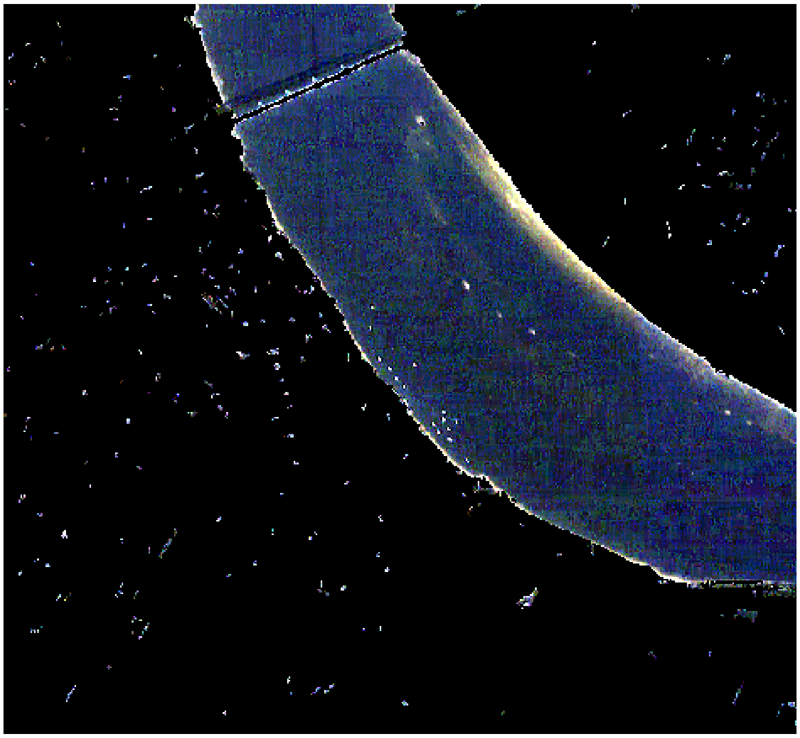


Figure 4.6 The image resulting after running the "Unsuper1" model and "MODEL". Note the land "islands" that remain [Munroe 2003].

Munroe [2003] describes:

> " the difference between the two stages of processing is that different parameters are used in the "SIEVE" module and a different code file is used in XPACE's "MODEL" module. During this second stage of processing, the land "islands" are grouped together with the pixels having gray values of zero by the "SIEVE" module. The code used by "MODEL" then stipulates that all pixels belonging to this null class will have their gray values set to zero."

4.4 Preprocessing Results

Figure 4.7 shows the result of the preprocessing. All the land based parcels have been set to a background value, and all the waters of the Saint John River and the vessels in it remain. The brighter portions within the river are shallow water that appears lighter due to linear enhancement of the image [Munroe, 2003].

Figure 4.8 shows the results of the preprocessing an IKONOS image of Cadboro Bay dataset using the methodology developed by Munroe [2003].

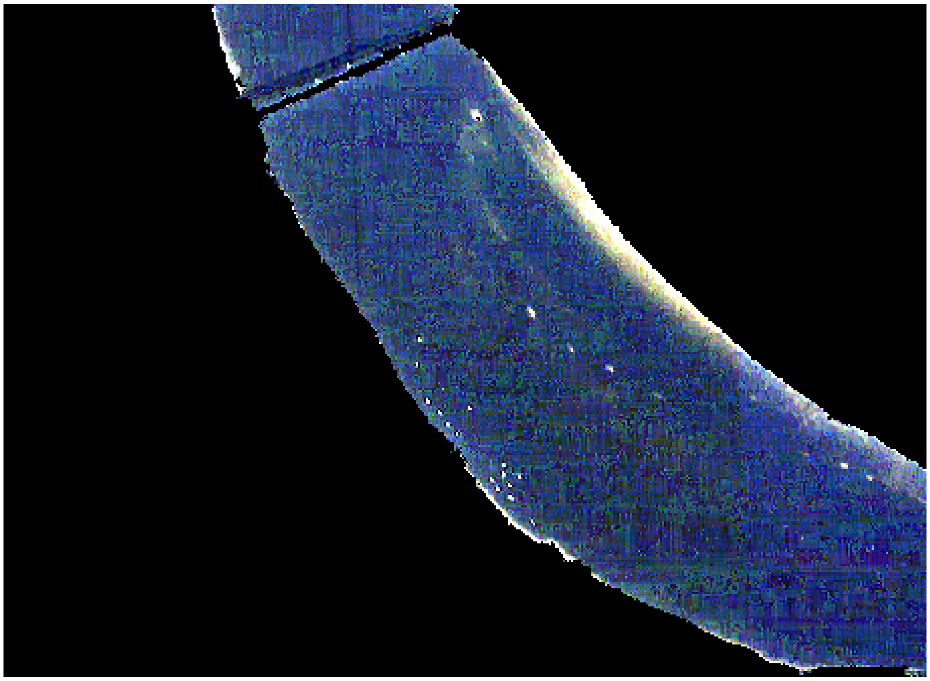


Figure 4.7 Results of preprocessing, IKONOS imagery, Saint John River [Munroe, 2003].

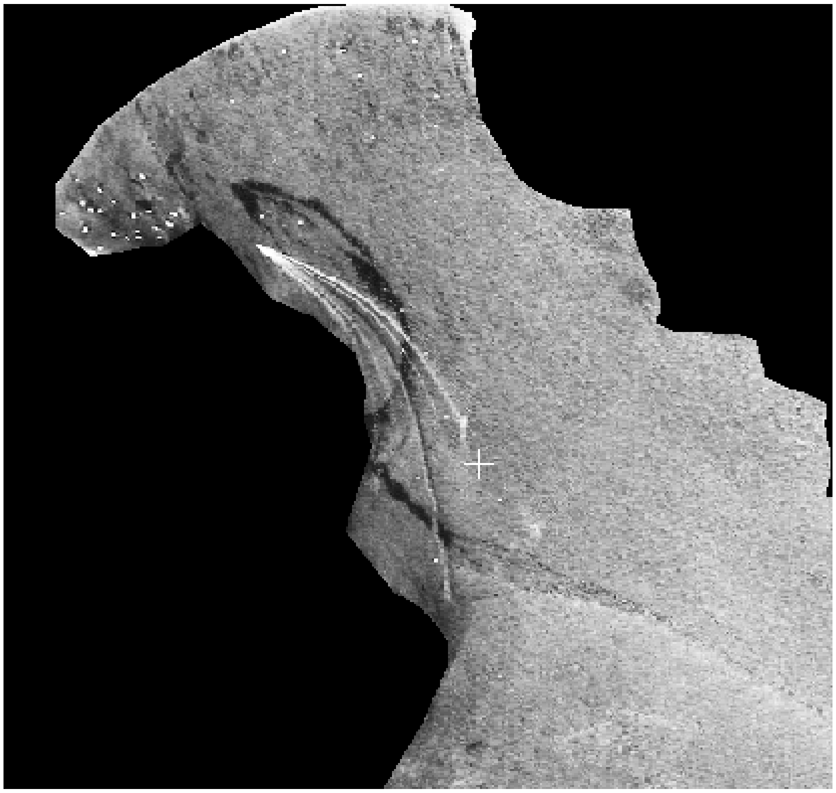


Figure 4.8 Results Pre-processing on IKONOS imagery, Cadboro Bay.

4.5 Summary

The goal of this chapter was to develop a fully automated preprocessing methodology to reliably mask out the upland portions of imagery. This goal was nearly realized with the sole exception that the “Model” application in PCI’s XPACE cannot, to date, be accessed through PCI’s visual scripting environment. It is believed that future versions of Geomatica will have this ability [Munroe, 2003]. The pre-processing methodology developed by Munroe [2003] was used to pre-process the Cadboro Bay dataset for use in testing the detection accuracy of MRV Recon.

CHAPTER 5

COMPARISON OF MAXIMUM DISTANCE METRICS FOR USE IN THE REMOTE SENSING OF SMALL TARGETS[2]

Previously, two chapters outlined the collection of the imagery for the final testing and the necessary preprocessing of the imagery prior to running MRV Recon for the final test.

This chapter returns to specific research surrounding the development and improvement of MRV Recon. It presents the second journal paper for this research, which has been accepted for publication in the Journal of Surveying Engineering [Pegler et al., 2004]. This paper describes the new engineering research for this thesis and builds on the first journal paper and the foundation of research laid down in the first four chapters of this dissertation. Specifically, a new distance metric is outlined, tested, and compared with the distance metrics used in the first journal paper.

5.1 Introduction

There are many applications for small target detection in engineering: topographic mapping, infrastructure inventories and pre-engineering. Near shore marine applications include: mapping breakwaters, piers, navigation structures, pilings, and vessel traffic patterns.

The application driving this research is the development of a reconnaissance system for the Canadian Coast Guard (CCG).

---

[2] Reprinted by permission of ASCE *Journal of Surveying Engineering*, n.d.

The system will be tailored for detecting small recreational marine vessels for example: kayaks, personal watercraft, and small sailing dinghies. In the context of developing a reconnaissance system, it is important to recognize that these are on the order of only a few metres long – if that - and most are less than a metre wide. This is near the best resolution for commercially available space borne optical imagery.

Recreational vessels are of interest to strategic decision makers of CCG because they represent a disturbing 76% of all marine search and rescue (SAR) incidents in Canadian waters. The small kayaks and personal watercraft are of particular interest to the CCG because they represent categories of vessels that are experiencing an increasing trend in the number of SAR incidents (Palliser 2000). CCG desires to reduce the costs associated with responding to these incidents by instituting prevention measures and increasing operational efficiency.

Efficiency can be increased in part by redistributing CCG SAR bases to more effective locations, thus reducing travel times to incidents. Further, a balance can be made between minimal response time and only maintaining the optimal number of bases and SAR platforms to minimize costs. To this end, strategic decision makers have contracted with researchers at Dalhousie University to construct a Maritime Risk Investigation System (MARIS) (Pelot 2000). The purpose of this system is to map, using a GIS, the output of a maritime risk model - the anticipated risk of a future maritime incident. This allows strategic decision makers to analyze the correlation between the location of SAR bases and locations where it is anticipated that future incidents will frequently occur.

Developing a small target detection system for the CCG will allow them to gather data on numbers, types and locations of recreational vessels. Of critical importance is information on the smaller vessels like personal watercraft (PWC) and kayaks as they represent increasing incidence trends. In this stage of testing a marine recreation vessel reconnaissance system (MRV Recon), the Mahalanobis distance metric is implemented to see if it can improve detection of small targets.

5.2 Previous Research

Research into reconnaissance or target detection falls primarily into two categories: automatic and semi-automatic. As we are constructing a system for users whose specialty is marine search and rescue and not geomatics, we will concentrate on research into automatic systems.

Trivedi suggests that an automated target detection system should be robust, accurate, fast, and flexible (Trivedi 1987). Others indicate it would be beneficial if the system could handle the detection without the aid of spectral libraries or templates of the desired targets (Subramanian and Gat 1998) or the requirement for radiometric corrections. Finally, the size, composition or orientation of the target should not hinder the system's ability to detect it.

Research in automatic target detection specific to vessel detection is often focused on Synthetic Aperture RADAR of larger commercial and military vessels (Gilliam et al., 1999; Kruzins et al. 1998). The current resolution of Synthetic Aperture RADAR sensors is just too coarse for the small targets (Pegler et al. 2003).

Investigations into specific research into target detection for search and rescue led to the work of Iverson (1997) and Subramanian and Gat (1998) about the development of a spatio-spectral template for use with hyperspectral imagery. Subramanian and Gat suggested in their research that it may be possible to adapt the template for use with high resolution multispectral imagery. Their “spatio-spectral template” relies on selecting pixels that are distinct from their neighbors. A kernel of user-specified size is passed over the data cube. At each kernel location, a distance is calculated for every pixel vector in the kernel from the mean vector. The pixel farthest away from the mean and/or above a certain threshold is labeled as an outlier. Those pixels possessing the highest outlier frequency are identified as “targets”.

Various distance metrics can be employed. Equation (5.1) describes the Minkowski distance metric (Hartigan 1975). When r = 1, it is known as the Manhattan distance and when r = 2, it is known as the Euclidian distance metric.

$$d = \sqrt[r]{\sum_{i=1}^{p} \left| xi - yi \right|^{r}} \tag{5.1}$$

In his discussions of clustering algorithms, Hartigan (1975) reviewed a “statistical” or weighted Euclidean distance developed by Mahalanobis. Intrigued to see if the lower spectral resolution inherent in multispectral imagery versus hyperspectral imagery could be offset by using the weighting of distances, the senior author decided to implement the Mahalanobis metric and compare it to the Minkowski metrics. The formula for the Mahalanobis metric is shown below as equation (5.2). The weighting is performed by the inverse covariance matrix.

$$d^2 = (x - m_x)^T C_X^{-1} (x - m_x) \tag{5.2}$$

where

C_x = the covariance matrix

m_x = the mean vector

5.3 Methodology

The initial development of MRV Recon began with adapting and implementing the work developed by Iverson and Subramanian and Gat. Only the Minkowski distance metrics were implemented for investigation of the feasibility of adapting this methodology to high resolution multispectral imagery. The early work suggests that the spatio-spectral template can be nicely adapted for use with multispectral imagery (Pegler 2003).

Of particular interest is discovering whether the spatio-spectral template can be improved upon to ultimately detect small recreational craft such as PWC and kayaks. However, the first step is to implement the new enhancement and test its performance against the original spatio-spectral template. The Mahalanobis distance metric was implemented to address the loss of spectral resolution when using multispectral imagery instead of hyperspectral imagery. The variance – covariance information is used to weight the distances to help offset the loss of spectral resolution.

The quality of the anticipated improvement from using the Mahalanobis distance metric can be established by comparing the number of detected targets, the number of false positives or false alarms, and the number of false negatives or the failure to detect a target, with the results from the Manhattan or Euclidean distance metrics.

The MRV Recon software is being constructed by the lead author using C/C++ code in conjunction with the PCI™ Version 7.0 C/C++ developer's tool kit.

5.3.1 Data

IKONOS data imaged in the spring of 2001 was used for testing in the early stages of this research. The image in Figure 5.1 shows vessels from the Fredericton Yacht Club resting on their moorings in the Saint John River, New Brunswick, Canada. The vessels are mostly sailboats approximately 6 – 13 metres in length. Ideally, we would employ imagery with much smaller targets in order to test the software's ability to detect the small targets we are interested in. However, data of that nature were not ready in time for this publication [Sept. 2003].

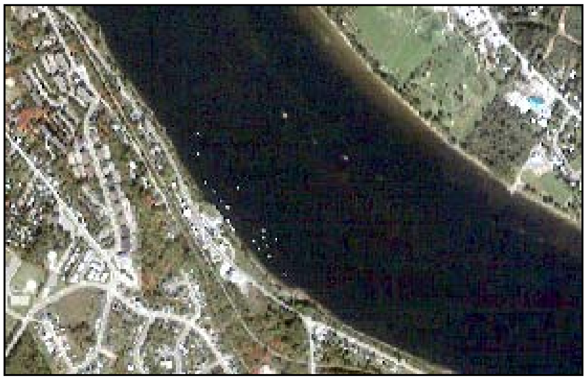
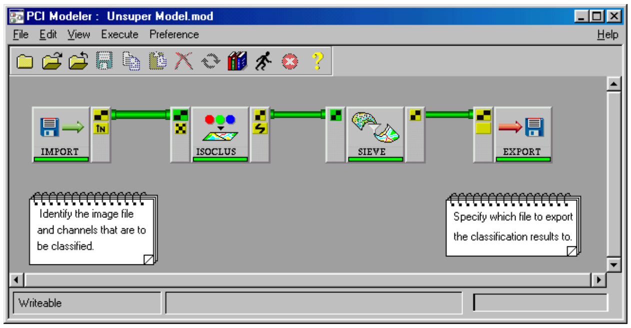


Figure 5.1 IKONOS imagery, Fredericton New Brunswick, Canada.

5.3.2 Preprocessing

The imagery was preprocessed to mask out all the "upland" pixels to a null value. This is done to improve the efficiency of the software and to avoid falsely identifying any "vessel" on land. An undergraduate student in fulfillment of her final year technical report on Geomatics Engineering developed the processing technique (Munroe 2003). This was done under the supervision of the principal author. The entire preprocessing technique was implemented using PCI Geomatica 8.0™ software. The goal was to develop a Modeler™ application (a visual macro tool) that would allow easy repetition of the preprocessing procedure.

After considerable testing, and in keeping with the project's desire to provide an automated solution for the CCG, the unsupervised ISODATA classifier was used. Figure 5.2 illustrates the graphical user interface and the individual operations used to construct the preprocessing masking procedure.



Identify the image file and channels that are to be classified.

Specify which file to export the classification results to.

Figure 5.2 PCI Modeler preprocessing application.

After the image was imported and classified using the ISODATA classifier, the resulting classified image is run through a SIEVE filter. The purpose of this filter was to assign to the surrounding water class any classes found within a body of water having a user-specified minimum size. This was necessary to avoid having the larger yachts grouped into a land class; otherwise the yachts would end up classed as something other than water and then given a background value. The use of the SIEVE ensures the target vessels are not mixed in with the background values as the classification schemes suggest. Figure 5.3 illustrates the results of the preprocessing.

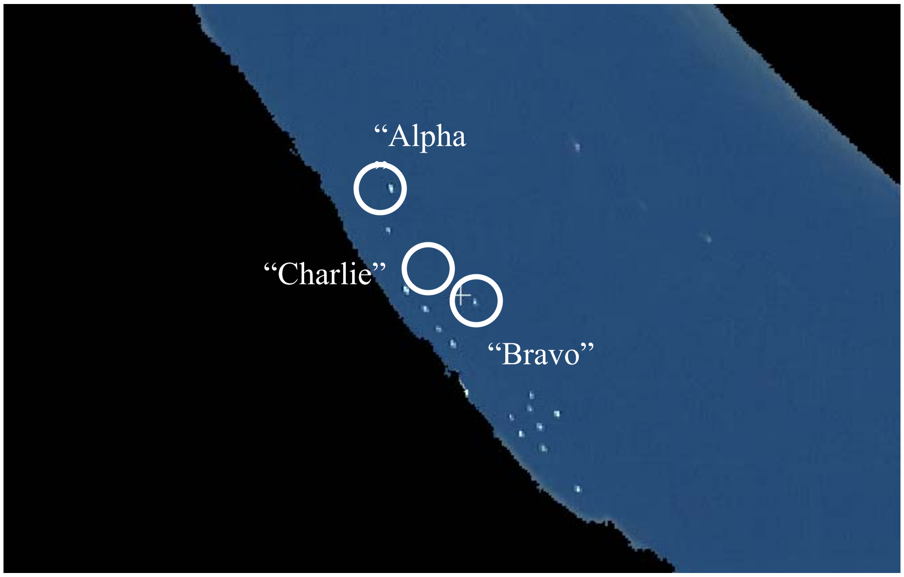


Figure 5.3 Preprocessing results.

5.3.3 Target Selection

To facilitate the investigation of the process and results of the software, three targets: *Alpha, Bravo*, and *Charlie* were singled out for detailed analysis. Shown in Figure 5.3 targets *Alpha* and *Bravo* are both fiberglass sailboats having lengths of 8m and 9m, respectively. Figures 5.4 and 5.5 are spectral response curves of targets *Alpha* and *Bravo* for the five IKONOS channels: panchromatic, blue, green, red, and near-infrared (labeled Channel 1 through 5 in the graph's legend, respectively). The response curve is for the profile lines drawn down the length of the vessels, as shown in the left hand image within each figure.

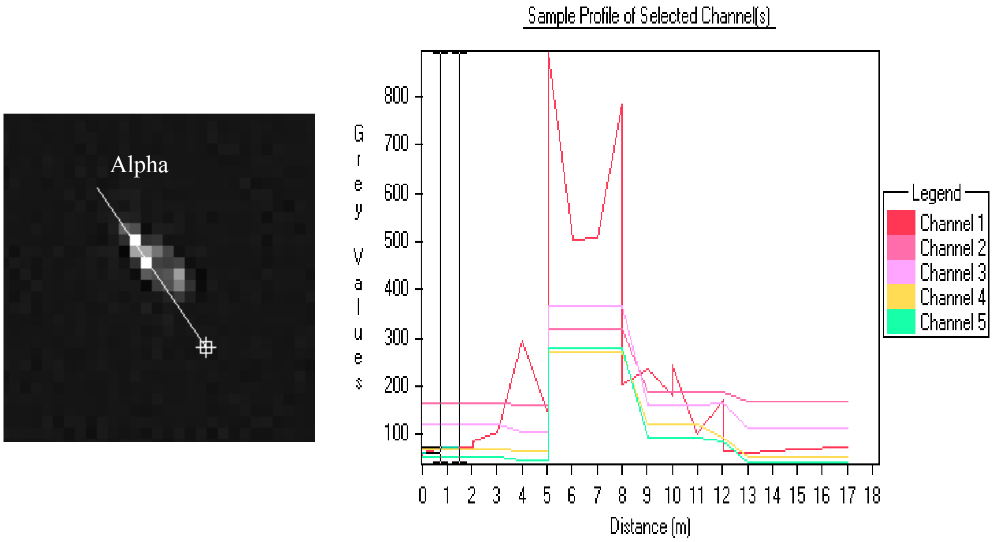


Figure 5.4 Spectral response curves for target *Alpha.*

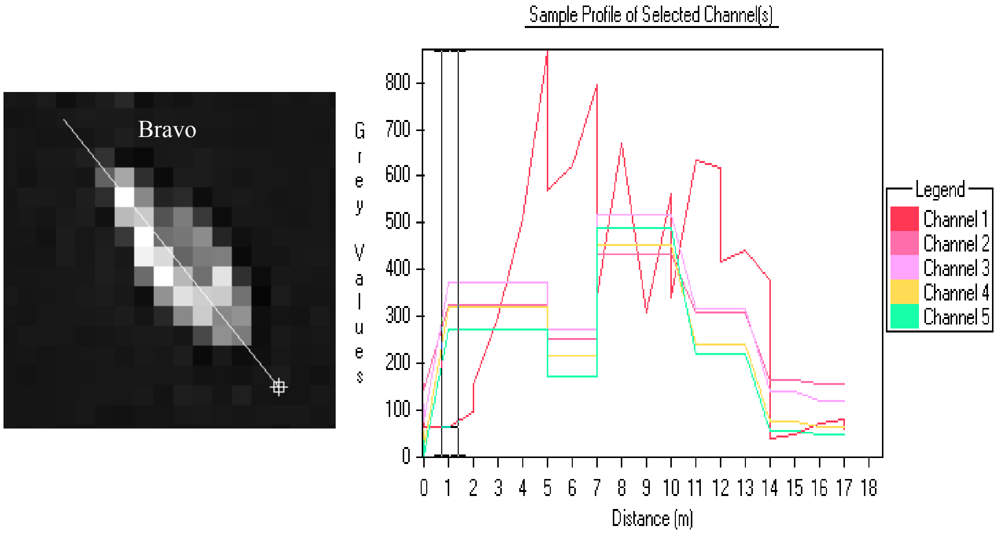


Figure 5.5 Spectral response curves for target *Bravo.*

Both targets show a dramatic jump in digital number (DN) for the panchromatic channel from the surrounding background values. DN values for the blue, green, and red channels exhibit similar behavior with parallel increases in response for the target pixels from those of the background water pixels. There is a loose correlation between increases in DN values for the panchromatic channel and the R,G,B channels with some minor exceptions. The near-infrared (NIR) channel exhibits the largest jump in value from the background to target pixels. This behavior makes sense given that NIR exhibits a very low response for the background water pixels.

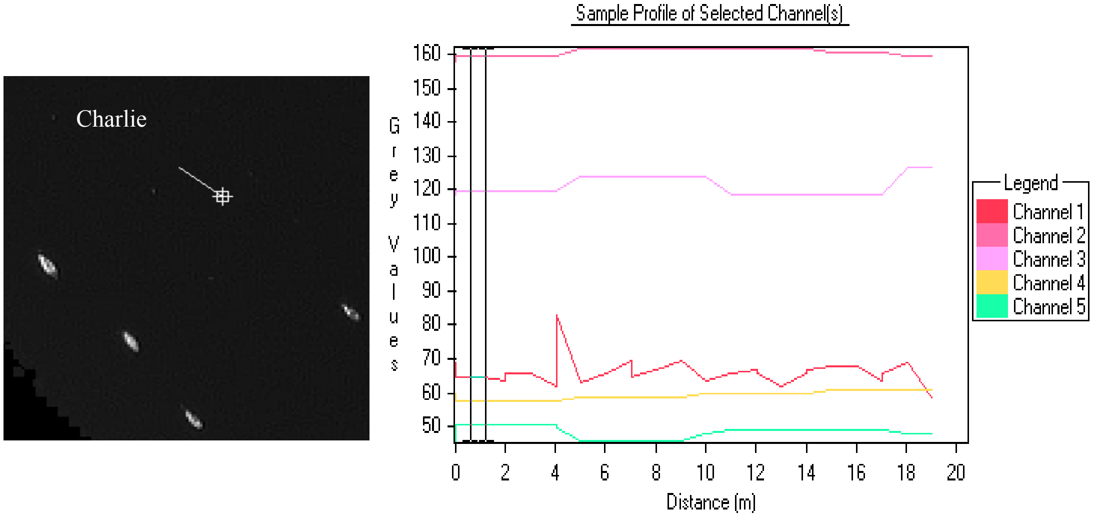


Figure 5.6. Spectral response curve for target *Charlie.*

Figure 5.6 is the response curve for target *Charlie. Charlie* is representative of a sample of the background pixels. As one would expect, the response curve shows the highest values are attributed to the blue and green channels. The response in the red band is quite low, and again the NIR has an extremely low response. The higher resolution panchromatic band demonstrates some fluctuations that could reasonably be attributed to small cresting waves or the glint of the sun off the water's surface. In general however, across the five bands, the spectral responses for the background pixels are much lower than those of the target pixels.

5.4 Results

The test image, Figure 5.1, was processed using each of the Manhattan, Euclidean, and Mahalanobis distance metrics. In turn, each distance metric was tested with a 3x3 and a 5x5 processing kernel. In addition, a particular pixel vector was selected for each of the targets and the various data values for the pixel and its neighbors were recorded and investigated.

5.4.1 3x3 Kernel Results

Table 5.1, shows the specific outlier frequency and distance results for each of the three target pixel vectors. For a 3x3 kernel, a particular pixel has the opportunity of being an outlier in 9 different cases as the kernel passes over the data cube. Target *Charlie*, the background target, was not selected as an outlier by the Manhattan or the Euclidean distance metrics. However, the Mahalanobis metric selected it as being an outlier 2 of 9 times. Scanning down the columns containing the actual distance values, the distances from the mean for target *Charlie* were the smallest – particularly for the Minkowski metrics. The range in distances from the mean for the Mahalanobis metric is much less than the other metrics making it more susceptible to falsely selecting an outlier. It is important to remember that only those cells that have the highest frequencies selected from the outlier class as “targets”.

Table 5.1 Outlier frequency and distance from mean for 3x3 kernel

3X3 Kernel						
Outlier Frequency				**Distance from Mean**		
Target	**Manhattan**	**Euclidean**	**Mahalanobis**	**Manhattan**	**Euclidean**	**Mahalanobis**
Alpha (294p,199l)	9	9	7	1016	610	7
Bravo (162p,18l)	9	9	7	504	330	3
Charlie (243p,146l)	0	0	2	11	9	2

The covariance matrix (Table 5.2) and spectral plot (Figure 5.7) for the 3x3 kernel surrounding the target pixel *Charlie* suggest that the greatest information content, described by the largest variance, is found in the panchromatic channel. The other channels exhibit lower variance than the panchromatic. The Blue/NIR covariance value is the largest, and the resulting weight for the Mahalanobis distance would be lower than all other covariance values.

Table 5.2 Covariance target *Charlie*, 3x3 kernel

	PAN	**Blue**	**Green**	**Red**	**NIR**
PAN	24.9	0.2	0.2	0.1	-0.1
Blue	0.2	4.1	-0.3	-1.5	-3.2
Green	0.2	-0.3	6.2	-0.1	-2.8
Red	0.1	-1.5	-0.1	0.8	1.4
NIR	-0.2	-3.2	-2.8	1.4	7.3

.

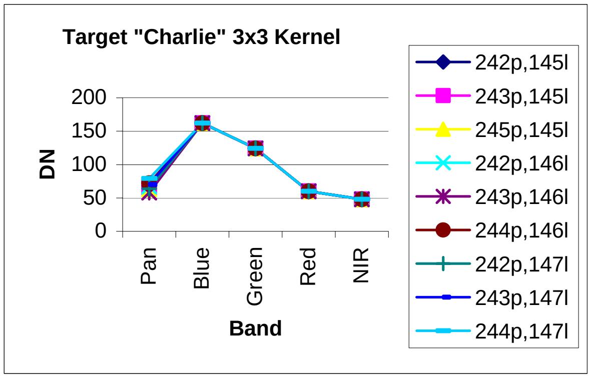


Figure 5.7 Spectral response, target *Charlie*, 3x3 kernel.

In the case of targets *Alpha* and *Bravo*, nine out of nine times the pixel vector was selected as being an outlier for the Manhattan and Euclidean distance metrics. In comparing its values with those of its neighbors in each of the nine individual kernel locations, the pixel was distinct from all the other pixels in the kernel. The distances in all cases were larger than the corresponding distance found for the background pixel vector *Charlie*.

Spectral graphs for targets *Alpha* and *Bravo* were similar. Figure 5.8 shows the response curve for target *Alpha* with a 3x3 kernel. Again, the greatest variance is found in the panchromatic channel. For example, pixel 294*p*, 199*l*, which we know to be part of a vessel, has a very large value in the panchromatic channel. Several other pixels also have high DN values for the panchromatic channel. Target *Alpha* is narrow enough, see Figure 5.4, that when a 3x3 kernel is centered over 294p, 199*l* the left side of the kernel would be over the water. The spectral plot shows that for 295*p*, 200*l* (to the left and below the target pixel vector) that the DN value in the panchromatic is quite a bit lower than those pixels comprising the vessel. Also, the blue, green, red, and especially the NIR have lower values which one would expect for the background water values.

The ability for the NIR to be used to discriminate a target from a background image is hampered by the 4m resolution of the multispectral channels. For example, the NIR response, while low for 295*p*, 200*l*, is not as low as NIR values found in the pixels surrounding the background target *Charlie*. This is due to part of the vessel contributing to the DN values for 295*p*, 200*l* because of the 4m resolution of the multispectral channels. Other pixels with higher responses in the multispectral have more of the vessel contributing to the DN value for those pixels. Clearly, the notion that one can distinguish a vessel from water by relying on the knowledge that water exhibits a very low response in the NIR channel is diminished by the coarseness of the resolution found in the multispectral portion of the IKONOS data cube.

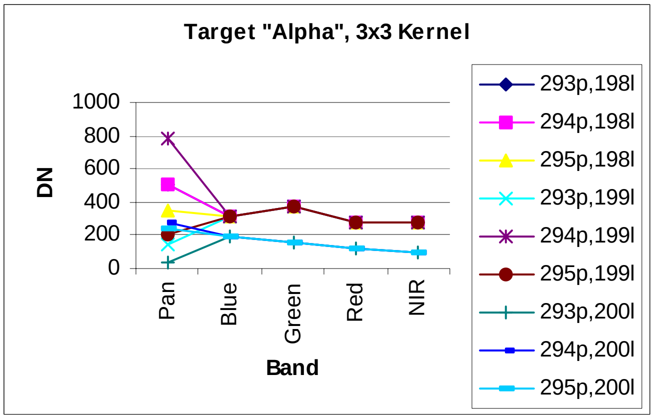


Figure 5.8 Spectral response, target *Alpha*, 3x3 kernel.

Table 5.3 Covariance, target *Alpha*, 3x3 kernel.

	PAN	Blue	Green	Red	NIR
PAN	100689	2112.9	3409.8	3110.5	3356.5
Blue	2112.9	2150.8	3401.7	2947.5	3272.5
Green	3409.8	3401.7	5442.4	4771.5	5225.1
Red	3110.5	2947.5	4771.5	4357.0	4609.2
NIR	3356.5	3272.5	5225.1	4609.2	5044.5

Table 5.4 Covariance target *Bravo*, 3x3 kernel.

	PAN	Blue	Green	Red	NIR
PAN	30904.2	11201.7	15553.6	15478.5	17116
Blue	11201.7	9641.2	13617.2	13505.3	14949.5
Green	15553.6	13617.2	19356.7	19175.7	21281.7
Red	15478.5	13505.3	19175.7	19051.5	21169.2
NIR	17116	14949.5	21281.7	21169.2	23634.5

The covariance matrices are slightly different for targets *Alpha* and *Bravo*, shown in Tables 5.3 & 5.4, respectively. Keeping in mind that target *Bravo* is considerably larger than target *Alpha*, it can be seen in both cases that the panchromatic channel carries the most variance. Relatively speaking, the larger target has a larger variance in the NIR likely due to contributions from a significant number of background water pixels. In the case of the smaller target, *Bravo*, the second largest variance is found in the green channel. In fact, there is a trend in both cases towards a larger variance in the green band. Off-diagonal covariance values exhibit similar relative amounts in that the green/NIR possesses the highest covariance. One would expect that blue/green water values would be correlated with lower values characteristic in the NIR band.

Table 5.5 Inverted covariance (weight) matrix used in Mahalanobis target *Bravo*.

	PAN	**Blue**	**Green**	**Red**	**NIR**
PAN	5.82817E-05	-0.00032	0.000121	0.000157	-9.1E-05
Blue	-0.000316898	0.029378	-0.00665	-0.02855	0.0132
Green	0.000120513	-0.00665	0.021445	-0.01872	0.001577
Red	0.000156729	-0.02855	-0.01872	0.070743	-0.02856
NIR	-9.06579E-05	0.0132	0.001577	-0.02856	0.015921

The Mahalanobis distance can again be thought of as a weighted Euclidean distance. Table 5.5 is the inverted covariance matrix for target *Bravo*. As a result, channels with larger variance and covariances are given lower weight. This would tend to de-emphasize the high variance we see in the panchromatic channel over a target vessel and emphasize lower variance background pixels. The implication of this is a lower likelihood of detecting a small target because we are de-emphasizing the channels that carry information necessary for detection.

5.4.2 5x5 Kernel Results

The results for the larger 5x5 kernel are found in Table 6.6. For all three distance metrics, the larger kernel did a poor job of determining that the target pixels for *Alpha* and *Bravo* were indeed targets. The larger kernel size had greater difficulty handling the larger target *Bravo*. The larger kernel size makes for a greater number of neighbors for a target pixel to be compared to and as a result it is harder to be distinct, particularly for a larger target where the target pixel is being compared not with the distinct background, but with other pixels comprising DN values representative of a target vessel.

Table 5.6 Outlier frequency and distance from mean for 5x5 kernel.

5X5 Kernel						
	Outlier Frequency			**Distance from Mean**		
Target	**Manhattan**	**Euclidean**	**Mahalanobis**	**Manhattan**	**Euclidean**	**Mahalanobis**
Alpha (294p,199l)	25	13	8	1244	710	7
Bravo (162p,18l)	5	2	0	1163	542	3
Charlie (243p,146l)	0	0	0	13	10	4

Figure 5.9, the spectral response curve for the 25 pixel vectors surrounding 162*p*,199*l* of target *Alpha,* is similar to the response curves for target *Bravo*. Figure 5.9 demonstrates the same behavior as the other curves we have investigated - a large range of values in the panchromatic band. The larger sample size in the 5x5 kernel confirms that in the multispectral band, pixels either have generally higher values across those channels indicating they represent a vessel, or they have lower values over those channels – particularly in the NIR – indicating their DN values are being driven by reflectance of the background water.

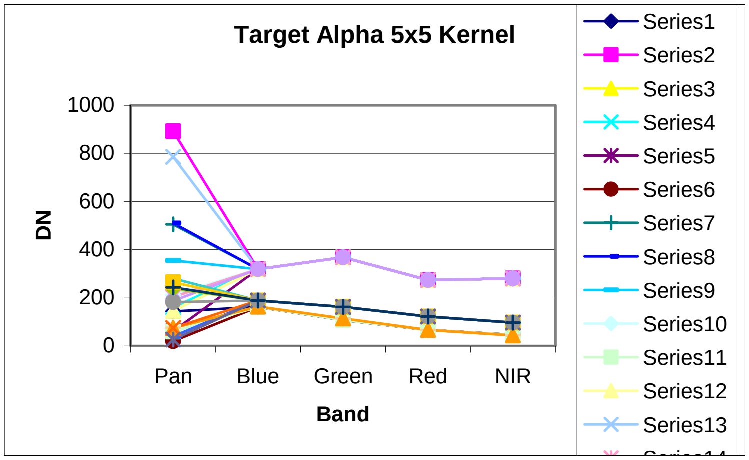


Figure 5.9 Spectral response target *Alpha* 5x5 kernel.

5.4.3 Overall Detection Results

Table 5.7 summarizes the detection results. The performance of the Mahalanobis distance metric was very poor, with a low number of positive detections and a very high number of false positives (i.e. detecting something that isn't there). The best overall performance was found using the Manhattan distance metric in combination with a 5x5 kernel. The Euclidean distance is more sensitive to an increase in DN value from the mean than the Manhattan metric and as a result more false positives are recorded than for the Manhattan distance metric.

Table 5.7 Detection Results.

Detection Results – 16 known targets						
Distance Metric	**Kernel Size**	**Distance Threshold**	**Frequency Threshold**	**Positive Detections**	**False Positives**	**False Negatives**
Manhattan	3x3	750	>8	7	9	7
	5x5	750	>20	16	6	0
Euclidean	3x3	250	>8	16	21	0
	5x5	250	>20	16	29	0
Mahalanobis	3x3	2.3	>8	1	240	15
	5x5	2.3	>20	0	197	16

5.5 Weighted Euclidean Distance

The results above demonstrate that the implementation of the Mahalanobis distance metric demonstrated no improvement in the ability to detect small targets over the Minkowski distance metrics. It only managed to correctly identify a single vessel, and the false positives numbered in the hundreds. Investigation into specific covariance matrices and the weighting they provide reveals that it does not reflect the weighting desired for our purposes.

Reviewing the spectral response curves for the targets, suggests that the targets are distinguished from the background clutter by increased DN values in the Panchromatic, Green, and NIR bands. These channels also carry higher variances and covariance values with the other channels. However, a higher weighting is assigned to lower covariance values using the Mahalanobis distance metric due to the inversion of the covariance matrix.

Equation 5.3, shown below, is a modified Mahalanobis distance. It is identical to the original equation with the single exception being that the covariance matrix ***Cx*** was not inverted before use. This modified Mahalanobis distance will be known herein as the Weighted Euclidean Distance (WED). After implementing the new WED into the software, it was subjected to similar testing as described above.

$$d^2 = (x - m_x)^T C_x (x - m_x) \tag{5.3}$$

where

C_x = the covariance matrix

m_x = the mean vector

5.5.1 WED 3X3 Kernel Results

Tables 5.8 and 5.9 compare the WED distances with those distances generated by the Manhattan, Euclidean, and Mahalanobis metrics for all three targets. The WED metric for targets *Alpha* and *Bravo* is behaving in a superior fashion to the Mahalanobis metrics. In the case of the WED, the range of distance values, between the targets and the background (as represented by target *Charlie*), is much greater than the corresponding range generated by the Mahalanobis metric. By not inverting the C_x matrix, higher weights are being generated for those bands having a greater variability and thus more information. The result is better detection of outliers because the outliers now have much larger distances.

Table 5.8 Distance from mean using Weighted Euclidean Distance for 3x3 kernel.

3x3 Kernel				
Distance from Mean				
Target	**Manhattan**	**Euclidean**	**Mahalanobis**	**WED**
Alpha (294p,199l)	1016	610	6	7527
Bravo (162p,18l)	504	330	2	9747
Charlie (243p,146l)	11	9	2	5

Table 5.9 Distance from mean using Weighted Euclidean Distance for 5x5 kernel.

5x5 Kernel				
Distance from Mean				
Target	**Manhattan**	**Euclidean**	**Mahalanobis**	**WED**
Alpha (294p,199l)	1016	610	7	2357
Bravo (162p,18l)	504	330	2	3637
Charlie (243p,146l)	11	9	2	5

5.5.2 Detection Results for WED

One important lesson coming out from the results was the importance of setting the distance threshold, and to a lesser extent, the frequency threshold for the Manhattan and Euclidean metrics. It was found that the detection algorithm had to be carefully “tuned” to achieve the best detection results. The WED metric was much more robust and did not require such careful tuning.

As seen in Figures 5.4, 5.5, and 5.6, the NIR values are higher for the targets than the background water pixels. In implementing the WED metric, a simple routine was added that filtered out any distances where the NIR value was very low – a characteristic response of water in the NIR band. In this version of MRV Recon, the threshold value is user-defined. In the future, there is no reason why the value could not be automatically calculated - perhaps using a moving average during the calculation of the mean values.

Table 5.10 further illustrates the improvement had by using the WED metric over the other metrics tested. Regardless of the kernel size, the WED scored 16 out of 16 positive detections with no false negatives. While the number of false positives for the WED are much lower than those for the Mahalanobis distance, the values still seem high.

Table 5.10 WED Detection Results.

Detection Results – 16 known targets						
Distance Metric	**Kernel Size**	**Distance Threshold**	**Frequency Threshold**	**Positive Detections**	**False Positives**	**False Negatives**
Manhattan	3x3	750	>8	7	9	7
	5x5	750	>20	16	6	0
Euclidean	3x3	250	>8	16	21	0
	5x5	250	>20	16	29	0
Mahalanobis	3x3	2.3	>8	1	240	15
	5x5	2.3	>20	0	197	16
WED	3x3	*NIR > 100	>8	16	24	0
	5x5	*NIR > 100	>20	16	40	0
*No distances are recorded for those pixels having a very low NIR value – a characteristic response of water in the NIR band						

There is a small string of falsely identified targets in the vicinity of the docks and near shore structures of the Fredericton Yacht Club. However, visual inspection of Figures

5.10 and 5.11 reveals that most of the false positives are surrounding the visible extent of the target yachts.

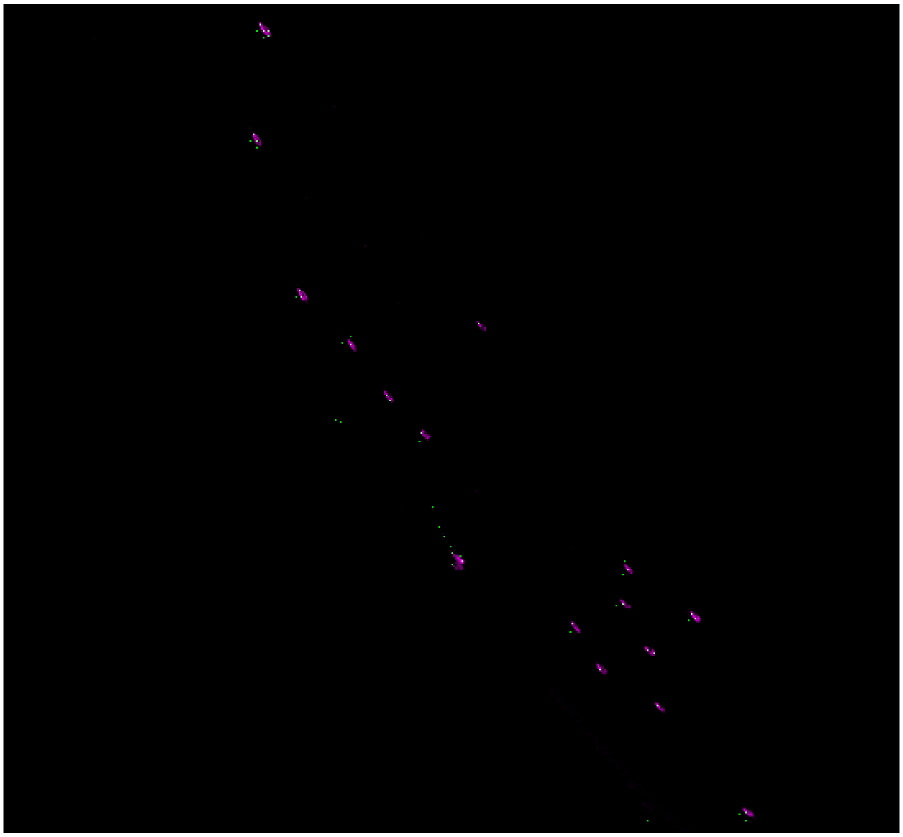


Figure 5.10 Detection results using WED with a 3x3 kernel.

5.6 Proximity Analysis

Recognizing that the blue, green, red, and NIR bands have a larger (4 m) resolution than the 1 m panchromatic channel, a proximity analysis was performed to see if the false positives are a function of the differing resolutions. Figure 5.12 is a bar graph of the results of a proximity analysis. It shows, for every metre outside of a target vessel, the number of false detections.

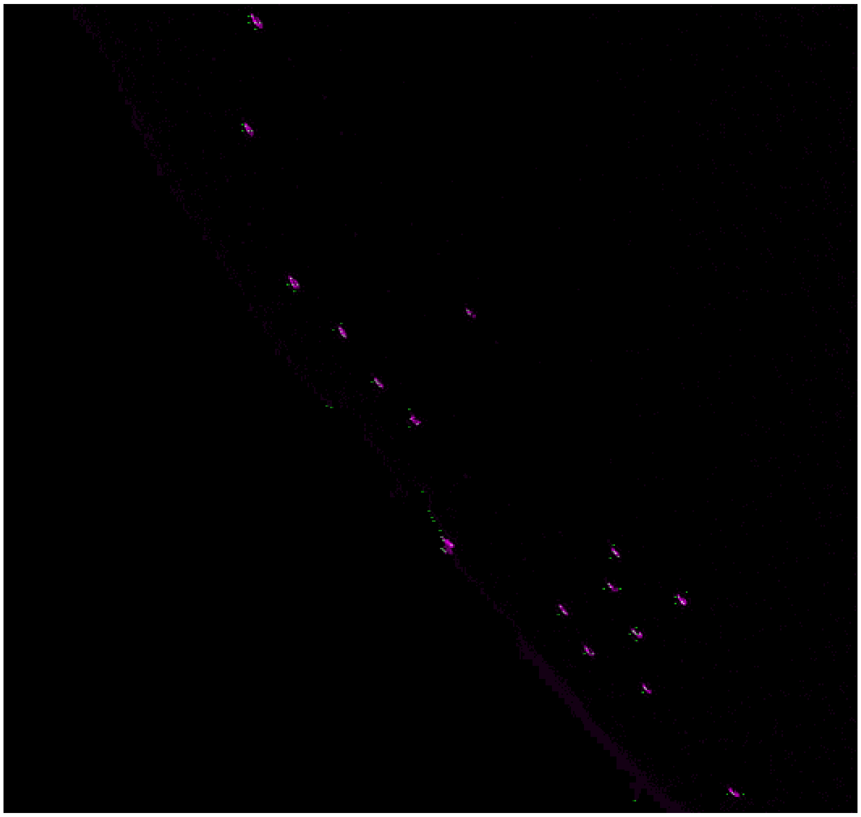


Figure 5.11 Detection results using WED with a 5x5.

The graph reveals that most of the false positives fall either within 4 m of a vessel or greater than 15 m away. Any false positive greater than 15 m away from any vessel is clearly an error in detecting a vessel that is not there. False positives that are less than four metres away from a vessel can be explained by 4 m resolution of the multispectral portion of the image cube. A pixel vector, lying near the extent of a vessel as found in the 1m panchromatic channel, can be selected as an outlier if the multispectral values are significantly influenced by the boat. That is to say, the pixels adjacent to the boat have their DN values altered because they partially overlap the boat.

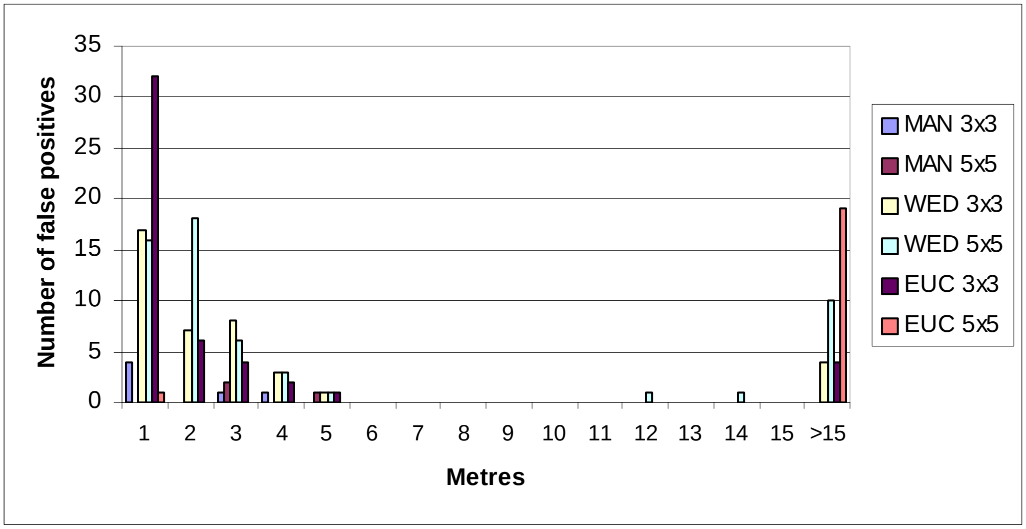


Figure 5.12 Proximity Analysis.

Analysis of the graph in Figure 5.12 clearly shows that the Euclidean metric with a 5x5 kernel performed the worst with 19 false positives more than 15 m away from a vessel.

The Manhattan distance, as demonstrated above, clearly performed the best. Although the overall results for WED did not seem encouraging, the proximity analysis shows that for the case of the 3x3 kernel the performance was acceptable. Of the 24 false positives recorded, only 3 were greater than 15 m away from a vessel. Further, the remaining false positives were within four m of the vessels.

5.6.1 Analysis on Re-sampled Data

At the outset of this work, we stated that we were interested in detecting vessels like the following: canoes, kayaks, PWC and sailing dinghies. The size of these vessels is much nearer the resolution of the IKONOS imagery. Up until now, we have been testing with considerably larger sailing yachts, most well over 6 m in length.

To get a sense of how the various distance metrics would perform on imagery of smaller vessels, the data set was re-sampled, by a factor of 4x, to a lower resolution. Those vessels that once comprised 8 or more pixels in the panchromatic channel, would now only comprise a few pixels.

Table 5.11 shows the detection results on the re-sampled data. The WED metric has superior performance to the other metrics tested. Although the other metrics did not record a false negative – except for the Manhattan metric using a 5x5 kernel – the number of false positives were at least twice as large as the number of false positives recorded for the WED.

Table 5.11 Detection Results for re-sampled data.

Detection Results for *Re-sampled* Data – 16 known targets						
Distance Metric	**Kernel Size**	**Distance Threshold**	**Frequency Threshold**	**Positive Detections**	**False Positives**	**False Negatives**
MAN	3x3	750	>8	16	30	0
	5x5	750	>20	15	14	1
EUC	3x3	250	>8	16	20	0
	5x5	250	>20	16	24	0
WED	3x3	*NIR >100	>3	15	6	1
	5x5	*NIR >100	>5	15	6	1
*No distances are recorded for those pixels having a very low NIR value – a characteristic response of water in the NIR band						

The detection results for the WED metric on the re-sampled data were an improvement to those for the full resolution imagery – the only exception being the recording of a false negative. These results were obtained by adjusting the frequency threshold to a lower value. The lower value suggests that the smaller vessels do not stand out from their neighbors as often and the threshold must be lowered to accommodate this. The target *Bravo* was not detected and thus a false negative was recorded for the 16 known targets in the image. Nonetheless, the overall detection results were excellent with 15 of the 16 targets detected. The number of false positives is proportional to those for the full resolution imagery. Visual inspection of Figure 5.14 again reveals that most of the false positives are closely surrounding the targets suggesting that as found above, the coarser resolution multispectral bands tend to "blur" the results.

Target *Bravo* (41*p*, 5*l*) is located only five lines from the edge of the image and was not detected by the WED. The WED covariance matrix is generated for a 10x10 kernel surrounding each pixel except for those very close to the edges where a smaller 3x3 matrix is used. In the case of target *Bravo*, it appears that the 3x3 kernel was not a large enough sample to produce a C_x matrix that was able to provide sufficient weighting for detection.

To ensure it was not *Bravo's* small size but its proximity to the edge of the image, another small target was investigated. The results, as shown in Table 5.12, of the outlier frequency (the number of times it was selected as being distinct from in neighbors) for target *Delta* were identical to target *Alpha* and the WED were proportional to for those recorded for *Bravo*.

Table 5.12 WED results for re-sampled data.

WED Results for *re-sampled* data				
	Outlier Frequency		**WED**	
Target	**3x3**	**5x5**	**3x3**	**5x5**
Alpha (74p,50l)	9	25	3086	3425
Bravo (41p,5l)	0	0	0	0
Delta (101p, 108l)	9	25	2666	3290
Charlie (55p,41l)	0	0	0	0

5.6.2 Proximity Analysis on Re-sampled Data

In comparing the proximity analysis results on the re-sampled data (Figure 5.13) to those for the full resolution data, it can be seen that the majority of the false positives no longer fall within four metres of the vessels. In fact, all the distances metrics recorded false positives outside this range attesting to the difficulty in detecting small vessels. The WED metric performed the best in this analysis, recording only two false positives beyond the 4 m limit furthering our understanding of WED's performance with smaller targets. The worst performance was the Manhattan metric recording many false positives beyond 4 m from the vessels. It also recorded 6 vessels lying greater than 15 m beyond any vessel.

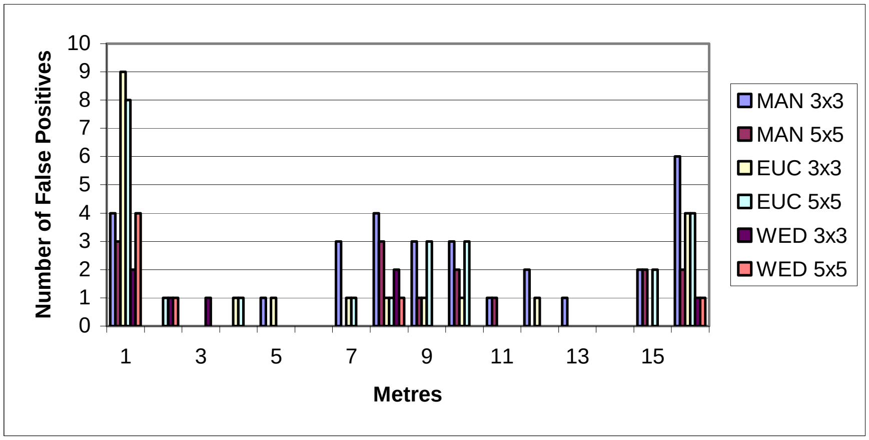


Figure 5.13 Proximity Analysis: Re-sampled data.

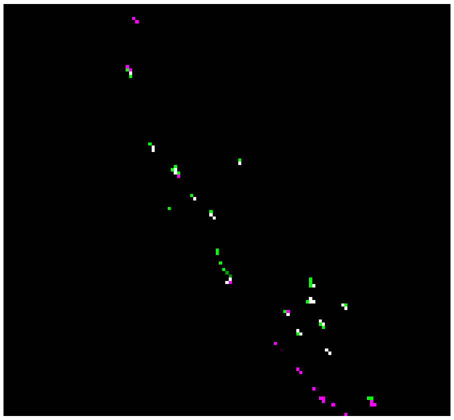


Figure 5.14 WED, 3x3, re-sampled data.

5.7 Conclusions

In working towards a prototype marine recreational vessel reconnaissance system for the Canadian Coast Guard, we are comparing the Manhattan, Euclidean and Mahalanobis distance metrics with a new Weighted Euclidean Distance (WED) to decide on the most suitable for detecting small marine vessels. Of course, this technology could be used in detecting other targets such as breakwaters, piers, navigation structures, and pilings, for applications in: topographic mapping, infrastructure inventories, and pre-engineering design in a near shore marine environment.

The testing was performed using IKONOS imagery known to contain 16 yachts of lengths ranging from 6 – 13 m resting on their mooring at the Fredericton Yacht Club on the Saint John River, New Brunswick, Canada. Three specific targets were investigated regarding their spectral characteristics, relationships with neighboring pixels, and detection performance. Two of the targets, *Alpha* and *Bravo*, are vessels, while the third, *Charlie*, is for comparison purposes a representative pixel vector of the background water.

The performance of the Mahalanobis distance metric was disappointing. It did, however, lead to altering the Mahalanobis metric to create the WED metric. Further testing on the performance of the WED was performed on both full resolution and re-sampled imagery. The re-sampled imagery was used to “mimic” small targets.

Detection results using the full resolution imagery for the two target vessels were superior using the Manhattan metric compared to results obtained using the WED, Euclidean or Mahalanobis distance metrics. The best results were obtained using the Manhattan metric with a 5x5 kernel with all 16 yachts detected, no false negatives, and 16 false positives. Of the 16 false positives, a proximity analysis demonstrated that the majority of the false positives were within 3 m of the extent of target vessels in the panchromatic band. Given that the resolution of the multispectral portion of the image is 4 m this does not seem surprising. However, 4 false positives were found to be at least 15 m from any vessel.

The WED performed best when the imagery is re-sampled to mimic smaller vessels. The WED metric, using either the 3x3 or 5x5 kernel detected 15 of 16 vessels, with one false negative and 6 false positives. Another major advantage is the robustness of the WED metric. As discussed above, the Minkowski metrics had to be “tuned” using thresholds to get optimal results. There were considerable problems with detecting false positives along the shoreline. No such problems existed with the WED distance. Given the emphasis on small targets, and the robustness of the WED metric, it will be the primary distance metric in future versions of MRV Recon.

Now having answered the question that the most appropriate distance metric for use in this application is the WED, work will continue on developing a robust and practical small vessel reconnaissance solution for the Canadian Coast guard. Future work includes refinement of a Weighted Euclidean Distance metric by scaling the weights assigned to the covariance matrix to further emphasize the Panchromatic, green and NIR bands. Other work could include testing on a dataset containing very small targets with associated ground truth, implement and test an expert system approach, and development of an operational reconnaissance system.

5.8 Acknowledgements

The authors would like to thank the support of the Canadian Coast Guard. This research is being done as part of a federally supported Network of Centres of Excellence called GEOIDE (Geomatics for Informed Decisions). Specifically the work is part of GEOIDE project #ENV 60 – Marine Geomatics and Risk Analysis and support from GEOIDE and the Canadian Coast Guard is gratefully acknowledged.

5.9 References

Palliser, J. (2000). Personal communication. Superintendent Victoria Rescue Coordination Centre, Victoria, British Columbia. June.

Pelot R. (2000). "Recreational/Tourism marine activity assessment in the Bay of Fundy, Nova Scotia." Unpublished Report of the Department of Industrial Engineering, Dalhousie University, Halifax, Nova Scotia. March.

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang (2003). "The potential for using very high spatial resolution imagery for marine search and rescue surveillance". GeoCarto International. Vol.18, No. 3, September. pp 35-39.

Trivedi M.M.(1987). "Object detection using their multispectral properties", Proceeding of the Society of Optical Engineering (SPIE), **754**, 255.

Subramanian S. and N. Gat (1998). "Subpixel object detection using hyperspectral imaging for search and rescue operations". Proceeding of the Society of Optical Engineering (SPIE), **3371**, 216.

Gilliam B., S.W. McCandless Jr., L. Reeves, B. Huxtable (1999). "RADARSAT-2 for search and rescue." Proceeding of the Society of Optical Engineering (SPIE), **3718** 189.

Kruzins E., Y.Dong, and B.C. Forster (1998). " Detection of vessels using SAR for support to search and rescue." Proceeding of the 9th Australian Remote Sensing and Photogrammetry Conference. **1**,7304.

Iverson, A.E. (1997). "Subpixel object detection and fraction estimation in hyperspectral imagery", Proceeding of the Society of Optical Engineering (SPIE), **3071**, 61.

Hartigan, J.A. (1975). "Clustering algorithms", John Wiley and Sons.

Munroe, K. M. (2003). “The development of a process to mask out upland portions of high resolution satellite imagery”, Unpublished undergraduate technical report, Department of Geodesy and Geomatics Engineering, University of New Brunswick.

CHAPTER 6
AUTOMATIC SMALL RECREATIONAL VESSEL DETECTION USING IKONOS DATA[3]

This chapter presents a paper submitted to the journal Photogrammetric Engineering and Remote Sensing in April 2004. It is the culmination of all the research presented in the previous chapters. Specifically, it once again describes the imagery collection (Chapter 3) and the preprocessing of that imagery (from Chapter 4). Moreover, the new WED metric (Chapter 5) and other enhancements to MRV Recon are investigated for their ability to detect small recreational boats through a blind test. It describes the findings and then some minor enhancements to the software. The test is run again, and the results are compared with the blind test. Several targets are investigated in depth. The chapter ends with conclusions regarding the results.

6.1 Introduction

Populating a database of marine activities for Canadian waters requires the locations and classifications of small recreational boats. Data on recreational boating are more challenging to acquire than other types of data for marine activities such as commercial fishing or ferry traffic. For example, ferry traffic maintains a schedule and follows well-defined routes. However, since recreational boating is very sporadic in nature, it is much more difficult to get accurate information regarding the location and kinds of activities taking place.

---

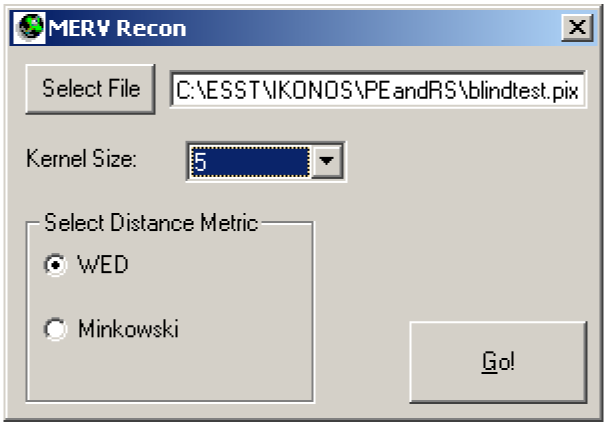
[3] Reprinted by permission of *Journal of Photogrammetric Engineering and Remote Sensing*, n.d.

Researchers at the Department of Industrial Engineering, Dalhousie University have developed the Maritime Activity and Risk Investigation System (MARIS) for the Canadian Coast Guard (CCG) (Pelot et al., unpublished report 2000). MARIS consists of a database containing information on all marine activities. This database supports a risk model that is used to predict areas where future marine search and rescue (SAR) incidents will occur. The final component of MARIS is a GIS to map and analyze the output of the risk model.

The team at Dalhousie expressed the difficulty in getting data on recreational boating activities (Pelot et al., unpublished report, 2000). As recreational boaters account for 76% of SAR incidents in Canadian waters, good data on recreational boating activities are important to get an accurate prediction from the risk model. In addressing the problem of gathering data on recreational boating, Pelot et al. (unpublished report, 2000) used human spotters in aircraft to gather reconnaissance data on recreational boating activities in the Bay of Fundy. In discussing the logistical hurdles in mounting such an operation, the question arose as to whether the new generation of high resolution remote sensing satellites along with automatic target detection software tailored for small recreational boats could be used instead of spotter aircraft. This was the genesis of the Marine Recreational Vessel Reconnaissance system or MRV Recon.

6.2 Objective

The objective of the research described in this paper is to investigate the ability of IKONOS imagery to detect small recreational boats. Further, to continue developing an end-to-end software solution to the small craft target detection problem. The goal for the so-called MRV Recon software is to be fully automated, simple, accurate and robust because its intended user is the CCG. The MRV Recon software was constructed using C language in conjunction with the PCI™ Version 7.0 C/C++ developer's tool kit. A Visual Basic GUI has been developed, as shown in Figure 8.1.



C:\ESST\IKONOS\PEandRS\blindtest.pix

Kernel Size: 5

Select Distance Metric

● WED

○ Minkowski

Figure 6.1 MRV Recon GUI.

6.3 Background

This paper is the third in a series investigating the use of high resolution imagery to detect small boats and describing the development of MRV Recon. The first paper investigated the potential for using high resolution satellite imagery – like IKONOS – for marine search and rescue reconnaissance (Pegler et al., 2003). The second paper (Pegler et al., unpublished Journal of Surveying Engineering paper, 2004) compares maximum distance metrics for use in the remote sensing of small targets.

The first paper reviewed the application of remote sensing technology to marine search and rescue. The literature review revealed intriguing work by Subramanian and Gat (1998) on the use of sub-pixel object detection using hyperspectral imaging for search and rescue operations. The sub-pixel detection method utilized local image statistics based on "spatio-spectral" considerations and was based on the work of Iverson (1997). These works describe a two-stage target detection process involving: (1) examination of local statistics and "outlier" selection; and (2) target selection.

Step (1) requires using distance metrics, specifically the maximum Euclidean or Manhattan distance from the mean, to discern those pixels which are distinct from others in a moving kernel. Those pixels that are distinct are labeled as "outliers". The algorithm counts the frequency for which an individual pixel is selected as an "outlier". The second step (target selection) labels those pixels that have the highest frequency as "outliers" as "targets".

Subramanian and Gat (1998) suggest that, although their spatio-spectral template was designed for use with hyperspectral imagery, it perhaps could be adapted for use with the "soon-to-arrive" (recall that their paper was published just prior to the launch of IKONOS) commercially available high resolution spaceborne multispectral imagery. This assertion led to the automation and implementation of the spatio-spectral template and to some preliminary investigations into the utility of this process for detecting vessels. The early results (Pegler et al., 2003) were satisfactory enough to suggest that, with more work, very high spatial resolution satellite imagery can be effective in accurately detecting small recreational vessels.

The research continued in the second paper that concentrated on adapting and enhancing the spatio-spectral template for use with high resolution multispectral imagery. In particular, it focused on comparing different distance metrics. As previously stated, Subramanian and Gat (1998) used Minkowski distance metrics for their work. In reviewing the literature on clustering algorithms, Hartigan (1975) described the Mahalanobis distance metric (see Equation 8.1).

$$d^2_{mahal} = (x - m_x)^T C_x^{-1} (x - m_x) \tag{6.1}$$

The Mahalanobis distance metric is essentially the Euclidean distance metric weighted by the inverted covariance matrix $\mathbf{C_x}$.

Upon implementing the Mahalanobis distance metric, the principal author realized that, although the idea of using the variance covariance matrix to weight the Euclidean distance was sound, inverting it was having the opposite of the desired effect. The Mahalanobis metric was designed to find *clusters* in data. Clusters by nature have small covariance values. However, in this application we are looking for “outliers” or those things that have large covariance values. As a result, the weighting was altered to favour the “outliers” and not clusters. With the new metric the weighting was altered by not inverting the covariance matrix as was done in the Mahalanobis distance metric.

$$d^2_{wed} = (x - m_x)^T C_x (x - m_x) \tag{6.2}$$

Equation (8.2) shows the new weighted Euclidean distance (WED) metric. The reasoning for use of such a metric is to make use of the covariance information to improve the detection of “outliers” when adapting “spatio-spectral” template for use with multispectral imagery as opposed to hyperspectral imagery for which it was originally designed.

The second paper concluded that the new WED distance was superior to the Manhattan and Euclidean distance metrics for use in small recreational vessel target detection. The WED metric was less sensitive to the thresholds required in the Manhattan and Euclidean distance metrics. Further, it was much more robust in not producing false positives in the shallow waters near shore. Based on the findings of the first two papers, it was decided to continue with the research by further enhancing the software and by generating a data set with associated ground truth to fully assess the detection accuracy of MRV Recon.

6.4 Study Site

The study site for this research is Cadboro Bay, near Victoria, British Columbia, Canada (Figure 6.2). Cadboro Bay was chosen for its favorable weather, year-round recreational boating activity, proximity to the infrastructure provided by the Royal Victoria Yacht Club (RVYC), and support from one of our research partners in the large, local sailing community.

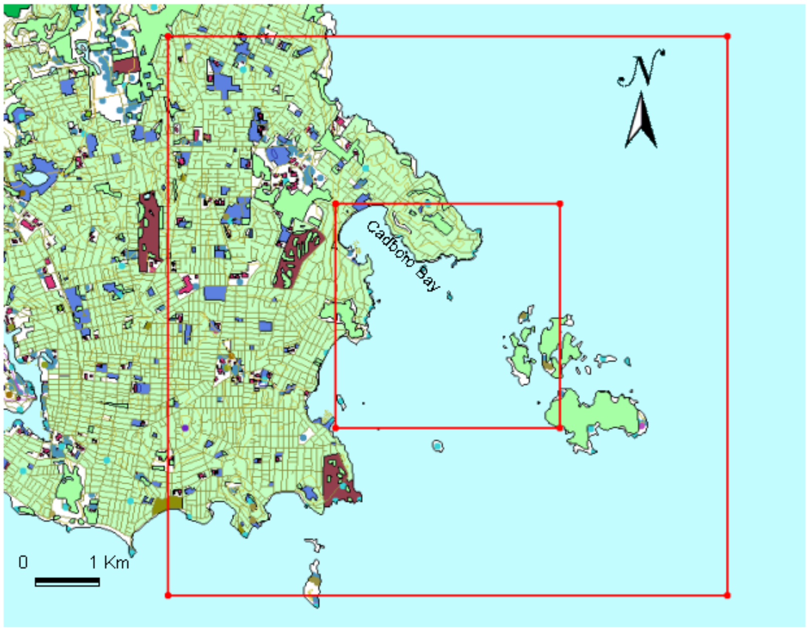


Figure 6.2. The study site: Cadboro Bay, Vancouver Island, British Columbia, Canada.

6.5 Data and Preprocessing

An agreement was reached with SpaceImaging International to purchase a 100 sq. km IKONOS imagery bundle for the area of interest (AOI) illustrated by the outer polygon shown in Figure 6.2. Further, it was agreed that, for a special tasking fee, the imagery would only be purchased if the AOI corresponding to the inner polygon shown in Figure 6.2 was cloud-free. Additionally, the dates/times of the data acquisition would be provided to allow for the deployment of the targets within Cadboro Bay.

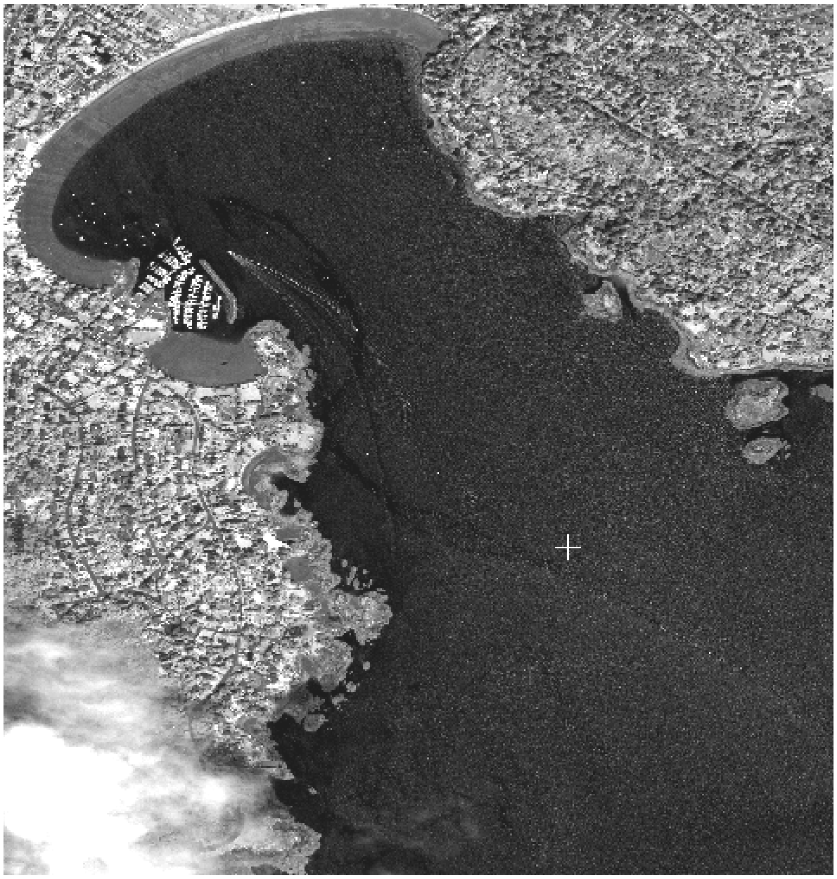


Figure 6.3 IKONOS Panchromatic Image of Cadboro Bay.

Figure 6.3 illustrates the resulting imagery collected on 18 May 2003. Some clouds infiltrated the lower left portion of the bay. However, since no targets were obscured, the imagery was accepted. The RVYC can be seen in the upper left hand side of the Bay. One of the support vessels and its wake can be seen rounding the breakwater to enter the channel to the club's shore facilities. Just northwest of the club, larger sailing yachts can be seen resting on permanent moorings.

Researchers for this project were approached by members of the RVYC with a proposal to perform the necessary work with respect to the gathering of the ground truth. The profits from the work went to the RVYC Junior sailing program.

The RVYC volunteers gathered position and cataloged attribute data for all the targets. Of the 53 targets, 28 were specifically selected for their small size and moored out into the Bay. These 28 targets, in addition to some one-metre diameter steel mooring balls, comprised the Category A targets - all being less than 6m in length. It was not necessary to set out the remaining 25 Category B targets (greater than 6m in length) as they were already resting on permanent moorings – the only exception being a large trawler and its tender resting at anchor in the bay.

Preprocessing of the data consisted of masking out all the land using an automated process through a script developed in PCI Author (Munroe, unpublished UNB undergraduate Technical Report, 2003). Imagery processed after the initial blind test was linearly stretched to enhance the contrast of the targets with respect to the background.

6.6 Methodology

Figure 6.4 presents a flowchart of MRV Recon. After the preprocessing, the first step was to calculate the average of the user-defined kernel for all input channels. Also at this point the means were calculated for the entire panchromatic and NIR channels. These values were then weighted to generate the thresholds for these channels that were utilized when deciding whether to record a WED distance.

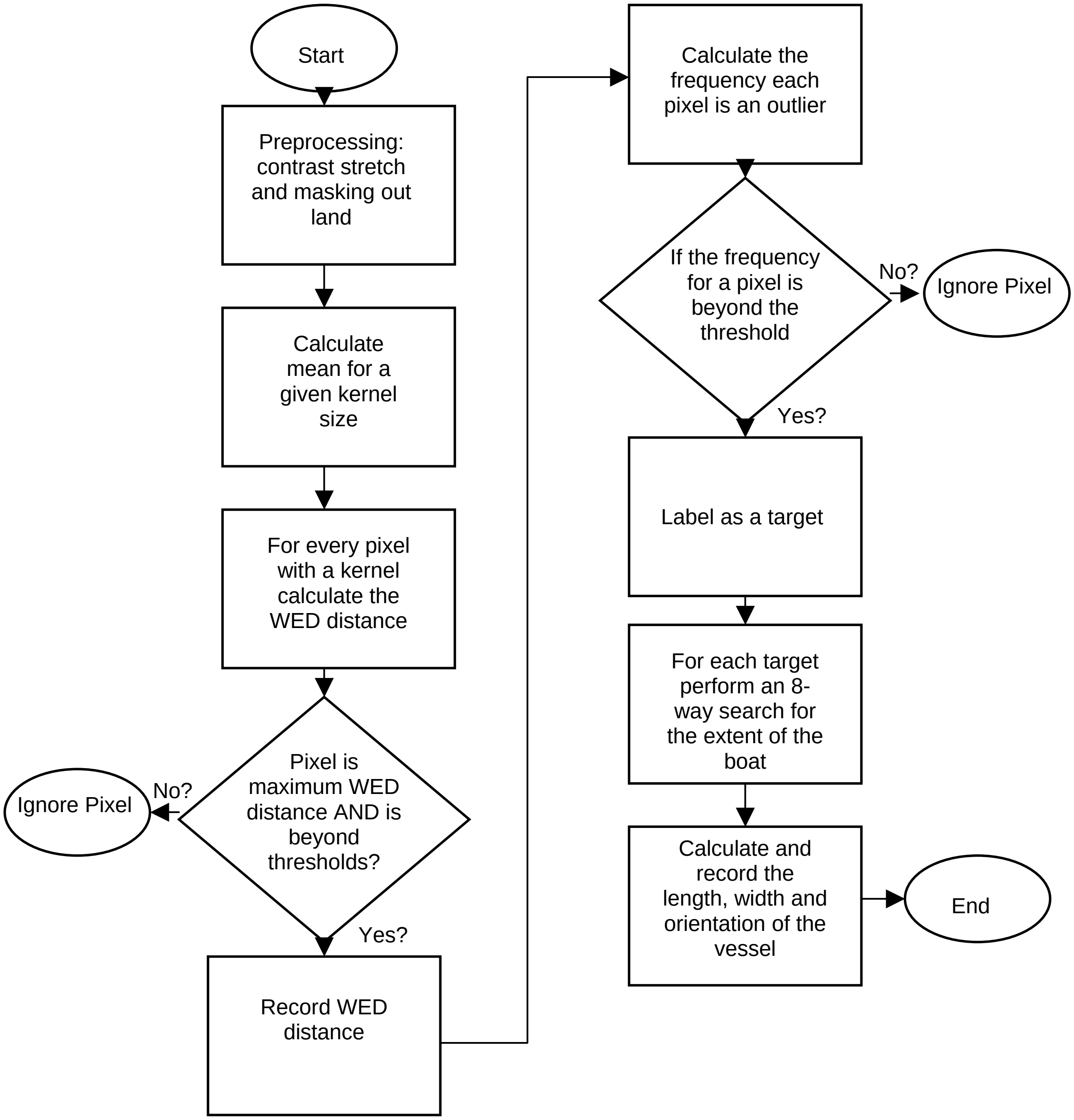

Start
Preprocessing: contrast stretch and masking out land
Calculate mean for a given kernel size
For every pixel with a kernel calculate the WED distance
Pixel is maximum WED distance AND is beyond thresholds?
No?
Ignore Pixel
Yes?
Record WED distance
Calculate the frequency each pixel is an outlier
If the frequency for a pixel is beyond the threshold
No?
Ignore Pixel
Yes?
Label as a target
For each target perform an 8-way search for the extent of the boat
Calculate and record the length, width and orientation of the vessel
End

Figure 6.4 Flowchart for MRV Recon.

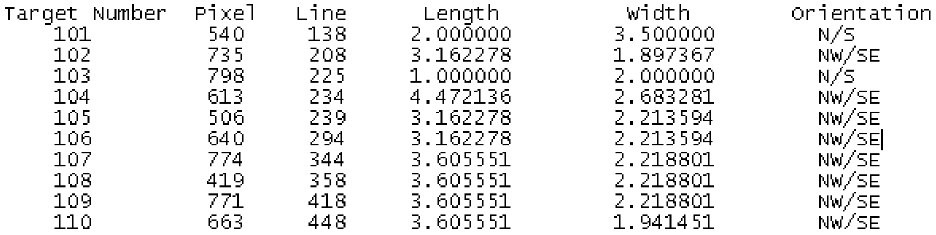
Once the means were calculated, the kernel is passed over the data cube and the covariance was generated for each pixel in the kernel. Then, utilizing the mean vectors, the WED for each pixel was calculated. To drastically reduce the number of potential false positives, the values of the panchromatic and NIR channels were compared with the thresholds calculated earlier. If the values exceed the thresholds, then the WED distance was recorded.

Once all the WED were recorded in a new channel, a kernel was passed over this channel and the pixel with the maximum distance was identified. The pixel possessing the maximum distance was identified as an "outlier" because it was very distinct from the others in a particular kernel location. Each time a pixel was identified as being an "outlier", its "outlier" frequency value was incremented by one. An *nxn* sized user-defined kernel meant that every pixel participated in n^2 kernel calculations. Therefore, the maximum times a pixel could be selected as an "outlier" was also n^2. So for example, if a 3x3 kernel was used, the maximum possible "outlier" frequency was 9. A particular pixel's status was changed from "outlier" to "target" if its "outlier" frequency was at or near the maximum frequency. This frequency threshold can be adjusted, however experience has shown that it should be kept very near the maximum value to reduce the number of false positives.

Having identified the target pixels, they could then be characterized. The characterization algorithm performed a pixel-by-pixel search through the channel containing the target labels. Once a target was encountered, an eight-way search was performed solely on the higher resolution panchromatic channel. Labeling of the target was continued out from a target pixel in each of the eight directions until the edge of the boat was encountered. The edge of the boat was reached when the value of the panchromatic channel fell below a threshold conservatively set at $4\sigma^2$ of the panchromatic channel.

Once the edge of the boat was found for each of the eight cardinal directions, the longest and shortest dimensions were recorded and the length was calculated and recorded. Also the direction of the longest axis of the target yields the orientation of the target vessel plus or minus 180˚. Point targets were assigned a “n/a” value for orientation as they are too small to determine their orientation.

Table 6.1 Output from MRV Recon.



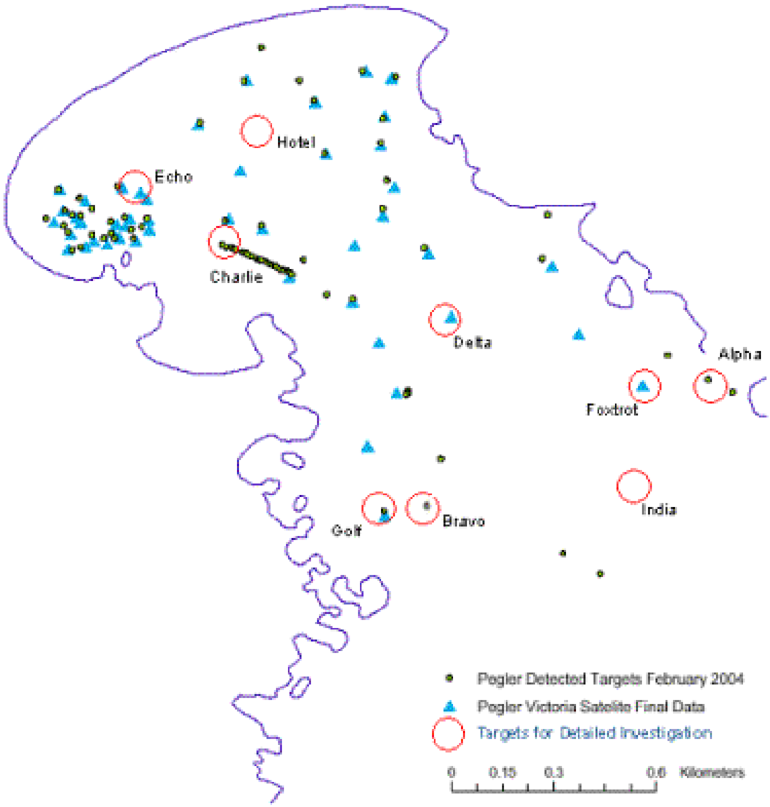
```
Target Number  Pixel   Line     Length          Width          Orientation
     101        540     138    2.000000        3.500000           N/S
     102        735     208    3.162278        1.897367           NW/SE
     103        798     225    1.000000        2.000000           N/S
     104        613     234    4.472136        2.683281           NW/SE
     105        506     239    3.162278        2.213594           NW/SE
     106        640     294    3.162278        2.213594           NW/SE|
     107        774     344    3.605551        2.218801           NW/SE
     108        419     358    3.605551        2.218801           NW/SE
     109        771     418    3.605551        2.218801           NW/SE
     110        663     448    3.605551        1.941451           NW/SE
```

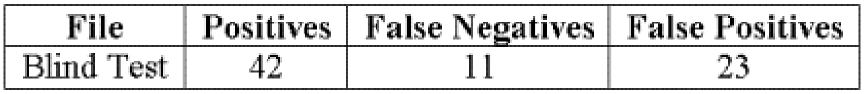
The process continued until all the target vessels were characterized. The output of the characterization was sent to a text file. Table 6.1 is a portion of the text output for this research. Each target was assigned a target number beginning at 100 to avoid confusion with the frequency values. The current version of MRV Recon provides the image coordinates of the centre of the target. Future versions would include geographic or UTM coordinates. Finally, the length, width, and orientation of the targets were printed.

A blind test was used to ascertain the detection ability of MRV Recon. As a result, once the image was collected it was processed with the MRV Recon software and the results were submitted to our research partners in Victoria who retained the ground truth data and then assessed the results of the blind test.

6.7 Results of the Blind Test

Figure 6.5 shows the results of the blind test. Of the 53 targets, 42 were positively detected, and 11 were not detected. The false alarm rate was 23. A large portion of the false alarms were due to the wake of the support vessel clearly visible in Figure 8.3.


Hotel
Echo
Charlie
Delta
Alpha
Foxtrot
India
Golf
Bravo
Pegler Detected Targets February 2004
Pegler Victoria Satelite Final Data
Targets for Detailed Investigation
0
0.15
0.3
0.6
Kilometers



File	Positives	False Negatives	False Positives
Blind Test	42	11	23

Figure 6.5 Accuracy Assessment Blind Test.

Figure 6.5 also shows the locations, depicted with an open circle, of specific targets of interest that were subjected to detailed investigations. These targets were selected as being representative of both point and area targets. In addition, the targets marked for further investigation were selected from all over the bay. Targets Alpha, Bravo, and

Charlie were false positives in the blind test. In the blind test, targets Delta, Echo, Golf, and Foxtrot, were false negatives. Targets India and Hotel were representative of background pixels.

The results from the blind test suggest that improvements were necessary to the MRV Recon software in order to increase the number of positive detections while reducing the false alarm rate.

6.8 Software Enhancements

Only a few minor enhancements were made to MRV Recon following the blind test. Primarily, the size of the sampling window used in generating the covariance matrix was reduced. The final results were achieved using a 5x5 window for calculation of the covariance matrix.

The NIR threshold was increased from 1.4µ to 1.65µ and the panchromatic threshold was also increase from 2.6µ to 2.7µ. Finally, the processing kernel size was altered to a 5x5 kernel from a 3x3 used in the blind test in an attempt to reduce the number of false positives.

6.9 Final Results and Analysis

After making the enhancements to MRV Recon described above, the image was reprocessed. Table 6.2 compares the accuracy assessment of the blind test with those of the final results. The final results show a reduction in the false alarm rate from 23 to 19. However, this was achieved at the expense of not being able to detect a target previously detected in the blind test. Correspondingly, the number of false negatives was increased by 1 from the blind test. Comparing results between the blind test with the final test underscored an important tradeoff of increasing the ability of software to achieve positive detections and also increasing the number of recorded false alarms. The easier the software labels a pixel a "target", the more likely it will include false positives within those targets.

Table 6.2 MRV Recon Accuracy Assessment.

File	**Positives**	**False Negatives**	**False Positives**
Blind Test	42	11	23
Final Test	41	12	19

Table 6.3 Categorized Accuracy Assessment.

	Positives	**False Negatives**	**False Positives**
Category A < 6m			
Blind Test	24	9	23
Final Test	19	12	19
Category B > 6m			
Blind Test	18	2	0
Final Test	22	0	0

While the ratio of positive detections to false negatives stayed nearly the same, there was a reduction in the number of false alarms found in the blind test. Table 6.3 is an accuracy assessment of the blind and final tests broken down by target category.

From Table 6.3, it can been seen that for the most part the larger Category B targets are well handled by the enhanced version of MRV Recon. It detected 22 of the 22 Category B targets. Not surprisingly, there are no false positives in this category. With the exception of the false positives due to the wake of the boat transiting through the AOI, false positives are always very small – on the order of one or two pixels in size.

MRV Recon currently allows these “outliers” to slip through in order to detect as many Category A targets as possible; again this at the expense of an increased false alarm rate. However, it is clear that MRV Recon is now quite robust in detecting the larger yachts improving on the 18/22 Category B detection rate in the blind test to a 22/22 Category B detection rate in the final test.

MRV Recon’s performance on the Category A targets (i.e., those less than 6 m in length) is not as good as that on the larger targets. Figure 6.6 shows that, for the final test, all of the undetected targets are less than 6 m in length. However, it also shows that nearly an equal number of Category A targets were successfully detected. The smallest detected target was a white, fiberglass, dinghy 2.2 m long and 1.14 m wide.

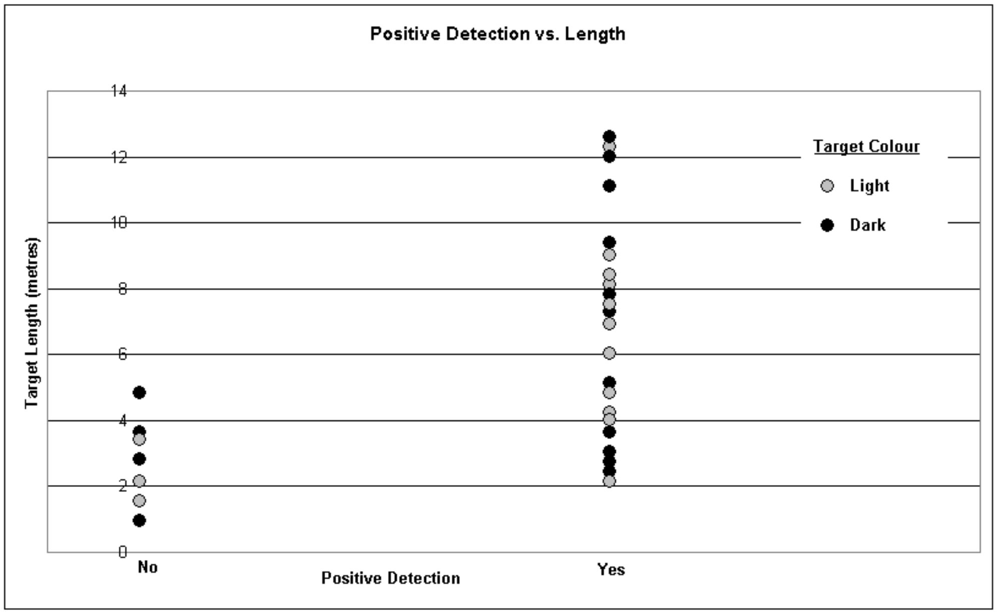


Figure 6.6 Positive Detections vs. Length.

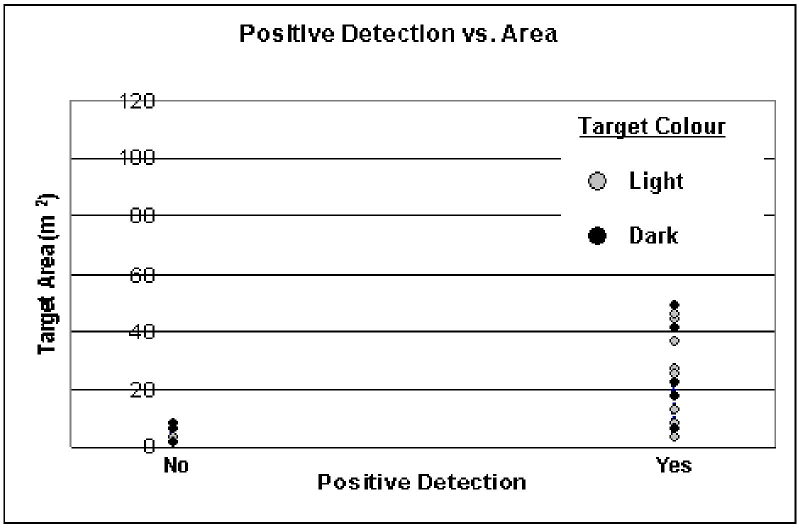


Figure 6.7 Positive Detection vs. Area.

Detection performance with respect to the area of the target is illustrated in Figure 6.7. By area, the smallest detected target has an area of 2.5 m^2. Also shown is that a very small target is less likely to be detected if it has a dark colour. Examples of the targets not successfully detected are:

- Two dark green, 1 m diameter, steel mooring balls,
- A dark pink Optimist dinghy (2.31 m long) Note: both identical white coloured Optimists were detected,
- Two long skinny dark coloured kayaks (5.03 m x .56 m), and
- Two dark grey Byte dinghies (3.66 m long).

Not all the small, undetected targets were dark in colour. Two pairs of yellow plastic yacht racing marks 1.6 x 0.8 m in size were not successfully detected. As with the dark green mooring balls, it is our belief that the very small size rather than colour of these targets is what prohibits their detection.

Figure 6.8 shows a side-by-side comparison of the reflectances from the dark grey Byte dinghy shown on the left and the similar-sized Flying Junior (FJ) dinghy on the right. Tables 6.4 and 6.5 show the pixel vectors and covariance matrices for each of the targets. The darker Byte target has lower values across the pixel vector for the five bands than the white FJ dinghy.

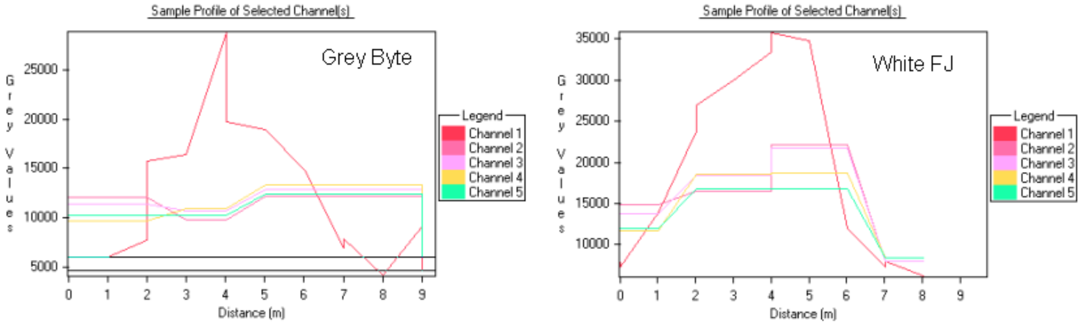


Figure 6.8 Reflectance Cross-sections for Targets Golf and Bravo.

Table 6.4 Reflectance Values and Covariance Matrix for Target Golf (Grey Byte).

Byte	17282.000	12150.000	11511.000	9725.000	10376.000	2287.664
	PAN	**Blue**	**Green**	**Red**	**NIR**	**WED**
PAN	6.495	0.000	0.000	0.000	0.000	
Blue	0.899	1.409	0.000	0.000	0.000	
Green	-19.670	-9.078	210.339	0.003	0.001	
Red	-23.715	-12.123	321.688	619.493	0.003	
NIR	-3.315	-10.761	123.617	356.872	766.004	

Table 6.5 Reflectance Values and Covariance Matrix for Target Bravo (White FJ).

FJ	33423.000	16545.000	18491.000	18574.000	16923.000	3795.976
	PAN	**Blue**	**Green**	**Red**	**NIR**	**WED**
PAN	1504.793	0.005	0.003	0.001	-0.001	
Blue	519.032	819.209	0.009	0.006	-0.001	
Green	332.548	950.223	1578.309	0.013	-0.001	
Red	-17.092	478.626	1139.768	1003.390	0.000	
NIR	-44.914	-30.955	4.058	33.579	79.613	

The covariance values for a 5x5 window surrounding the center of the target shows that for the white FJ dinghy the variances are much higher – except for the NIR channel – than those recorded in the covariance matrix for the Byte dinghy. This provides a higher weighting for the WED distance for the FJ than the Byte. Thus the WED distance for the FJ is 3796 vs. 2287.7 for the Byte. So for this example, when the targets are not extremely small (as in the mooring balls), colour does play an important role in the ability for MRV Recon to detect smaller targets.

When the colour is dark, the spike in the panchromatic channel is not large enough to compensate for the smaller contributions of a small target to the 4 m resolution multispectral channels. A question needing to be explored is - what other characteristic(s) of the undetected Category A targets are causing them to remain undetected? Further, what are the causes of the false alarms?

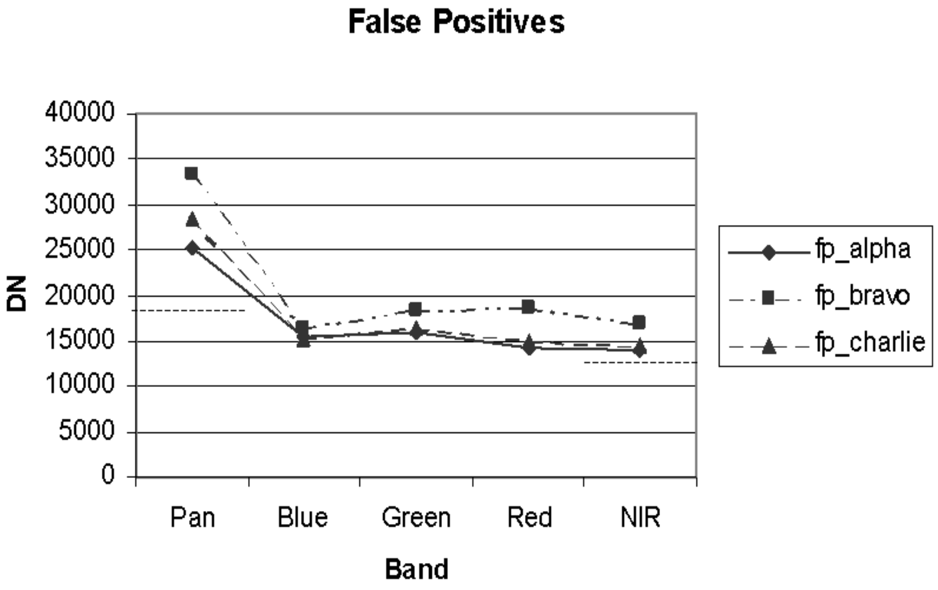


Figure 6.9 False Positives Response Curves.

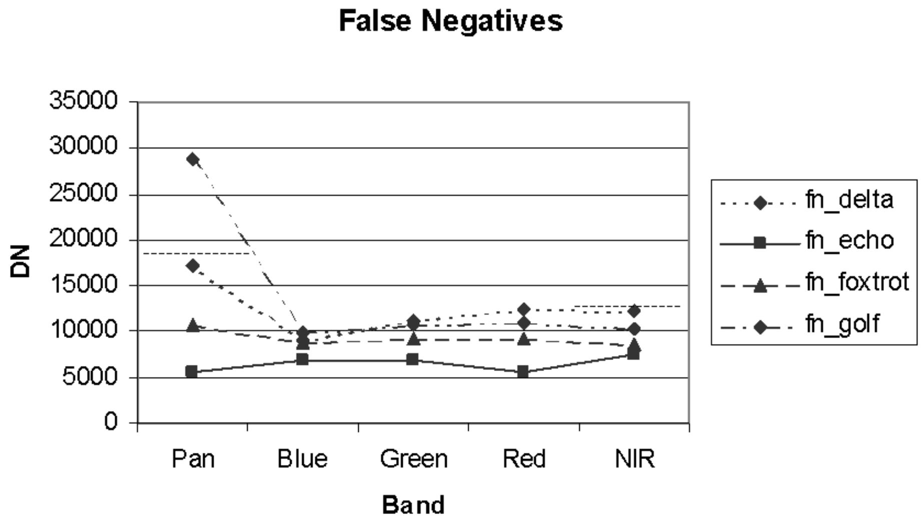


Figure 6.10 False Negatives Response Curves.

Figures 6.9 and 6.10 plot the digital number (DN) against the bands for the false positives and false negatives respectively. Figure 6.11 is a similar plot but for two samples of the background. In all three graphs, a dashed line is drawn on the columns for the panchromatic and NIR channels. These lines represent the threshold values set for these channels.

Comparing the graphs for the background pixels and the false positives begs the question: what is causing a background pixel, like Alpha, to have high enough NIR and panchromatic values to allow it to pass through the thresholds? Moreover, what causes the false positive to have a high enough frequency to be selected as an “outlier”? And finally, what is causing it to have a high enough threshold ratio to be labeled as a target?

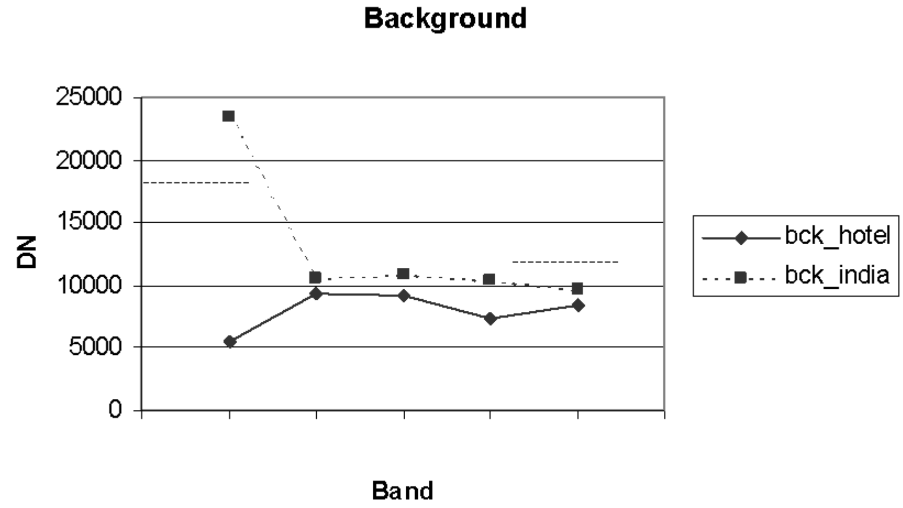


Figure 6.11 Background Response Curves.

It was anticipated that sun glint and breaking waves might cause some localized “outliers” driven by an increased value from the panchromatic channel. As a result, the NIR threshold was introduced and then later the panchromatic threshold. However, in rare cases like target Alpha and portions of the wake, both the panchromatic value and the NIR values are high enough to pass the thresholds and generate a large enough WED distance that they are falsely labeled as targets.

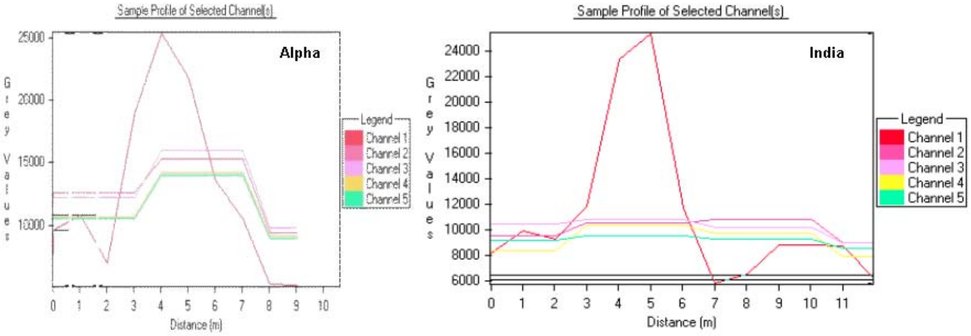


Figure 6.12 Comparison of Cross-section for Target Alpha and India.

Figure 6.12 compares the measured reflectance curves for the Alpha (a false positive) on the left and India (background) on the right. The reflectance curves are for a cross-section through each of the targets and, to get an idea of the surroundings, through some of the background. In both cases there is a sharp increase in the panchromatic band. However, the false positive is distinct from the background sample, India, in that it also experienced higher values in the blue, green, red, and NIR bands. The ground truth recorded no target in the vicinity of Alpha.

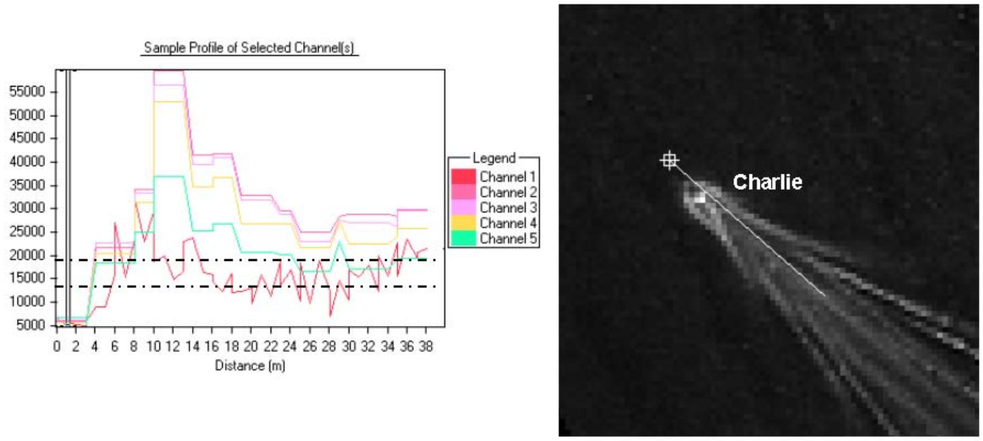


Figure 6.13 Cross-section through target Charlie and its wake.

Figure 6.13 is a cross-section of the support boat and part of its wake, labeled as target Charlie. This cross-section allows for direct comparison of a valid target and the false positives generated by a wake. On the right-hand side of Figure 6.13 is the panchromatic image of the moving boat showing the location of the cross-section. On the left-hand side are the five reflectance curves for the cross-section. The dashed lines drawn across the reflectance curves indicate the panchromatic and NIR thresholds.

From left to right, the curves begin with low reflectance values characteristic of the background water in front of the moving boat. Then the curves sharply increase corresponding to the higher reflectance values from the target vessel. The reflectances then diminish to a jagged, saw-toothed pattern over the wake. The peaks of the saw teeth are sufficiently high to be over the thresholds and thus get falsely identified as a target.

Although the false positives due to the wake indicate a different scenario than the point target false positives like Alpha, it seems that some physical property is causing the multispectral reflectance values to increase. In the case of the wake, clearly the turbulence of the propeller is the catalyst of this phenomenon.

The catalyst for the point target false positives is not as readily apparent as that for a wake. It could be a random sensor error, or something on the water with no ground truth such as a crab pots (we did observe active fishing activities taking place during the setting out of targets). In any case, it would be simple enough to remove them all by creating a function within MRV Recon to ignore all single pixel targets. Unfortunately, at the resolution of IKONOS, we would be eliminating any possibility of detecting the very small targets of interest. However, if higher resolution imagery was used such as QuickBird this would be an option for reducing the false positives.

The purpose of MRV Recon is not only to detect marine recreational vessels but also to characterize them. Figures 6.14 and 6.15 compare the generated lengths and widths with those measured during the ground truth. For the most part, the generated lengths and widths are shorter than those measured during the ground truth. The average deviation for the length and width is 1.59 and 0.76 m, respectively. Clearly the threshold set within the characterization function for finding the edge of the boat is slightly too conservative. However, since the resolution is 1 m in the panchromatic channel, this means that the targets are being represented generally within in a few pixels of their correct length and width.

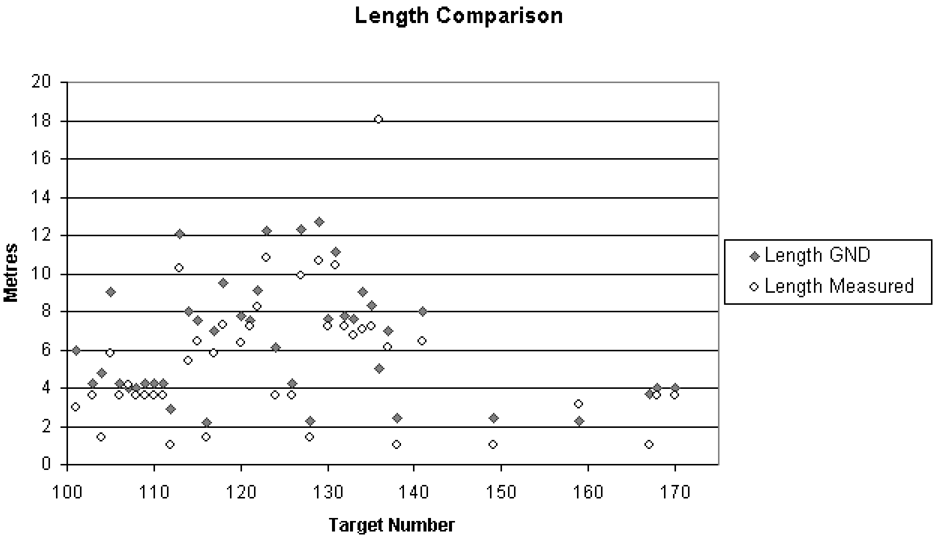


Figure 6.14 Comparison Between Measured and True Length for Targets.

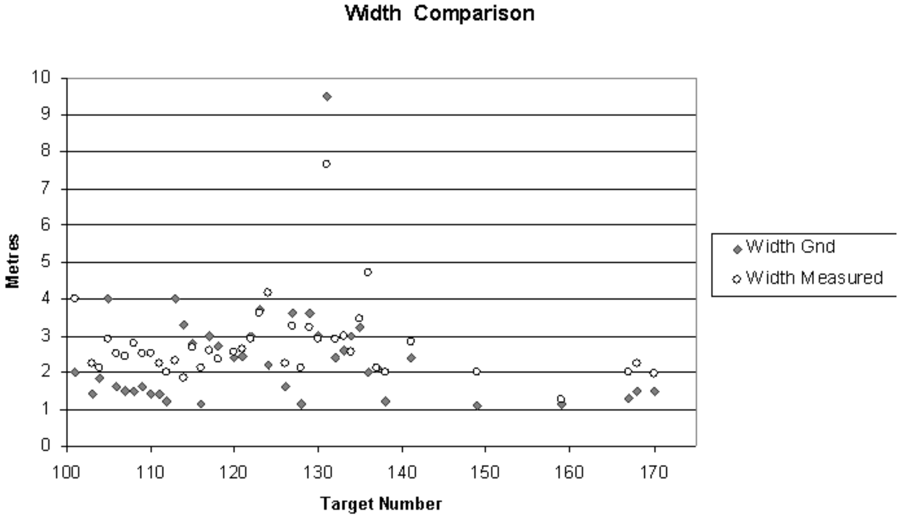


Figure 6.15 Comparison Between Measured and True Width for Targets.

6.10 Conclusions

As was previously stated, the main objective this research is to demonstrate that very high resolution satellite imagery can be used to detect small recreational boats. A total of 53 targets were set out in Cadboro Bay, near Victoria, British Columbia. IKONOS imagery was collected over the AOI in May 2003.

The additional objective of this work was the development of automatic target detection software that doesn't require the use of a priori knowledge of the target such as spectral libraries. A robust solution was built using the C language called MRV Recon. The imagery was processed using MRV Recon software that uses a weighted Euclidean distance metric to detect small recreational vessels. A blind test generated a 79% detection rate with 11 false negatives and 23 false positives.

A final test was performed after making some minor enhancements to the MRV Recon software. In that test, the detection rate was found to be 77% with 12 false negatives, but only 19 false positives.

An investigation into the false positives has revealed that a large number of the false positives were caused by the wake of a boat. In addition, investigations of the reflectance curves for selected false positives suggest that something is creating higher multispectral values. There appears to a correlation between the boat's wake and the unknown process. Further investigation of this process is necessary to lower the number of false positives generated by MRV Recon.

In the final testing, 61% of targets shorter than 6 metres and 100% of the targets greater than 6 m in length were detected. The smallest target detected was 2.2 m long and 1.1 m wide.

The analysis also revealed that the ability to detect targets between 2.2 m and 6 m long was diminished if the target was a dark colour. With a few exceptions MRV Recon was able to determine the length and width of the target vessels within a few pixels of their correct size.

This research demonstrates that that very high resolution satellite imagery can be used to detect small recreational boats by processing IKONOS data with MRV Recon. It is anticipated that using higher resolution data such as that from QuickBird would produce superior results.

6.11 References

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang, 2003. *The potential for using very high spatial resolution imagery for marine search and rescue surveillance*, GeoCarto International, 18(3):35-39.

Subramanian S. and N. Gat, 1998. *Subpixel object detection using hyperspectral imaging for search and rescue operations*, SPIE, **3371**, 216-225.

Iverson, A.E., 1997. *Subpixel object detection and fraction estimation in hyperspectral imagery*, SPIE, **3071**: 61-71.

Hartigan, J.A., 1975. *Clustering Algorithms*, John Wiley and Sons.

Note: While differing from the referencing requirements of the University of New Brunswick, the referencing style of the journal of Photogrammetric Engineering and Remote Sensing only allows the inclusion of refereed journals and published books into the references. All other materials must be referenced within the body of the article as done herein.

CHAPTER 7
RESULTS, CONCLUSIONS, AND FUTURE RESEARCH

The empires of the future are the empires of the mind.
- Sir Winston Churchill, Speech at Harvard University, September 1943

Although the journal paper included in Chapter 6 describes the majority of the results, some of the details surrounding the results are included in this final chapter due to space limitations imposed by the journal. Further, while some conclusions were drawn in Chapter 6, they will be expanded upon in this chapter. Finally, recommendations for further work will be made.

7.1 Results

Figure 7.1 illustrates an enlarged portion of the final results. Using the 21 m total error value (see section 3.8) in determining positive detections, MRV Recon's overall detection accuracy is 77 %. The targets are broken down into two categories: A) less than 6 m in length, and B) greater than 6 m long. The detection rate for targets greater than 6m long was 100%. The detection rate for targets less than 6 m in length is 61%. The smallest correctly detected target was 2.2 m long and 1.1 m wide.

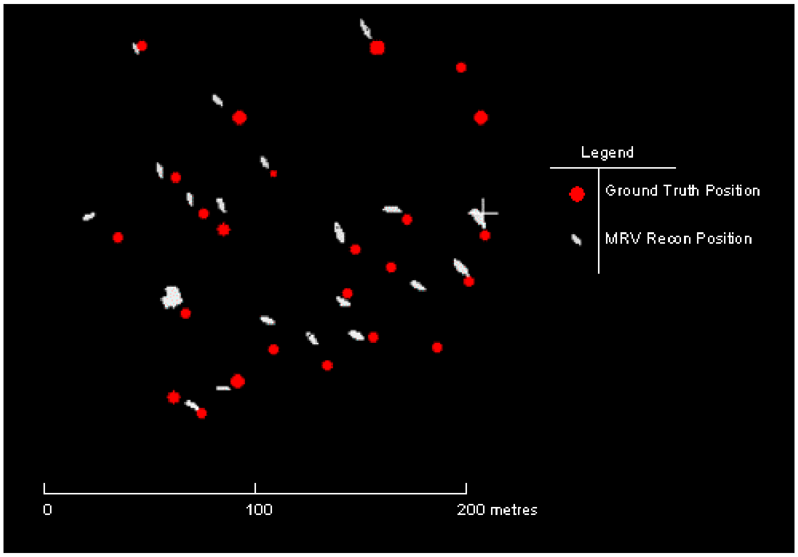


Figure 7.1 Detection results.

Table 7.1 Cumulative Binomial distribution for determination of the true error rate.

Number of Missed Targets	**N=53, p=.30**	N=53, p=.35	N=53,p= 42	N=53,p=0.43	N=53, p=.50
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000002	0.000000	0.000000	0.000000	0.000000
3	0.000013	0.000000	0.000000	0.000000	0.000000
4	0.000074	0.000003	0.000000	0.000000	0.000000
5	0.000330	0.000019	0.000000	0.000000	0.000000
6	0.001207	0.000087	0.000001	0.000001	0.000000
7	0.003733	0.000333	0.000006	0.000003	0.000000
8	0.009955	0.001094	0.000025	0.000014	0.000001
9	0.023289	0.003142	0.000095	0.000054	0.000003
10	**0.048434**	0.007994	0.000319	0.000188	0.000012
11	0.090559	0.018208	0.000953	0.000584	0.000045

Not detecting 11 of the 53 targets yields a 21% observed error rate. Table 7.1 is the cumulative binomial distribution for up to 11 undetected targets out of a total of 53 (targets). Using a similar process to that discussed in Chapter 3, it can be said with 95% (1 - 0.048434) confidence that - having not detected 11 targets in 53 targets - the true error rate is no more than 30% (p = 0.30). A 30% true error rate suggests that the observed error rate is high and this is because we deliberately selected smaller targets to test the detection limits of the MRV Recon using IKONOS imagery. The end result is a higher true error rate than desired but, when put in context, makes sense.

7.2 Conclusions

At the outset of this work, the hypothesis of this research was "to prove that commercially available high resolution IKONOS satellite imagery can be used to detect small recreational vessels [Pegler, 2003]". The research has demonstrated that IKONOS imagery is effective in detecting small recreational vessels. Clearly, with a 100% detection accuracy for recreational vessels over 6 m in length, it is very effective with the larger sized recreational vessels. Evidently, it is less effective for targets less than 6 m long. It is important to remember that many of the targets selected in this category were specifically chosen to push the detection limits of IKONOS and MRV Recon.

Three years after asserting this thesis, it is somewhat dated. IKONOS is no longer the only commercially available high resolution satellite imagery. QuickBird is a major competitor. It is a very similar product, basically the same 5 bands as IKONOS, but with a higher resolution. QuickBird has a resolution of 0.64 m in the panchromatic band versus IKONOS' 1 m resolution. In addition, QuickBird has a resolution of 2.5 m in the multispectral bands versus IKONOS' 4 m resolution. This (almost double the resolution) data should yield superior detection accuracy in the under 6 m in length targets with no changes necessary to MRV Recon.

In the end, a working reconnaissance system for small recreational vessels has been developed.

7.3 Future Research

The following is a list of recommendations for future research.

1. As previously described in Chapter 2, a “material of construction” system should be researched.
2. The detection accuracy of MRV Recon should be tested with a dataset similar to that gathered in Cadboro Bay, but using the higher resolution QuickBird, or similar imagery.
3. When more experience is gained by using MRV Recon on additional imagery, investigations should be made in moving from empirically dervived thresholds for the NIR, PAN, and TR values to thresholds automatically calculated from image statistics.
4. With the availability of higher resolution imagery, more emphasis is being paid to contextual information. For example, detecting fire hydrants would be restricted to areas between the edge/curb of a road and the roadside edge of a sidewalk. If an object suspected of being a fire hydrant is outside of this contextual relationship, then it is removed as a possible target. For vessel detection, one might construct a contextual rule that a target larger than a few pixels in size, should have a pointed bow and a long narrow aspect. Although that is an extremely simple example, the point is that MRV Recon relies heavily on spectral characteristics to label targets. Further work should be done to leverage the contextual information resident in higher resolution imagery.

5. A brief attempt was made to use the PCI Toolkit to access the Image Handler function to display the results of the target detection. This function couldn't be made to work and was abandoned in favour of more pressing research. Future versions of MRV Recon would include display functionality and perhaps the ability to query the results for individual targets.
6. MRV Recon should be part of a larger trial by the CCG or MARIS. This trial should mimic as closely as possible the operational environment in which these systems would operate. The only way to access the utility of MRV Recon is to let those for whom it is intended try it out to see if it meets their needs.
7. In the case of MRV Recon, it has primarily been implemented as a proof of concept system. This is not to say that any consideration for practicality was set aside; in fact at the outset decisions such as restricting the design to an automatic system were made. However, some areas were neglected in favour of getting the basic system working. For example, little work was done trapping errors. Any operational system would need to trap any errors made by the user such as selecting an incorrect file type, or informing the user of a corrupted file. Although implementing error handling is very time consuming, any operational system without this functionality would not be very useful.

7.4 Accomplishments and Contributions

Table 7.2 summarizes the objectives of this research as described in the proposal [Pegler, 2001]. Chapters 2 and 5 described the development of an automatic marine recreational vessel reconnaissance system called MRV Recon. Chapter 2, described the implementation of the spatio-spectral template. Chapter 3 outlined the design and creation of a data set of small targets with ground truth. Chapter 4 outlined the preprocessing methods developed to mask out the upland portions the imagery. Chapter 5, the second journal article, described the development of the new WED metric as an enhancement to the spatio-spectral template. The accuracy of the detection and characterizations were described in the final journal paper found in Chapter 6.

Table 7.2 Research Objectives

Objectives	**Satisfactorily Completed**	
	Yes	**No**
Develop a method to Mask out Upland area	✔	
Implement an automated vessel detection system based on Spatio-Spectral Template	✔	
Develop a vessel characterization System	✔	
Create a small target dataset with ground truth	✔	
Enhance the Spatio-Spectral Template	✔	
Develop a prototype marine recreational vessel reconnaissance system	✔	
Material of Construction System		✔

When setting the objectives for this research, an objective to develop a “material of construction system” was included in the proposal. It was envisioned that once a target was labeled, its spectral signature would be used in determining its construction material. There are two factors to explain why this objective is not met. First, the challenge and difficulty to develop a reliable small target detection system was greatly underestimated. Towards the middle of the research and subsequent to attending a target detection conference it became apparent that this would be the most demanding objective. Secondly, the coarseness of the multispectral bands of IKONOS was explored in Chapter 5 when comparing the Minkowski and WED metrics. This coarseness of resolution makes the job of identifying the material of construction difficult. For these reasons, this objective remains uncompleted, for now.

Despite the lack of a method to identify the material of construction of the target vessels, it has been shown that MRV Recon will provide the Canadian Coast Guard with a unique and effective tool for gathering crucial data on recreational vessels.

REFERENCES

Ashton, E. (1999). “Multialgorithm Solution for Automated Multispectral Target Detection.” *Opt. Eng.*, Vol. 38, No. 4, pp. 717-724, April.

Dare, P. (2003). Personal Communication. Professor and Chair, Department of Geodesy and Geomatics Engineering, Univerisity of New Brunswick, Fredericton, New Brunswick. November.

Eldhuset, K. (1996). “An Automatic Ship and Ship Wake Detection System for Spaceborne SAR Images in Coastal Regions.” *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 34, No. 4, pp. 1010 – 1019, July.

Gilliam B., S.W. McCandless Jr., L. Reeves, and B. Huxtable (1999). “RADARSAT-2 for Search and Rescue”. *SPIE Conference on Automatic Target Recognition IX*, Orlando, Florida, SPIE Vol. 3718, pp.189-194, April.

Hartigan, J.A. (1975). *Clustering Algorithms*, John Wiley and Sons.

Iverson, A.E. (1997). “Subpixel Object Detection and Fraction Estimation In Hyperspectral Imagery”, *SPIE*, Vol. 3071, pp. 61-71.

Hongyan S. and M. Shiyi (1995). “Multisensor Data Fusion for Target Identification”. *Chinese Journal of Electronics.* Vol. 4, No.3, July 1995. pp 78-84.

Kruzins E., Y.Dong, and B.C. Forster (1998). “Detection of Vessels Using SAR for Support to Search and Rescue”. *Proceeding of the 9th Australian Remote Sensing and Photogrammetry Conference*. Vol. 1, p.7304. July 20-24.

Munroe, K. M. (2003). “The Development Of A Process To Mask Out Upland Portions Of High Resolution Satellite Imagery”. Unpublished undergraduate technical report, Department of Geodesy and Geomatics Engineering, University of New Brunswick.

Palliser, J. (2000). Personal Communication. Superintendent Victoria Rescue Coordination Centre, Victoria, British Columbia, June.

Parker, J. (1997). *Algorithms for Image Processing and Computer Vision,* John Wiley and Sons.

PCI Geomatics (2000). *PACE C Programmer’s Reference*, Version 7.0.

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang (2004). “Comparison of Maximum Distance Metrics for use in the Remote Sensing of Small Targets”, *Journal of Surveying Engineering*. Accepted for publication.

Pegler, K., D.J. Coleman, R. Pelot, and C.P. Keller (2004). "Automatic Small Recreational Vessel Detection using IKONOS Data", *Photogrammetric Engineering and Remote Sensing*. Submitted for review.

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang (2003). "The Potential For Using Very High Spatial Resolution Imagery for Marine Search and Rescue Surveillance", *GeoCarto International*, Vol.18, No. 3, pp.35-39.

Pelot R. (2000). "Recreational/Tourism Marine Activity Assessment in the Bay of Fundy, Nova Scotia ". *Unpublished Report of the Department of Industrial Engineering, Dalhousie University*, Halifax, Nova Scotia, March.

Rancourt, M. Captain, K. Pegler, D.J. Coleman, R. Pelot [2002]. "Development of an IKONOS Coverage Prediction Application". Proceedings of The 95th CIG Annual Geomatics Conference / Joint International Symposium on Geospatial Theory, Processing and Applications. Ottawa, Jul 8-12.

Randell, C., M. Rokonuzzaman, J. Youden, and R. Khan (1999). "Evaluation of RADARSAT for Detection, Classification and Discrimination of Icebergs", Paper Presented at the IEEE Newfoundland Electrical and Computer Engineering Conference, St. John's, November.

Scheaffer, R. and J. McClave (1995). *Probability and Statistics for Engineers*, Duxbury Press.

School of Graduate Studies, University of New Brunswick (n.d.). *Graduate Student Handbook.* Retrieved 9 February 2004 from http://www.unb.ca/gradschl/guidelines/thesis_regulations.pdf.

Shirvaikar, M. and M. Trivedi (1990). "Design and Evaluation of a Multistage Object Detection Approach", *SPIE* Vol. 1293, pp 14-22.

Shortis, M.R., T.A. Clarke, and T. Short (1994). "A Comparison of Some Techniques for the Sub-pixel Location of Discrete Target Images". *SPIE Conference on Videometrics III*, Boston, *SPIE* Vol. 2350. pp 239-250.

Space Imaging, Inc.(2000). "CARTERRA Geo Technical Specs" Retrieved 17 November, 2000. http://www.spaceimaging.com/carterra/geo/prodinfo/geotech.htm.

Subramanian S., and N. Gat (1998). "Sub-pixel Object Detection Using Hyperspectral Imaging for Search and Rescue Operations". *Proceeding of the SPIE Conference on Automatic Target Recognition VIII*, Orlando, Florida. Vol. 3371. pp. 216-225, April.

Tiberius, C. (2003). “Standard Positioning Service Handheld GPS Receiver Accuracy.” *GPS World.* pp 44-51, February.

Tingley, M. (2003). Personal Communication. Director, Applied Statistics Centre, University of New Brunswick, Fredericton, New Brunswick, January.

Trivedi, M., A. Bokil, M. Takla, G. Maksymonko, J.T. Broach, (1989). “Analysis of High Resolution Aerial Images for Object Detection.” *SPIE* Vol. 1099. pp 58-65.

Trivedi, M.M.(1987). “Object detection using their multispectral properties”, *Proceeding of the Society of Optical Engineering SPIE*, Vol. 754, pp. 255-264.

Vachon, P.W., J.W.M. Campbell, C. Bjerkelund, F.W. Dobson, and M.T. Rey (1996). "Validation of Ship Detection by the RADARSAT SAR", Proceedings of Pacific Ocean Remote Sensing Conference (PORSEC'96), 13-16 Aug.,Victoria, Canada.

Wallace, R., D. Affens, and S. McCandless (1998). “Search and Rescue from Space”. *SPIE Conference on Automatic Target Recognition VIII*, Orlando Florida, April. Vol.3371 pp174-184.

Zhang, Y. (2000). “A Method for Continuous Extraction of Multispectrally Classified Urban Rivers”. *Photogrammetric Engineering and Remote Sensing*. Vol. 66. No. 8. August. pp.991-999.

Zhang, Y. (2001). “Imaging and Mapping I – GGE3342 Lecture Notes.” Department of Geodesy and Geomatics Engineering, University fo New Brunswick, Fredericton, N.B., Canada

.

APPENDIX I

MRV RECON C CODE

```
/* ---------------------------------------------------------------------
 * - MRV Recon: ESST Plus - Enhanced Spatio Spectral Template  All rights reserved.
 * - By Kevin H. Pegler M.Eng.,P.Eng  -
 * - Dept. Geodesy and Geomatics Engineering, UNB
 * - GEOIDE Project #ENV 60
 * - Not to be used, reproduced or disclosed without permission.
 * - Esst.c Fall/Winter 2001
 * - Mahal.c Fall 2002/Winter 2003
 * - WED Summer 2003
 * - Characterization Fall 2003
 * - esstplus.c -> includes both esst and mahal(disabled)  and WED with some prompting
 * ---------------------------------------------------------------------
 * Usage: from ~/debug open an easi cursor from a DOS prompt window
 * easi> file = "D:\esst\ikonos\ikonostest.pix"
 * > run esstplus >debug.txt
 *
 *
/* ------------------------------------------------------------------- */
/*      Include "pci.h" in all C programs.                             */
/* ------------------------------------------------------------------- */
#include "pci.h"
#include <stdlib.h>
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

/* ------------------------------------------------------------------- */
/*      Function Protocols                                             */
/* ------------------------------------------------------------------- */
void   PixelFill (float *OutBuff2, int pixel, int half_kernel, int numofChans );
void   LineFill (float * OutBuff3, int half_kernel, int num_pixels, int numIkonosChan);
void   ImgAvg (float *line_ptr2[], float *OutBuff4, int num_pixels, int numIkonosChan,  int
half_kernel, int kernel_size, float *nirthrhld, float *panthrhld);
void   MaxDistFnct ( float *image_buffrMD,  float *out_Buff5, int num_pixels, int
numIkonosChan, int TwonumIkonosChan1, int  half_kernel, int kernel_size, int line_pos, int
exponent, float *nirthrhld, float *panthrhld, FILE *outfile);
void   MahalDistFnct ( float *image_buffrMD,  float *out_Buff5, int num_pixels, int
numIkonosChan, int TwonumIkonosChan1, int  half_kernel, int kernel_size, int
image_line_pos, int dbic_list[], int BigWindow_flag, FILE *outfile, float *nirthrhld, float
*panthrhld, FILE *fp);
void   MahalMaxFnct (float *image_buffrMahMax, int num_pixels, int num_lines,  int
half_kernel, int kernel_size, int dbic_list[]);


//declare some structures
typedef struct
{
      signed int pixel[1];
      signed int line[1];
} element;
```

```
typedef struct
{
        int top;
        int stack_size;
        element stack[3600];
} element_stack;
// define direction control
 struct individ_dir{
     int C[1];
     int R[1];
 };
// initialize an array of structures
struct individ_dir dir_list[8];
struct attribute {
     int smpixel[1];
     int smline[1];
     int cntrpixel[1];
     int cntrline[1];
     int lrgpixel[1];
     int lrgline[1];
     int numbrpixs[1];
     float length[1];
     float width[1];
     char orientation[5];
};
struct attribute the_target_attributes[500];
/*-------------------------------------------------------------------------- */
/*      Declare some Function Protocols for Matrix Manipulations   Dr.D. Kim  */
/*-------------------------------------------------------------------------- */
int      gjmatinv(double  a[], int  n);
void   mul_mm(double  *A, double  *B, double  *C, int  n1, int  n2, int  n3);
void   mul_mtm(double *A, double *B, double *C, int  n1, int  n2, int  n3);
double  calc_mean(double *A, int n);
double  calc_var(double *A, int n);
double  calc_cov(double *A, double *B, int n);
void   calc_vc(double *A, double *B, int n1, int n2);

/*------------------------------------------------------------------------- */
/*      Declare some Function Protocols for Characterization                */
/*-------------------------------------------------------------------------*/
void CharacterFnct (float *image_buffrCharactr, int num_pixels, int num_lines,int
half_kernel,int kernel_size, int dbic2_list[],float thrshold, FILE *targetsfile);
void init_stack (element_stack *the_stack);
void boatsizefnctn(element first_targetcoord, int heading[], FILE *targetsfile, int targetlabel);

/*################################################################################*/
/* -------------------------------------------------------------------- */
/*     main                                                            */
/*      ESST - Enhanced Spatio Spectral Template                        */
/*                                                                     */
/* ---------------------------------------------------------------------- */
/*###########################################################################*/
int main( int main_argc, char **main_argv )
{
/* ---------------------------------------------------------------------- */
/*     Declare parameters for IMPStatus.                           */
/* ---------------------------------------------------------------------- */
    char   file[257];
    char * gui_filename = NULL;
    //char tempname[50]="c:\\covartmp.pix";
    char    outfilename[50]="c:\\esst_out.txt";
    char   targetsfilename[50]="c:\\targets_out.txt";
    int      argcnt[1];
    void   *args[1];
/*----------------------------------------------------------------------- */
/*      Declare some other variables for ESST                              */
/*----------------------------------------------------------------------- */
    FILE  *fp,*outfile,*targetsfile;
    int      dbnc, num_lines, num_pixels, numIkonosChan,kernel_size = 0,line_pos,exponent,
buffsize,bmap, pid, status;
    int      i, half_kernel,junk,numReqChan,
TwonumIkonosChan1,numclosed,closeFlag,control_flag,BigWindow_flag;
    //int   pixel,channel;
    int      dbic_list[100],dbic2_list[2];
    int      dboc_list[5],dboc2_list[2],dbic_array[1],windoh[4];
    float   *image_buffr,*image_buffr2,*out_buffr,*image_buffrMahMax, *nirthrhld,
*panthrhld, *image_buffrCharactr;
    float   Corl[25],meen[5],sigma [5],covr[25],thrshold,timer;
    long    numpixssmpld;
    float  PercentComplete;
    /*tmpnam(tempname);*/




/*---------------------------------------------------------------------------- */
/*      Declare some Function Protocols for Matrix Manipulations   Dr.D. Kim  */
/*---------------------------------------------------------------------------- */
//int        gjmatinv(double  a[], int  n);
//void mul_mm(double  *A, double  *B, double  *C, int  n1, int  n2, int  n3);
//void mul_mtm(double *A, double *B, double *C, int  n1, int  n2, int  n3);

/* ---------------------------------------------------------------------- */
/*   Declare a large number of line pointers to allow for any reasonable  kernel size  */
/* --------------------------------------------------------------------- */
    float   *line_ptr[250];
    float   *lineOut_ptr[250];
/*----------------------------------------------------------------------- */
/*      Get the desired parameters
    //very important to have c:\MRV_Recon_GO.bat set up environment and pass parameters
from
```

```
    // C:\Documents and Settings\pegler\Desktop\MRV_Recon.exe
/*----------------------------------------------------------------- */
    gui_filename = main_argv[1];
    /*prompt for desired kernel size*/

    kernel_size = atoi(main_argv[2]);
    control_flag = atoi(main_argv[3]);

    /*for (i=0;i<main_argc;i++)
    {
        printf("Arg %d is %s\n",i,main_argv[i]);
    }*/

    /*puts(" Please enter desired kernel size");
    scanf("%d", &kernel_size);*/

    /* must get user to select type of distance metric they'd like*/

    /*puts ("Enter 1 for Minkowski distance metric or, Enter 2 for WED or Mahalanobis");
    scanf("%d", &control_flag);*/


    //kernel_size = main_argv[2];

/* ---------------------------------------------------------------- */
/*     Initialize argument list for IMPStatus.                       */
/* ---------------------------------------------------------------- */
    args[0] = (void *) file;
    //args[1] = (void *) &kernel_size;
/* ---------------------------------------------------------------- */
/*     Get parameters using IMPStatus.                               */
/* ---------------------------------------------------------------- */
        //printf("Just before IMPStatus\n");
    IMPStatus ( "FILE", "C", "64","1","esstplus.","ON",argcnt,args,main_argc, main_argv );
    //IMPStatus
("FILE,KERNEL_SIZE;","C,I;","64,2;","1,1;","esstplus.","ON",argcnt,args,main_argc,
main_argv );

    //printf("filename = %s\n",file);

    IMPPutChar("FILE", gui_filename);

    IMPStatus ("FILE","C","64","1","esstplus.","ON",argcnt,args,main_argc, main_argv );
    /*IMPStatus ("FILE,KERNEL_SIZE;",
            "C,         I;",
            "64,    2;",
            "1,     1;",
            "esstplus.","ON",argcnt,args,main_argc, main_argv );*/

    //printf("filename = %s\n",file);
/* ---------------------------------------------------------------- */
/*     Open a file                                                    */
/*  ------------------------------------------------------------------ */
     IDBRegister(); /* There is no documentation for this function just in examples*/
     fp = GDBOpen (file, "r+");

/* ------------------------------------------------------------------- */
/*    Run CLR to set up the file                                       */
/*  ------------------------------------------------------------------ */
     //printf("Into IMPRunTask\n");
     //status = IMPRunTask ("EASI", "r clr", RTF_WAIT, NULL,NULL);
     //printf("Out of IMPRunTask\n");

/* ------------------------------------------------------------------- */
/*     Query a file                                                    */
/*  ------------------------------------------------------------------ */
     num_lines = GDBChanYSize(fp);
     num_pixels = GDBChanXSize(fp);
     dbnc = GDBChanNum(fp);


/* ------------------------------------------------------------------- */
/*     Print Results to screen                                         */
/*  ------------------------------------------------------------------ */
     //printf("The number of lines are: %d \n", num_lines);
     //printf("The number of pixels are: %d \n", num_pixels);
     //printf("The number of Channels are: %d \n", dbnc);
/* ------------------------------------------------------------------- */
/*    Allocate Memory for the DBIC_List Array   fixed at 5 channels    */
/*  ------------------------------------------------------------------ */
     numIkonosChan = 5; /*changed to 2 just for testing*/
     TwonumIkonosChan1 = numIkonosChan * 2 + 1; // two times the original channels plus one
         /*numIkonosChan = 1; /* for testing*/

     /*dbic_list = (int *) malloc(sizeof(int) * numIkonosChan);*/

     for ( i = 0; i< numIkonosChan; i++ ){
          dbic_list[i] = i+1;  /* this "casts" the values into the array into image channel nums*/

     }
/* ------------------------------------------------------------------- */
/*    Allocate Memory for the DBOC_List Array                          */
/*  ------------------------------------------------------------------ */
         /*dboc_list = (int *) malloc(sizeof(int) * numIkonosChan);*/
     for ( i = 0 ; i <  numIkonosChan; i++ ){

          dboc_list[i] = i + numIkonosChan + 1;  /* this "casts" the values into the array into
image channel nums*/

         }
/* ------------------------------------------------------------------- */
/*    Allocate Memory for the floating buffer Arrays                   */
/*  --------------------------------------------------------------------- */
     image_buffr = (float *) malloc(sizeof(float) * num_pixels * numIkonosChan * kernel_size);

     out_buffr = (float *) malloc(sizeof(float) *(num_pixels * numIkonosChan ));

// allocate memory for NIR & PAN thresholds and initialize

     nirthrhld = (float *) malloc(sizeof(float) * 3);

     *(nirthrhld + 0) = 0.000;
     *(nirthrhld + 1) = 0.000;
     *(nirthrhld + 2) = 0.000;

     panthrhld = (float *) malloc(sizeof(float) * 3);

     *(panthrhld + 0) = 0.000;
     *(panthrhld + 1) = 0.000;
     *(panthrhld + 2) = 0.000;
/* ---------------------------------------------------------------------- */
/*    Pointers for the image buffer                              */
/*  --------------------------------------------------------------------- */
          for ( i = 0; i < kernel_size; i++)
               {
                         junk = num_pixels * numIkonosChan * i;
                         line_ptr[i] = image_buffr + (num_pixels * numIkonosChan * i);
                         lineOut_ptr[i] = out_buffr + (num_pixels * numIkonosChan * i);
               }   /* end of for loop for image buffer pointers */

/* ---------------------------------------------------------------------- */
/*    Loop over the imagecube                                     */
/*  --------------------------------------------------------------------- */
     //printf("Starting averaging!\n");
     for (line_pos = 0; line_pos < num_lines; line_pos++)
     {
     //printf("line # = %d \n", line_pos);

               if (line_pos%10 == 0)
                    //printf("Averaging %d  of %d lines\n", line_pos, num_lines);

/* ---------------------------------------------------------------------- */
/*    Read in the ImageData                                     */
/*  --------------------------------------------------------------------- */
          if (line_pos <=  (num_lines - kernel_size ) ){

               IDBRealChanIO (fp,IDB_READ, 0, line_pos, num_pixels, kernel_size,image_buffr,
                         num_pixels,kernel_size,numIkonosChan,dbic_list);
          }
          else {

          /* just set the image buffer pointers to last bit of good data */
          /* and read in the remaining original data lines */
```

```
        IDBRealChanIO (fp,IDB_READ, 0, line_pos, num_pixels, (num_lines -
line_pos),image_buffr,
                        num_pixels,(num_lines - line_pos),numIkonosChan,dbic_list);

            for ( i = (num_lines - line_pos + 1); i < ( kernel_size + 1); i++)
                {

                line_ptr[i-1] = image_buffr;

                }   /* end of for loop for image buffer pointers */

        } /* end of if else block*/

/* ---------------------------------------------------------------- */
/*    Test to see if it will be an unprocessed line                 */
/*  --------------------------------------------------------------- */

    half_kernel = floor(kernel_size / 2);

    if (line_pos < half_kernel || line_pos >= (num_lines - half_kernel)){

        /* then call LineFill function */

        LineFill (out_buffr, half_kernel,  num_pixels,  numIkonosChan);

        /* then write out the result of LineFill */

        IDBRealChanIO (fp,IDB_WRITE, 0, line_pos, num_pixels,
1,out_buffr,num_pixels,1,numIkonosChan,dboc_list);

    } /* end of unprocessed line operations*/

    else {
/* ---------------------------------------------------------------- */
/*     Calculate Average (Mean filter)                                  */
/*      Note: PixelFill is called from ImgAvg                           */
/*  --------------------------------------------------------------- */

        ImgAvg ( line_ptr, out_buffr, num_pixels, numIkonosChan, half_kernel, kernel_size,
nirthrhld, panthrhld);

        IDBRealChanIO (fp,IDB_WRITE, 0, line_pos, num_pixels,
1,out_buffr,num_pixels,1,numIkonosChan,dboc_list);
            }   /* end of else*/

    //Free up memory
    /*for ( i = 0; i < kernel_size; i++){
        free(line_ptr[i]);
        free(lineOut_ptr[i]);
        }   */
    } /* end of for loop over the image cube*/

    printf("Done averaging!\n");
/* ------------------------------------------------------------------ */
/*     Free up memory from the two buffers                              */
/*                                                         */
/*  ------------------------------------------------------------------ */
    free(image_buffr);
    free(out_buffr);
    image_buffr = NULL;
    out_buffr= NULL;
/* ------------------------------------------------------------------ */
/*     calulate average of NIR for a threshold                       */
/*                                                         */
/*  ------------------------------------------------------------------ */
    *(nirthrhld + 2) = *(nirthrhld)/(*(nirthrhld + 1)) * 1.65; // 1.7 for the River - slightly higher
for caddy bay
    *(panthrhld + 2) = *(panthrhld)/(*(nirthrhld + 1)) * 2.7;

    printf("PAN Thrhld - weighting is 1.8x = %f \n", *(panthrhld + 2));
    printf("NIR Thrhld - weighting is 1.4x = %f \n", *(nirthrhld + 2));
/*
################################################################################
#####################################*/
/* ------------------------------------------------------------------------ */
/*     Make Image Buffer larger to accomodate the original channels          */
/*     and the average channels & mah_buff to accomodate the covariance calcs */
/*  ------------------------------------------------------------------------*/
    if (kernel_size <= 11){
        buffsize = 11;
    }
    else {
        buffsize = kernel_size;
    }
    image_buffr2 = (float *) malloc(sizeof(float) * num_pixels * TwonumIkonosChan1  *
buffsize);
    out_buffr = (float *) malloc(sizeof(float) * num_pixels * TwonumIkonosChan1  * buffsize
); /*10 times larger just for extra room*/
    //covar_buffr = (float *) malloc(sizeof(float) * num_pixels * TwonumIkonosChan1  * 10 *
5); // covariance window maximum of 10 x 10

 //printf(" The number of bytes = %d\n", sizeof(float) * num_pixels * TwonumIkonosChan1  *
buffsize *2);


/* ------------------------------------------------------------------ */
/*     Beginning of Calculation of Outliers                          */
/*      Create a new DBIC list   Note: numIkonosChan still valid* 2 */
/*     includes the outlier channel                              */
/*  ---------------------------------------------------------------- */
        /*dbic_list = (int *) malloc(sizeof(int) * 2 * numIkonosChan);*/
    for ( i = 0 ; i <  TwonumIkonosChan1  ; i++ ){  /* ie. all original channels to averages calc'd
for originals*/

         dbic_list[i] = i+1;  /* this "casts" the values into the array into image channel nums*/
         /*printf("The new dbic_list[%d]=  %d\n", i, dbic_list[i]);
         /*printf("The new dbic_list=  %p\n", dbic_list);*/
    }

/* ----------------------------------------------------------------- */
/*     DBOC list                                                  */
/*  ---------------------------------------------------------------- */
    numReqChan = 1; /*one outlier channel required*/
    /*dbic_list = (int *) malloc(sizeof(int) * numReqChan);*/

    dboc_list[0] = 2 * numIkonosChan + 1;
/* ----------------------------------------------------------------- */
/*   Important! - Prior Calculating Outlier Frequency                  */
/*   You must ensure the histo channel is cleared and set to zero      */
/*  ---------------------------------------------------------------- */

/*------------------------------------------------------------------ */
/* Program Control                                                 */
/*------------------------------------------------------------------ */
if (control_flag == 1) {
    puts("You have Requested the Minkowski Distance Metric ");
    //printf("control_flag = %d\n",control_flag);
    puts("Enter the exponent value: 1 for Manhattan Distance or 2 for Euclidean");
    scanf("%d",&exponent);
    /*if (exponent != 1 || exponent != 2) {
        puts("You've enter an incorrect value, EXITING PROGRAM");
        exit(0);

    }*/
}

/*------------------------------------------------------------------*/
/* Set up file for formatted output for use in analysis              */
/*------------------------------------------------------------------*/

/* open the file*/
/* using fopen with the "w" option - if the file doesn't exist then it is created if exists it is
cleared out*/
outfile = fopen(outfilename, "w");
//IMPTermProgressCounter (-1.0,NULL,NULL);
/* ----------------------------------------------------------------- */
/*    Pointers for the image buffer                                */
/*  ---------------------------------------------------------------- */
        for ( i = 0; i < kernel_size; i++)
            {
```

```
            line_ptr[i] = image_buffr2 + (num_pixels * numIkonosChan * i);

         }   /* end of for loop for image buffer pointers */

            lineOut_ptr[0] = out_buffr;
/* ----------------------------------------------------------------- */
/*     Loop Through the Image cube                               */
/*  ----------------------------------------------------------------- */

   for (line_pos = 0; line_pos < num_lines; line_pos++){

   //percent complete counter

      //PercentComplete = (line_pos/num_lines) * 100;
      //IMPTermProgressCounter (PercentComplete,NULL,NULL);
      //printf("ESSTPLUS %f", PercentComplete);
      /* set a flag for closing temporary file at the end*/
      closeFlag = 0;
      BigWindow_flag = 0;
      if (line_pos%10 == 0)
         //printf("line_pos = %d  of %d lines\n", line_pos, num_lines);

//**********************************************************************
//       Start of complex program control
//**********************************************************************
      half_kernel = floor(kernel_size / 2);
      switch (control_flag) {
      // Load Buffers for Minkowski
      case 1: {
         /*     Read in the ImageData  -> Must read in the extra outlier channel  */


         if (line_pos <  (num_lines - kernel_size  ) ){

            //printf(" case == 1 First IDBRealChanIO \n");
            IDBRealChanIO (fp,IDB_READ, 0, line_pos, num_pixels,
kernel_size,image_buffr2,
                  num_pixels,kernel_size,TwonumIkonosChan1,dbic_list);
         MaxDistFnct ( image_buffr2, out_buffr, num_pixels, numIkonosChan,
TwonumIkonosChan1, half_kernel, kernel_size, line_pos, exponent, nirthrhld, panthrhld,
outfile);
               /*IDBRealChanIO (fp,IDB_WRITE, 0, line_pos, num_pixels,
kernel_size,image_buffr2,num_pixels,kernel_size,
         numReqChan,dboc_list);*/

         IDBRealChanIO (fp,IDB_WRITE, 0, line_pos, num_pixels,
kernel_size,image_buffr2,
                  num_pixels,kernel_size,TwonumIkonosChan1,dbic_list);

         continue;
```

```
            }

            else {


            /* and read in the remaining original data lines */
            /* then call LineFill function */

            LineFill (out_buffr, half_kernel,  num_pixels,  numReqChan);

            /* then write out the result of LineFill */

            IDBRealChanIO (fp,IDB_WRITE, 0, line_pos, num_pixels,
1,out_buffr,num_pixels,1,numReqChan,dboc_list);

            continue;

            } /* end of if else block*/

        } // end of case 1


        // Load Buffers for MAHALANOBIS


        case 2: {

                //printf("into control_flag == 2 \n");
                // 1) Unprocessed line operations

                if (line_pos < half_kernel || line_pos > (num_lines - 1 - kernel_size)){


                    /* then call LineFill function */

                    //LineFill (out_buffr, half_kernel,  num_pixels,  numReqChan);

                    /* then write out the result of LineFill */
                    //printf(" == 2 First IDBRealChanIO \n");
                    //IDBRealChanIO (fp,IDB_WRITE, 0, line_pos,
num_pixels,1,out_buffr,num_pixels,1,numReqChan,dboc_list);
                    //printf(" == 2 First IDBRealChanIO \n");

                    continue;

                } /* end of unprocessed line operations*/

                // 2) Special Larger CoVariance Window

                else if ( line_pos > 4 && line_pos < (num_lines - 5)){ //based again on a 11x11
                    BigWindow_flag = 1; // 1 means use a larger window for Covar

                    IDBRealChanIO (fp,IDB_READ, 0, (line_pos - 2), num_pixels,
5,image_buffr2,
                   num_pixels,5,TwonumIkonosChan1,dbic_list);

                    //printf(" == 2 Before GDBCreate \n");
                    //tmpfile = GDBCreate (FL_IDB, tempname, num_pixels, 11,
TwonumIkonosChan1, CHN_16U, "");
                    closeFlag = 1;
                    //printf(" == 2 Second IDBRealChanIO \n");
                    //IDBRealChanIO (fp,IDB_READ, 0, (line_pos-5), num_pixels, 11
,covar_buffr,   num_pixels,11,TwonumIkonosChan1,dbic_list);
                    //printf(" == 2 Third IDBRealChanIO \n");
                    //IDBRealChanIO
(tmpfile,IDB_WRITE,0,0,num_pixels,11,covar_buffr,num_pixels,11,TwonumIkonosChan1,dbi
c_list);



                    /*     Calculate Mahalanobis Distance - MahalDist                    */

                    MahalDistFnct (image_buffr2, out_buffr, num_pixels, numIkonosChan,
TwonumIkonosChan1, half_kernel, kernel_size, line_pos, dbic_list, BigWindow_flag, outfile,
nirthrhld, panthrhld, fp);

                    /* ---------------------------------------------------------------- */
                    /*     Write to file the Updated Histo channel for the entire kernel   */
                    /*  --------------------------------------------------------------- */


                    /* testing what is in image_buffr2*/
                    /* read in and print out one line of image_buffr2*/
                    //printf("Larger Window print buffr contents \n");
                    //for ( i = 0; i < (num_pixels * TwonumIkonosChan1 * kernel_size); i++){

                    //printf(" \n");
                    //printf("DN = %f\n",*(image_buffr2 + i));
                    //printf(" \n");

                    //}
                    //printf(" == 2 Fourth IDBRealChanIO \n");

                    IDBRealChanIO (fp,IDB_WRITE, 0, (line_pos - 2), num_pixels,5
,image_buffr2,num_pixels,5,TwonumIkonosChan1,dbic_list);

                    if (closeFlag == 1){

                   //printf(" closing tmpfile \n");
                   //GDBClose(tmpfile);
```

```
                          //remove(tempname);
                    }

                    BigWindow_flag = 0;
                    continue;
                  } //else if for Larger CoVar

              // 3) Regular sized CoVar window

                    /*    Read in the ImageData  -> Must read in the extra outlier channel  */
                    /*      note the extra lines of data are read in to accomodate ImgStats */
                    //printf(" == 2 Fifth IDBRealChanIO \n");

                    //IDBRealChanIO (fp,IDB_READ, 0, line_pos, num_pixels,
kernel_size,image_buffr2,num_pixels,kernel_size,TwonumIkonosChan1,dbic_list);


                    /*   Output the ImageData, for each line pos to a temp file - used in CoVar */

                    //tmpfile = GDBCreate (FL_IDB, tempname, num_pixels, kernel_size,
TwonumIkonosChan1, CHN_16U, "");
                    //closeFlag = 1;
                    //printf(" == 2 Six IDBRealChanIO \n");

                    //IDBRealChanIO
(tmpfile,IDB_WRITE,0,0,num_pixels,kernel_size,image_buffr2,num_pixels,kernel_size,Twon
umIkonosChan1,dbic_list);


                    //printf("Into MahalDistFnct \n");

                    //MahalDistFnct (image_buffr2, out_buffr, num_pixels, numIkonosChan,
TwonumIkonosChan1, half_kernel, kernel_size, line_pos, dbic_list, BigWindow_flag, tmpfile,
outfile, nirthrhld);

                    /* ------------------------------------------------------------------- */
                    /*     Write to file the Updated Histo channel for the entire kernel   */
                    /*  ---------------------------------------------------------------- */


                    /* testing what is in image_buffr2*/

                    //printf("Everything else print buffr contents \n");
                    //for ( i = 0; i < (num_pixels * TwonumIkonosChan1 * kernel_size); i++){

                    //printf(" \n");
                    //printf("DN = %f\n",*(image_buffr2 + i));
                    //printf(" \n");

                    //}
                    //printf(" == 2 Seventh IDBRealChanIO \n");
                    //this really screws things up.
                    //IDBRealChanIO (fp,IDB_WRITE, 0, line_pos,
num_pixels,1,image_buffr2,num_pixels,1,1,dboc_list);


                    //printf("closeFlag = %d \n", closeFlag);
                    //printf(" End of loading for Mahalanobis \n");
                    if (closeFlag == 1){
                        //printf(" closing tmpfile \n");
                        //GDBClose(tmpfile);
                        //remove(tempname);
                    }

                    BigWindow_flag = 0;

                } // end of loading for Case = 2

        }//end of switch


/* ----------------------------------------------------------------- */
/*     Bottom - Cleaning Up                                          */
/*  ---------------------------------------------------------------- */
    //Free up memory
        /*for ( i = 0; i < kernel_size; i++){

            free (line_ptr[i]) ;

            }

        free (lineOut_ptr[0]); */


    //printf("Bottom of loop over image cube \n");
    } /* end of for loop over the image cube*/


    //printf("Finished second loop...\n");


    free(image_buffr2);
    free(out_buffr);

    image_buffr2 = NULL;
    out_buffr = NULL;
    //free(covar_buffr);
```

```
//############################################################################
####################################


/* ------------------------------------------------------------------------ */
/*    If using Mahal dist then loop over the final channel and select outlier */
/*  ----------------------------------------------------------------------- */

// only do this step for Mahal processing

    if(control_flag == 2){ //22 is just an escape for testing

    printf("Enter final Mahal outlier selector \n");

        image_buffrMahMax = (float *) malloc(sizeof(float) * num_pixels * 3 * num_lines );

        dbic2_list[0] = 11; //TwonumIkonosChan1 - 1;//this contains the distance
        dbic2_list[1] = 12;  //TwonumIkonosChan1;
        dboc2_list[0]  = 11; //TwonumIkonosChan1 - 1;
        dboc2_list[1]  = 12; //TwonumIkonosChan1;
        //printf("dbic2_list[0] = %d dbic2_list[1] = %d dboc2_list[0] = %d \n",
dbic2_list[0],dbic2_list[1],dboc2_list[0]);

        // read in the entire image but only the last two channels


        //printf("into first read \n");
        IDBRealChanIO
(fp,IDB_READ,0,0,num_pixels,num_lines,image_buffrMahMax,num_pixels,num_lines,2,dbic
2_list);



        // Go out and select the max Mahalanobis Distance
        //printf("Into MahalMaxFnct\n");
        MahalMaxFnct (image_buffrMahMax, num_pixels, num_lines,  half_kernel, kernel_size,
dbic2_list);

        //printf ("Max threshold ratio > 0.8 \n");


            /* testing what is in image_buffr2*/
            /* read in and print out one line of image_buffr2*/

                /*for ( i = 0; i < 500; i++){

                    for ( pixel = 0; pixel  < num_pixels * num_lines * 2; pixel++){

                            printf(" \n");
```

```
                        printf(" pixel = %f\n", *(image_buffrMahMax + pixel));
                        printf(" \n");

                }

            }

            printf("exiting early for testing purposes \n");
            exit(1);*/

        // write out the second last channel to the last channel

        //printf("Writing out  \n");
        IDBRealChanIO (fp,IDB_WRITE, 0, 0, num_pixels,
num_lines,image_buffrMahMax,num_pixels,num_lines,2,dboc2_list);
        //printf("num_pixels = %d num_lines = %d \n", num_pixels, num_lines);
        //IDBRealChanIO (fp,IDB_WRITE, 0, 0,
num_pixels,num_lines,image_buffrMahMax,num_pixels,num_lines,5,dbic_list);

    free(image_buffrMahMax);
    image_buffrMahMax = NULL;

    }// end of control flag for mahal freq


    //Free up some memory

free(nirthrhld);
nirthrhld = NULL;

//*******************************************************************


// TARGET CHARACTERIZATION


//*******************************************************************

//printf("Gonna do target characeterization\n");

image_buffrCharactr = (float *) malloc(sizeof(float) * num_pixels * 2  * num_lines * 5);

//read in the imagery to the buffr
//only the pan and freq channel


dbic2_list[0] = 1; //Pan channel

if(control_flag == 1)//Minkowski
dbic2_list[1] = 11;//Freq channel
```

```
if(control_flag == 2)//Mahal
dbic2_list[1] = 12;//Freq channel

dbic_array[0] = 1; //Pan
bmap = 0;
windoh[0] = 0;
windoh[1] = 0;
windoh[2] = num_pixels;
windoh[3] = num_lines;

/* open the targets_out file*/
/* using fopen with the "w" option - if the file doesn't exist then it is created if exists it is
cleared out*/


targetsfile = fopen(targetsfilename, "w");

// print the column headings

fprintf(targetsfile,"Target Number  Pixel   Line      Length         Width        Orientation \n");


IDBRealChanIO
(fp,IDB_READ,0,0,num_pixels,num_lines,image_buffrCharactr,num_pixels,num_lines,2,dbic2
_list);

//require the mean and std deviation for pan channel for threshold

numpixssmpld = ImageStats(fp, 1, dbic_array, bmap, windoh, covr, Corl, meen, sigma);

thrshold = meen[0] + (4 * sigma [0]); //using one stnd deviation to tighten things up a bit
printf("End of boat threshold = %f\n",thrshold);

//printf("Frequency threshold = >8 \n");
//printf("threshold = %f\n",thrshold);

CharacterFnct (image_buffrCharactr, num_pixels, num_lines, half_kernel, kernel_size,
dbic2_list, thrshold, targetsfile);




printf("final write \n");

if(control_flag == 1)//Minkowski
dbic2_list[1] = 12;//Freq channel

IDBRealChanIO (fp,IDB_WRITE, 0, 0, num_pixels,
num_lines,image_buffrCharactr,num_pixels,num_lines,2,dbic2_list);
```

```
/* ------------------------------------------------------------------ */
/*    SPAWN the HANDLER                                                */
/*  ------------------------------------------------------------------ */

//IMPRunTask("handler",NULL,0, NULL,&pid);
/* ------------------------------------------------------------------ */
/*     Close a file                                         */
/*  ------------------------------------------------------------------ */

   /* All files are closed: */
   /*numclosed = _fcloseall( );*/
     fclose (outfile);
     fclose (targetsfile);
   numclosed = fclose(fp);
  timer = (float)(clock() / 1000);
     printf("Program Execution Time = %f minutes\n",timer/60);
/* ------------------------------------------------------------------ */
/*     Exit program using IMPReturn.                              */
/* ------------------------------------------------------------------ */
printf("\n\nControl C to EXIT ....");
while (1) {

}
exit(1); //IMPReturn(); //IMPReturn always exits with an error msg.
} /*End of Main*/








/* ------------------------------------------------------------------ */
/*    ALL FUNCTIONS LOCATED BELOW                                       */
/* ------------------------------------------------------------------ */

/*##########################################################################*/
/* ------------------------------------------------------------------ */
/*     PixelFill                                                  */
/*      Function to put a Zero value in unprocessed pixels               */
/*      found at the edge of images                              */
/* ------------------------------------------------------------------ */
/*##########################################################################*/



void   PixelFill (float *OutBuff2, int pixel, int half_kernel, int numofChans )
{

int i;
```

```
for ( i = 0; i < numofChans; i++)
    {
    /*printf("i = %d \n", i);*/


    *(OutBuff2 + pixel * numofChans + i) = 0;
    /*printf ("Pixel Fill value = %f\n", *(OutBuff2 + pixel * numofChans + i));*/

    }

}


/*###########################################################################*/
/* ------------------------------------------------------------------ */
/*     LineFill                                                        */
/*      Function to put a Zero value in unprocessed lines                  */
/*      found at the edge of images                                  */
/* ------------------------------------------------------------------ */
/*###########################################################################*/


 void  LineFill (float * OutBuff3, int half_kernel, int num_pixels, int numChan)
 {

int pixel;

     /* fill a line up, pixel by pixel by number of channels, with zeros */
     for  (pixel = 0; pixel < ( ((num_pixels - 1) * numChan) + 1); pixel++){



        *(OutBuff3 + pixel) = 0;
     }

 }



/*###########################################################################*/
/* ------------------------------------------------------------------ */
/*     ImgAvg                                                          */
/*      Function to perform an mean filter                                 */
/*      Called for each line of image                                 */
/* ------------------------------------------------------------------ */
/*###########################################################################*/


void   ImgAvg (float *line_ptr2[], float *OutBuff4, int num_pixels, int numIkonosChan,  int
half_kernel, int kernel_size, float *nirthrhld, float *panthrhld)
```

```
{

    int j, k, channel, pixel;

    float sum;
    /*printf("Into ImgAvg: \n");*/
/* ------------------------------------------------------------------ */
/*  Average calc'd for a single line                              */
/* ------------------------------------------------------------------ */



    for ( pixel = 0; pixel < num_pixels; pixel++ ){

        //printf("pixel = %d \n", pixel);



            if (pixel < half_kernel || pixel >= (num_pixels - half_kernel)) {

                 PixelFill(OutBuff4, pixel, half_kernel,  numIkonosChan );
            }
            else {

                //**************************************************
                //calc running sum for NIR and PAN threshold
                //only calc for pixels not set to background value of 0
                //**************************************************

                if (  *(line_ptr2[0] + (pixel * numIkonosChan) + 4) != 0  &&  *(line_ptr2[0] +
(pixel * numIkonosChan) + 1) != 0){
                    *(nirthrhld) = *(nirthrhld) + (*(line_ptr2[0] + (pixel * numIkonosChan) + 4));
//sum up the NIR channel
                    *(panthrhld) = *(panthrhld) + (*(line_ptr2[0] + (pixel * numIkonosChan)));
                    *(nirthrhld + 1) = ( *(nirthrhld + 1) + 1); //add one to the total number of pixels
                    //printf(" running counter = %f \n", *(nirthrhld + 1));
                }
                //**************************************************

                for(channel = 0; channel < numIkonosChan; channel++){

                    sum = 0;

                    for (j = 0; j < ((2 * half_kernel) + 1 ); j++){  /* line in kernel*/

                        for ( k = (pixel - half_kernel); k < ((pixel + half_kernel)+ 1); k++){ /* pixel
in kernel*/
                        /*printf("Channel = : %d ", channel);
                        printf("J= : %d ", j);
                        printf("K= : %d ", k);
                        junk = (2 * half_kernel) + 1;
                        printf("*(line_ptr2[j] + k * numIkonosChan + channel) = %f ",
*(line_ptr2[j] + k * numIkonosChan + channel));
                        printf("line_ptr2[%d]=: %f", j,*line_ptr2[j]);*/
                        sum = sum +  *(line_ptr2[j] + k * numIkonosChan + channel);

                        /*printf(" Sum = : %f \n", sum);*/
                        }/* end of k loop*/

                    }/* end of j loop*/

                    *(OutBuff4 + pixel * numIkonosChan + channel) = sum / ( pow(kernel_size,
2));
                    /*printf(" Avg= : %f \n", *(OutBuff4 + pixel * numIkonosChan + channel));*/
                }/*end of channel loop*/

            } /*end of the else statement*/
        } /* end of pixel for loop*/

} /* bottom of image average*/

/*###########################################################################*/
/* ------------------------------------------------------------------- */
/*     MaxDist                                                         */
/*      Function to calculate max dist - outliers                      */
/*      Called for each line                                   */
/* ------------------------------------------------------------------- */
/*###########################################################################*/



    void MaxDistFnct ( float *image_buffrMD,  float *out_Buff5, int num_pixels, int
numIkonosChan, int TwonumIkonosChan1, int  half_kernel, int kernel_size, int
image_line_pos, int exponent, float *nirthrhld, float *panthrhld, FILE *outfile){

        int pixel, channel, j,k, MaxLine, MaxPixel;

        int numlinesdown, numpixintoline,posinkernel;

        float dist, maxdist;

        //printf("Into MaxDist \n");
        //printf("num_pixels = %d half_kernel = %d kernel_size = %d \n",
num_pixels,half_kernel,kernel_size);


        for ( pixel = 0; pixel < num_pixels ; pixel++){

                    if (pixel < half_kernel || pixel > (num_pixels - kernel_size)) {

                     PixelFill(out_Buff5, pixel, half_kernel, TwonumIkonosChan1 );
            }
            else {

                    maxdist = 0;
                    /*printf("maxdist = %d \n", maxdist);*/
                    for (j = 0; j < kernel_size; j++){  /* line in kernel*/
                        /*printf("line in kernel: j  = %d \n", j);*/
                        for ( k = (pixel); k < ((pixel + kernel_size)); k++){ /* pixel in kernel*/
                            /*printf("pixel in kernel: k  = %d \n", k);*/
                            dist = 0;

                            numlinesdown = num_pixels * TwonumIkonosChan1 * j;
                            numpixintoline = pixel * TwonumIkonosChan1;
                            posinkernel = (k - pixel) * TwonumIkonosChan1;

                            //printf ("TwonumIkonosChan1 = %d numIkonosChan = %d\n",
TwonumIkonosChan1,numIkonosChan);
                            //printf( "Numlinesdown = %d numpixintoline = %d posinkernel =
%d\n", numlinesdown,numpixintoline,posinkernel);*/

                            for(channel = 0; channel < numIkonosChan  ; channel++){


                                /*printf("channel = %d \n", channel);*/

                                /*printf("*(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + channel)= %f \n",
                                *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + channel));
                                printf("*(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + channel + numIkonosChan)= %f \n",
                                *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + channel + numIkonosChan));*/
                                dist =  dist + pow( fabs( *(image_buffrMD + numlinesdown +
numpixintoline + posinkernel + channel) -
                                      *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + channel + numIkonosChan)), exponent);
                                /*printf("dist before root:%f \n", dist);*/
                            }/*end of channel loop*/

                            /* Don't take the root for Manhattan distances*/

                            if (exponent != 1) {
                                dist = sqrt(dist);
                            }
                            /* ------------------------------------------------------------------- */
                            /*     Formatted Text Output                                           */
                            /*      for specific pixels of interest                                */
                            /*      note: this is hardwired for simplicity                       */
                            /*      just using multiple IF statements to control - quick & dirty    */
                            /* ------------------------------------------------------------------- */
```

```
                            /*if (image_line_pos >= (107 - half_kernel) && image_line_pos <= (107
+ half_kernel)){
                                if (pixel >= (100 - half_kernel) && pixel <= (100 + half_kernel)){
                                       /*note the compensation for counting from 0 to actual pixel &
line values*/
                                       /*fprintf(outfile,"%d,%d,%f,%f,%f,%f,%f,%f \n",(pixel +
1),(image_line_pos + 1),                               *(image_buffrMD + numlinesdown +
numpixintoline + posinkernel + 0),
                                               *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + 1),
                                               *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + 2),
                                               *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + 3),
                                               *(image_buffrMD + numlinesdown + numpixintoline +
posinkernel + 4),
                                               dist);

                                }
                            }


                            /*printf("dist = %f \n", dist);*/
                            if (dist > maxdist) {

                                 maxdist = dist;
                                 /*printf("max dist @ line %d, pixel %d = %f \n", j,k,maxdist);*/
                                /* store the "coordinates" of the new MaxDist"*/

                                MaxLine = j;
                                MaxPixel = k;


                            } /*end of if statement*/

                        } /* end of k "pixel in kernel" loop*/

                    } /* end of j "line" loop*/
                     /* Increment the exisiting histogram value for the outlier by 1*/
                    /* recalc image buffer "coords" based on MaxLine and MaxPixel*/

                    numlinesdown = num_pixels * TwonumIkonosChan1 * MaxLine;
                    /*numpixintoline = pixel * TwonumIkonosChan1; Doesn't change from
above*/
                    posinkernel = (MaxPixel - pixel) * TwonumIkonosChan1;
                    /*
                    printf( "Numlinesdown = %d numpixintoline = %d posinkernel = %d\n",
numlinesdown,numpixintoline,posinkernel);

                     printf(" \n");
                    printf(" \n");
                    printf("*(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1)= %f \n",
                        *(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1));
                    printf("*(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 -1 ) + 1 = %f \n",
                        (*(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1) + 1));*/

                    /*to improve performance, only select an outlier if maximum distance is
beyond a threshold*/

                    if (maxdist > (*(nirthrhld + 2)) && maxdist > (*(panthrhld + 2)) ){

                        /*printf("Image Line Number = %d\n", image_line_pos);
                        printf("PIXEL IN IMAGE = %d \n", pixel);
                        printf("*(image_buffrMD + numlinesdown + numpixintoline + posinkernel
+ TwonumIkonosChan1 -1 ) + 1 = %f \n",
                        (*(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1) + 1));*/

                            *(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1) =
                            ( *(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1) + 1);


                    }


                    /*printf(" After buffr incremented:  *(image_buffrMD + numlinesdown +
numpixintoline + posinkernel + TwonumIkonosChan1 - 1)= %f \n",
                        *(image_buffrMD + numlinesdown + numpixintoline + posinkernel +
TwonumIkonosChan1 - 1));
                    printf(" Image Line Position: %d \n", image_line_pos);
                    printf(" \n");*/



            } /*end of the else statement*/



        } /*end of pixel loop*/

        //printf("End of MaxDist \n");
    }/*End of MaxDist function*/
```

```
/*######################################################################*/
/* ---------------------------------------------------------------- */
/*     MahalDist                                                    */
/*      Function to calculate mahalanobis dist - outliers              */
/*      Called for each line                                   */
/* ---------------------------------------------------------------- */
/*######################################################################*/


   void MahalDistFnct ( float *image_buffrMD,  float *out_Buff5, int num_pixels, int
numIkonosChan, int TwonumIkonosChan1, int  half_kernel, int kernel_size, int
image_line_pos, int dbic_list[], int BigWindow_flag,  FILE *outfile, float *nirthrhld, float
*panthrhld, FILE *fp){

        int cell, channel,ll,i,j,k,cntr,n1,bitmap;
        //int singular;
        int numlinesdown, numpixintoline, posinkernel, buffoffset;

        long pixs_sampled;

        double dist;

        float XminusM[5],Corr[25], means[5], std_dev[5];

        float CoVar[25];

        int window[4];

        double XminusMDbl[5],IntoCoVar[605],CoVarDbl[25],X_MCovar[5],MahalSqrd[1];

        bitmap = 0;//set to zero to sample the entire channel
        n1 = 1;
        //printf("Into MahalDist \n");
        //printf("BigWindow_flag = %d \n", BigWindow_flag);

        //printf("num_pixels = %d half_kernel = %d kernel_size = %d \n",
num_pixels,half_kernel,kernel_size);

        numlinesdown = 0;
        posinkernel = 0;
        numpixintoline = 0;
        buffoffset = 2 * num_pixels * TwonumIkonosChan1;

        window[0] = 0;
        window[1] = 0;
        window[2] = 0;
        window[3] = 0;

        for ( cell = 0; cell < num_pixels ; cell++){
            cntr = 0;
            //printf("cell IN IMAGE (cell) = %d \n", cell);

            //clear the array
            for ( i = 0; i < 605; i++){

                IntoCoVar[i] = 0;

            }
            // for testing content
            numpixintoline = cell * TwonumIkonosChan1;
            /*  if (image_line_pos ==20 && (cell > 29 || cell < 37)){

                    printf("test output\n");
                    printf("%d, %d, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f\n"
                    ,(cell + 1),(image_line_pos + 1),*(image_buffrMD + numpixintoline  + 0),
                    *(image_buffrMD  + numpixintoline  + 1),*(image_buffrMD  +
numpixintoline + 2),
                    *(image_buffrMD  + numpixintoline  + 3),*(image_buffrMD  +
numpixintoline  + 4),
                    *(image_buffrMD + numpixintoline  + 5),*(image_buffrMD  + numpixintoline
+ 6),
                    *(image_buffrMD  + numpixintoline + 7),*(image_buffrMD  + numpixintoline
+ 8),
                    *(image_buffrMD  + numpixintoline  + 9));

                }*/


            if (cell < 5  || (cell > (num_pixels - 11 ))) {

                PixelFill(out_Buff5, cell, half_kernel, TwonumIkonosChan1 );
                continue;
            }

            else {

                        //printf ("TwonumIkonosChan1 = %d numIkonosChan = %d\n",
TwonumIkonosChan1,numIkonosChan);
                        //printf( "Numlinesdown = %d numpixintoline = %d posinkernel = %d\n",
numlinesdown,numpixintoline,posinkernel);

                        dist = 0;

                        //printf("BigWindow_flag = %d \n", BigWindow_flag);

                        // Window size = kernel size BigWindow_flag == 1 // 1 means possibly use
a larger window for Covar
                        if (BigWindow_flag == 1 && (cell > 4 && cell < ( num_pixels - 12))){
//upsize that window{

                                //printf(" Setting a big window \n");
                                window[0] = cell;

                                window[1] = (image_line_pos - 5);

                                window[2] = 11;

                                window[3] = 11;

                                //printf("window[0]=%d\n",window[0]);
                                //printf("window[1]=%d\n",window[1]);
                                //printf("window[3]=%d\n",window[3]);
                                //printf("window[2]=%d\n",window[2]);
                                //printf("image_line_pos = %d \n",image_line_pos);
                                //printf("cell= %d \n",cell);

                                // Populate IntoCoVar to calculate Covariance
                                int wsize = 5;
                                //printf("into CoVar \n");
                                for ( channel = 0; channel < numIkonosChan; channel++){
                                  //printf("channel = %d \n",channel);
                                  for ( j = 0; j < wsize; j++){ //whole lines down
                                      //printf("j or whole lines down = %d \n", j);
                                    for ( k = 0; k < wsize; k++){ //pixels in
                                        //printf("pixels into line = %d \n", k);
                                      IntoCoVar[cntr] = *(image_buffrMD  + (j *
TwonumIkonosChan1 * num_pixels) + ( (cell + k - 2) * TwonumIkonosChan1 ) + channel);
                                      cntr++;
                                    } //end of pixel loop in 11 x 11 covar kernel


                                  }// end of line loop in 11 x 11 covar kernel


                                }//end of IntoCoVar Channel loop


                        } //end of BigWindow

                        else {  // regular size covar window
                        //printf(" setting a regular sized window\n");
                          window[0] = cell;

                          window[1] = 0;

                          window[2] = kernel_size;

                          window[3] = kernel_size;

                          //printf("window[0]=%d\n",window[0]);
                          //printf("window[1]=%d\n",window[1]);
                          //printf("window[3]=%d\n",window[3]);
```

```
                    //printf("window[2]=%d\n",window[2]);
                    //printf("image_line_pos = %d \n",image_line_pos);
                    //printf("cell= %d \n",cell);

                } //end of BigWindow else



                //printf("calling ImageStats \n");
                //pixs_sampled = ImageStats(fp, numIkonosChan, dbic_list, bitmap,
window, CoVar, Corr, means, std_dev);

                calc_vc(IntoCoVar, CoVarDbl, 5, numIkonosChan);

                /*printf("out of ImageStats \n");

                printf("cell= %d \n",cell);

                printf("kernel size = %d \n", kernel_size);

                /* change each element of CoVar to a double*/

                for (i = 0; i < 25; i++){

                    CoVarDbl[i] = CoVarDbl[i]/100000;
                    //CoVarDbl[i] = 1;
                }
                    // weight the pan channel heavily and NIR? for small target
discrimination
                    //if (i == 1)

                CoVarDbl[0] = CoVarDbl[0] * 3; //pan
                CoVarDbl[24] = CoVarDbl[24] * 2;//NIR

                if (cell == 0 && image_line_pos == 1)

                    printf("Weighting the Pan channel by 100 and NIR by 5 \n");
                //}

                // Identity matrix for testing Mahal distance - sum of squares

                //for (i = 0; i < 5; i++){

                //   for (j = 0; j < 5; j++){

                //       if (i == j)
                //           CoVarDbl[i*5+j] = 1;
                //       else
                //           CoVarDbl[i*5+j] = 0;

                        //printf("CoVarDdbl[%d]= %f\n", i, CoVarDbl[i]);
```

```
            //   }

            //}

            /* invert the Covariance Matrix*/


            /*printf("cell= %d \n",cell);
            printf("kernel size = %d \n", kernel_size);
            printf("going to mat inv \n");*/
            //testing to see if not inverting the Covar matrix helps
            //singular = gjmatinv(CoVarDbl,numIkonosChan);

            //printf("out of mat inv \n");
            //printf("singular = %d \n", singular);

            /* zero the XM matrix*/
            for ( ll = 0; ll < numIkonosChan; ll++){

                XminusM[ll] = 0;

            }


            //printf("kernel_size = %d \n",kernel_size);
            //printf("cell= %d \n",cell);

            numpixintoline = cell * TwonumIkonosChan1;

            //printf ("TwonumIkonosChan1 = %d numIkonosChan = %d\n",
TwonumIkonosChan1,numIkonosChan);
            //printf( "numpixintoline = %d \n", numpixintoline);

            for(channel = 0; channel < numIkonosChan  ; channel++){

                //printf("channel = %d \n", channel);

                /*printf("*(image_buffrMD  + numpixintoline  + channel)= %f \n",
                    *(image_buffrMD  + numpixintoline  + channel));
                printf("*(image_buffrMD  + numpixintoline  + channel +
numIkonosChan)= %f \n",
                    *(image_buffrMD  + numpixintoline  + channel +
numIkonosChan));*/

                XminusM[channel] =  (*(image_buffrMD + buffoffset + numpixintoline
+ channel) -
                          *(image_buffrMD + buffoffset + numpixintoline  + channel +
numIkonosChan));

                //printf("XminusM[%d] = %f \n", channel, XminusM[channel]);
            }/*end of channel loop*/

            /*>>>>>>>>>>>>>>>>>>>>>>>>>>>>*/
            /* Matrix manipulations here */
            /*>>>>>>>>>>>>>>>>>>>>>>>>>>>>*/

            /* Cast the arrays over to double from float*/

            for (i = 0; i < numIkonosChan; i++){


                XminusMDbl[i] = XminusM[i];


            }


            //printf("into first mul_mtm\n");

  mul_mtm(XminusMDbl,CoVarDbl,X_MCovar,n1,numIkonosChan,numIkonosChan);
            //printf("out of first mul_mtm\n");

            //printf("into 2nd mul_mm\n");
            mul_mm(X_MCovar, XminusMDbl, MahalSqrd, n1,numIkonosChan,n1);
            //printf("out of 2nd mul_mm\n");

            //printf("MahalSqrd[0] = %f\n",MahalSqrd[0]);

            dist = sqrt(fabs(MahalSqrd[0])/10000); //just for testing getting rid of neg
values
            //dist = sqrt(MahalSqrd[0]);



            /* --------------------------------------------------------------- */
            /*     Formatted Text Output                                        */
            /*      for specific pixels of interest                             */
            /*      note: this is hardwired for simplicity                   */
            /*      multiple IF statements for control - quick & dirty               */
            /* --------------------------------------------------------------- */
            if (image_line_pos >= (1374 - half_kernel) && image_line_pos <= (1374 +
half_kernel)){
                if (cell >= (1275 - half_kernel) && cell <= (1275 + half_kernel)){
                //note the compensation for counting from 0 to actual pixel & line
values
                    printf("Writing out CoVar\n");
                    fprintf(outfile,"%d,%d,%f,%f,%f,%f,%f,%f\n\n"
                    ,(cell + 1),(image_line_pos + 1),*(image_buffrMD +
numpixintoline  + buffoffset + 0),
                    *(image_buffrMD  + numpixintoline + buffoffset +
1),*(image_buffrMD  + numpixintoline + buffoffset + 2),
                                    *(image_buffrMD  + numpixintoline + buffoffset +
3),*(image_buffrMD  + numpixintoline + buffoffset + 4),
                                        dist);
                                    fprintf(outfile,"%f,%f,%f,%f,%f\n",
                                            CoVarDbl[0],
                                            CoVarDbl[1],
                                            CoVarDbl[2],
                                            CoVarDbl[3],
                                            CoVarDbl[4]);
                                    fprintf(outfile,"%f,%f,%f,%f,%f\n",
                                            CoVarDbl[5],
                                            CoVarDbl[6],
                                            CoVarDbl[7],
                                            CoVarDbl[8],
                                            CoVarDbl[9]);
                                    fprintf(outfile,"%f,%f,%f,%f,%f\n",
                                            CoVarDbl[10],
                                            CoVarDbl[11],
                                            CoVarDbl[12],
                                            CoVarDbl[13],
                                            CoVarDbl[14]);
                                    fprintf(outfile,"%f,%f,%f,%f,%f\n",
                                            CoVarDbl[15],
                                            CoVarDbl[16],
                                            CoVarDbl[17],
                                            CoVarDbl[18],
                                            CoVarDbl[19]);
                                    fprintf(outfile,"%f,%f,%f,%f,%f\n\n",
                                            CoVarDbl[20],
                                            CoVarDbl[21],
                                            CoVarDbl[22],
                                            CoVarDbl[23],
                                            CoVarDbl[24]);
                            }
                        }


                        //----------------------------------------------
                        // Put the Mahalanobis distance back into the file
                        //----------------------------------------------

                        /*printf("dist = %f \n", dist);*/


                        //} /*end of if statement*/


                //------------------------------------------------------------
                // Put the Mahalanobis Distance in its place in an empty channel
                //------------------------------------------------------------
                    //printf("dist = %f \n", dist);
                    // Probable location of future EXPERT SYSTEM
                        //printf("nirthrhld + 2 = %f \n", *(nirthrhld * 2));
                    if( *(image_buffrMD + buffoffset + numpixintoline  + 4) > (*(nirthrhld + 2))
&& (*(image_buffrMD + buffoffset + numpixintoline) > (*(panthrhld + 2))))
                        *(image_buffrMD + buffoffset + numpixintoline  + TwonumIkonosChan1 -
1 ) = dist;// minus 1 gets into the last channel.
                    //printf("*(image_buffrMD  + numpixintoline  + TwonumIkonosChan1 - 1) =
%f \n", *(image_buffrMD  + numpixintoline  + TwonumIkonosChan1 - 1));

                }//end of else that includes all mahal calculations
            //printf("bottom of cell loop, next cell\n");
        } /*end of cell loop*/

        //printf("End of MahalDist \n");
    }/*End of MahalDist function*/


//
===========================================================================
=========
//  C_math.cpp
//
//  HISTORY:  Original -  3 AUGUST  2001
//            Modified - 11 October 2001
//
//  PURPOSE:  Common Math Functions.
//
// ---------------------------------------------------------------------------
//  All rights reserved.  --  Dr. Donghyun Kim  [February 16, 2002]
//  GPS Research Laboratory, University of New Brunswick, Canada
//
===========================================================================
=========


// ---------------------------------------------------------------------------
//  Gauss-Jordan matrix inverse
//  - a[]: an input square matrix to be overwritten
//  - n:   matrix dimension
//  - det: determinent
//  Return value:  =0:  non-singular (but will be cross-checked later.)
//                =-1: singular
// ---------------------------------------------------------------------------
int  gjmatinv(double  *a, int  n){

    int  ii, jj, kk;
    double  det, t, xx;
    /*printf("Into Mat Inverse \n");*/

    det = 1;
    for (kk=0; kk<n; kk++)  {
        t = a[n*kk+kk];

        // Singularity check
        if (t == 0.0)
            return(-1);

        det *= t;
        for (ii=0; ii<n; ii++)
            a[n*kk+ii] /= t;

        a[n*kk+kk] = 1/t;
        for (jj=0; jj<n; jj++)
            if (jj != kk)  {
                xx = a[n*jj+kk];
                for (ii=0; ii<n; ii++)
                    if (ii != kk)  a[n*jj+ii] -= a[n*kk+ii]*xx;
                    else  a[n*jj+ii] = -xx/t;
            }
    }

    return(0);
}



// ---------------------------------------------------------------------------
//  Matrix multiplication (mul_mm()) function definition
// ---------------------------------------------------------------------------
void  mul_mm(double  *A, double  *B, double  *C, int  n1, int  n2, int  n3){

    int  m, j, k;
    double  sum;
    /*printf("Into mul_mm function \n");*/
    //  Dimension : (n1,n2)*(n2,n3)
    //  Calc. matrix : A*B = C
    for (m=0; m<n1; m++){
        /*printf("m= %d of n1 = %d \n", m,n1);*/
        for (j=0; j<n3; j++)  {
            /*printf("j= %d of n3 = %d \n", j,n3);*/
            sum = 0;
            for (k=0; k<n2; k++){
                /*printf("k= %d of n2 = %d \n", k,n2);*/
                sum += A[m*n2+k]*B[k*n3+j];
                /*printf("sum = %f\n",sum);*/
            }
            /*printf("c[%d*%d+%d] = c[%d] = MahalSqrd[0] = %f\n",m,n3,j,(m*n3+j),sum);*/
            C[m*n3+j] = sum;

        }
    }
        /*printf("Out of mul_mm function \n");*/
```

```
}



// ---------------------------------------------------------------------------
//  Matrix multiplication (mul_mtm()) function definition
// ---------------------------------------------------------------------------
void  mul_mtm(double *A, double *B, double *C, int  n1, int  n2, int  n3)  {

    int  m, j, k;
    double  sum;
    /*printf("Into mul_mtm function\n");*/
    //  Dimension : (n2,n1)t*(n2,n3)
    //  Calc. matrix : At*B = C
    for (m=0; m<n1; m++)
        for (j=0; j<n3; j++)  {
            sum = 0;
            for (k=0; k<n2; k++)
                sum += A[k*n1+m]*B[k*n3+j];
            C[m*n3+j] = sum;
            /*printf("C[%d] = %f \n",(m*n3+j),sum);*/
        }
        /*printf("Out of mul_mtm function \n");*/
}


// ---------------------------------------------------------------------------
//  Mean calculation
// ---------------------------------------------------------------------------
double  calc_mean(double *A, int n)  {
    int  i;
    double  sum, mean;

    sum = 0;
    for (i=0; i<n; i++)
        sum += A[i];

    if (n < 2)  mean = 1.0e100;     // set an infinite value
    else         mean = sum/n;

    return(mean);
}


// ---------------------------------------------------------------------------
//  Variance calculation
// ---------------------------------------------------------------------------
double  calc_var(double *A, int n)  {
    int  i;
    double  var, mean, sum;
    mean = calc_mean(A,n);
    sum = 0;
    for (i=0; i<n; i++)
        sum += pow(A[i]-mean,2.0);

    if (n < 2)  var = 1.0e100;     // set an infinite value
    else         var = sum/n;

    return(var);
}


// ----------------------------------------------------------------------------
//  Covariance calculation
// ----------------------------------------------------------------------------
double  calc_cov(double *A, double *B, int n)  {
    int  i;
    double  cov, mean1, mean2, sum;

    mean1 = calc_mean(A,n);
    mean2 = calc_mean(B,n);

    sum = 0;
    for (i=0; i<n; i++)
        sum += (A[i]-mean1)*(B[i]-mean2);

    if (n < 2)  cov = 1.0e100;     // set an infinite value
    else         cov = sum/n;

    return(cov);
}


// ----------------------------------------------------------------------------
//  Variance-covariance matrix calculation
// ----------------------------------------------------------------------------
void  calc_vc(double *A, double *B, int n1, int n2)  {
    int  i, j, k;
    double  M1[1000] = {0,};     // NOTE: select a large number (e.g., 1000)
    double  M2[1000] = {0,};     //       make it sure (n1<1000) in this example.
    double  coVAR;

    // Dimensions
    // A(n1,n2), B(n2,n2)

    for (i=0; i<n2; i++)  {
        // Get the first column vector
        for (k=0; k<n1; k++)
            M1[k] = A[i*n1+k];

        for (j=0; j<n2; j++)  {
          if (i >= j)  {

              if (i == j)
                  // compute variance
                  coVAR = calc_var(M1, n1);
              else  {
                  // Get the second column vector
                  for (k=0; k<n1; k++)
                      M2[k] = A[j*n1+k];

                  // compute covariance
                  coVAR = calc_cov(M1, M2, n1);
              }
              // Augment vc-matrix
              B[i*n2+j] = coVAR;

          }  else  {

              // Utilize symmetric characteristics
              B[i*n2+j] = B[j*n2+i];

          }

      }

  }

}


/*#######################################################################*/
/* ------------------------------------------------------------------ */
/*     MahalMaxDist                                                  */
/*      Function to select the max mahalanobis dist in a kernel          */
/*      Called once only at end of Mahal processing                   */
/* ------------------------------------------------------------------ */
/*#######################################################################*/



void MahalMaxFnct (float *image_buffrMahMax, int num_pixels, int num_lines,  int
half_kernel, int kernel_size, int dbic_list[]){


int line, pixel, j, i,MaxLine,MaxPixel,MaxFlag, cntr;

int numlinesdown, numpixsintoline;


float maxmahdist,dist,mean,sum,diffsrd,stnd_dev,maxthresratio;
```

```
//printf("inside MahalMaxFnct \n");

dist = 0;
cntr = 0;
sum = 0;
diffsrd = 0;
/*
    for (line = 0; line < num_lines; line++){

        for ( pixel = 0; pixel < num_pixels; pixel++){

            numlinesdown = line * num_pixels * 2;
            numpixsintoline = pixel * 2;
            //printf("Chan 10 before =  *(image_buffrMahMax + numlinesdown +
numpixsintoline) = %f \n", *(image_buffrMahMax + numlinesdown + numpixsintoline));
            *(image_buffrMahMax + numlinesdown + numpixsintoline) = 0;
            //printf("Chan 10 after =  *(image_buffrMahMax + numlinesdown + numpixsintoline)
= %f \n", *(image_buffrMahMax + numlinesdown + numpixsintoline));
            //printf("next channel 11 = %f \n", *(image_buffrMahMax + numlinesdown +
numpixsintoline + 1));
        }
    }

*/
//printf("exiting early /n");
//exit(1);




    //two for loop to go through the entire image
    for (line = half_kernel; line < (num_lines  - half_kernel); line ++){

        for (pixel = half_kernel; pixel < (num_pixels - half_kernel); pixel ++){
        //printf ("New Kernel \n");
        maxmahdist = 1;
        maxthresratio = 0;
        MaxLine = 0;
        MaxPixel = 0;
        MaxFlag = 0;
        numlinesdown = 0;
        numpixsintoline = 0;
        //printf("MaxFlag = %d \n", MaxFlag);


        // Calculating Kernal Stats


            cntr = 0;
            dist = 0;
            mean = 0;
            diffsrd = 0;
            stnd_dev = 0;
            for ( i = (line - half_kernel); i < ( line + half_kernel + 1); i++){


                for ( j = ( pixel - half_kernel); j < (pixel + half_kernel + 1); j++){

                    numlinesdown = i * num_pixels * 2;
                    numpixsintoline = j * 2;

                    //printf("dist = %f \n", *(image_buffrMahMax + numlinesdown +
numpixsintoline));
                    dist = dist + *(image_buffrMahMax + numlinesdown + numpixsintoline);
                    //printf("cuml_dist = %f \n", dist);
                    cntr = cntr++;
                    //printf("cntr = %d \n", cntr);


                } // end of pixel in kernel loop

            } //end of line in kernel loop


            // sample mean

            mean = dist/cntr;
            //printf("mean = %f \n\n", mean);
            //sample standard deviation


            for ( i = (line - half_kernel); i < ( line + half_kernel + 1); i++){


                for ( j = ( pixel - half_kernel); j < (pixel + half_kernel + 1); j++){

                    numlinesdown = i * num_pixels * 2;
                    numpixsintoline = j * 2;

                    //printf("dist = %f \n", *(image_buffrMahMax + numlinesdown +
numpixsintoline));
                    diffsrd = diffsrd + pow( mean - *(image_buffrMahMax + numlinesdown +
numpixsintoline), 2);

                } // end of pixel in kernel loop

            } //end of line in kernel loop

            stnd_dev = sqrt (fabs(diffsrd / cntr));
            //printf("standard Dev  = %f \n\n", stnd_dev);
            dist = 0;
            for ( i = (line - half_kernel); i < ( line + half_kernel + 1); i++){


                for ( j = ( pixel - half_kernel); j < (pixel + half_kernel + 1); j++){

                    numlinesdown = i * num_pixels * 2;
                    numpixsintoline = j * 2;

                    //printf("dist = %f \n", *(image_buffrMahMax + numlinesdown +
numpixsintoline));

                    dist = *(image_buffrMahMax + numlinesdown + numpixsintoline);
                    //printf("mean = %f \n", mean);
                    //printf("dist - mean = %f \n", (dist - mean));
                    //printf("standard Dev  = %f \n", stnd_dev);
                    //printf("Threshold Ratio  = %f \n\n", (fabs(dist - mean) / stnd_dev));
                    //THRESHOLD RATIO TEST
                    if (dist > maxmahdist) {
                        //printf("Inside dist if, MaxFlag = %d \n", MaxFlag);
                        maxmahdist = dist;
                        maxthresratio = (fabs(dist - mean) ) / stnd_dev;
                        //printf("Max dist = %f \n", maxmahdist);
                        MaxLine = i;
                        MaxPixel = j;
                        MaxFlag = 1;

                    } // end of if statement

                } // end of pixel in kernel loop

            } //end of line in kernel loop

            //printf(" Before Max Flag If MaxFlag = %d \n", MaxFlag);
            // Add one to the Frequency channel for the pixel having the largest maxdist

            if (MaxFlag == 1 && maxthresratio > 0.5){ //list is printed to the screen for the user

                if (line == 10 && pixel == 10)
                    printf("Maximum Threshold Ratio = 0.5 \n");
                //printf("Inside freq if, MaxFlag =  %d\n", MaxFlag);

                //printf(" A max dist was selected \n");

                numlinesdown = MaxLine * num_pixels * 2;
                numpixsintoline = MaxPixel * 2;
                //printf("Chan 12 before =  *(image_buffrMahMax + numlinesdown +
numpixsintoline + 1) = %f \n", *(image_buffrMahMax + numlinesdown + numpixsintoline +
1));
                *(image_buffrMahMax + numlinesdown + numpixsintoline + 1) =
(*(image_buffrMahMax + numlinesdown + numpixsintoline + 1) + 1);
```

```
                //printf("Chan 12 after =  *(image_buffrMahMax + numlinesdown +
numpixsintoline + 1) = %f \n", *(image_buffrMahMax + numlinesdown + numpixsintoline +
1));
            } //end of MaxFlag if statement

        }//end of pixel loop

    } // end of line loop


}// end of MahalMaxFnct


/*##########################################################################*/
/* ------------------------------------------------------------------ */
/*     init_stack                                                   */
/*      Function to initial the stack for characterization by J.Beaudoin*/
/* ------------------------------------------------------------------ */
/*##########################################################################*/


void init_stack (element_stack *the_stack)
{
      the_stack->top = -1;
      the_stack->stack_size = 3600;
      memset(the_stack->stack,0,3600 * sizeof(element));


      return;
}



/*##########################################################################*/
/* ------------------------------------------------------------------ */
/*    Pop                                                          */
/*      Function to get pixel coords off the stack                   */
/*      coding by J.Beaudoin idea by K.P.                           */
/* ------------------------------------------------------------------ */
/*##########################################################################*/

element pop(element_stack *the_stack)
{
      element return_element;


      if (the_stack->top == -1)
      {
            return_element.pixel[0] = -999;
            return_element.line[0] = -999;
      }
```

```
        else
        {
              return_element.pixel[0] = the_stack->stack[the_stack->top].pixel[0];
              return_element.line[0] = the_stack->stack[the_stack->top].line[0];
              the_stack->top--;
        }


        return(return_element);
}




/*#########################################################################*/
/* ------------------------------------------------------------------- */
/*      push                                                        */
/*       Function to put a set of pixel coords on the stack               */
/*       coding by J.Beaudoin idea by K.P.                            */
/* ------------------------------------------------------------------- */
/*#########################################################################*/


void push(element_stack *the_stack, element shit_to_push)
{
        the_stack->top++;


        the_stack->stack[the_stack->top].pixel[0] = shit_to_push.pixel[0];
        the_stack->stack[the_stack->top].line[0] = shit_to_push.line[0];


        return;
}

/*#########################################################################*/
/* ------------------------------------------------------------------- */
/*      Characterizaton                                               */
/*       Function to perform target labelling and characterization        */
/* ------------------------------------------------------------------- */
/*#########################################################################*/



void CharacterFnct (float *image_buffrCharactr, int num_pixels,
                    int num_lines,  int half_kernel, int kernel_size, int dbic2_list[],float thrshold,
FILE *targetsfile){


int targetlabel,line,pixel,numlinesdown,numpixsintoline, direct_cntr,
numlinesdownb, numpixsintolineb, pixsindirectn, heading[9],i,j,sigma_targets;
```

```
float deltapix, deltaline;

element a_targetcoord, first_targetcoord, b_targetcoord;

element_stack targets_stack;

char orient[7];

//attribute_array the_target_attributes;

//printf("Into Characterization \n");

targetlabel = 101; //labels begin at 100 and end up in the freq channel which has a max value of 10x10 window - crazy

dir_list[0].R[0] = -1;
dir_list[0].C[0] = 0;

dir_list[1].R[0] = -1;
dir_list[1].C[0] = -1;

dir_list[2].R[0] = 0;
dir_list[2].C[0] = -1;

dir_list[3].R[0] = 1;
dir_list[3].C[0] = -1;

dir_list[4].R[0] = 1;
dir_list[4].C[0] = 0;

dir_list[5].R[0] = 1;
dir_list[5].C[0] = 1;

dir_list[6].R[0] = 0;
dir_list[6].C[0] = 1;

dir_list[7].R[0] = -1;
dir_list[7].C[0] = 1;

//get the stack ready to go


init_stack (&targets_stack);
//printf("Past Init stack \n");
//loop through the 2 channel image
for (line = 0; line < num_lines; line++){
    //printf("Characterization line# = %d\n", line);
        for ( pixel = 0; pixel < num_pixels; pixel++){

            numlinesdown = line * num_pixels * 2;
            numpixsintoline = pixel * 2;

            //first test to see if this pixel is a target by its freq or label
            if ( *(image_buffrCharactr + numlinesdown + numpixsintoline + 1) < 100) { //if
greater than 100 go to the next pixel as it has already been labelled
                //frequency tests
                //1) if < highest frequency then set to zero
                //if ( *(image_buffrCharactr + numlinesdown + numpixsintoline + 1) <
(kernel_size * kernel_size) ){
                //   *(image_buffrCharactr + numlinesdown + numpixsintoline + 1) = 0;
                //}

                if ( *(image_buffrCharactr + numlinesdown + numpixsintoline + 1) >
((kernel_size * kernel_size) - 2)){


                if (line == 10 && pixel == 10)
                    printf("Frequency Threshold = %d \n", ((kernel_size * kernel_size) - 1));
                // any pixel making it to here is to be labelled
                // begin search
                    //push the pixel onto the stack


                    first_targetcoord.pixel[0] = pixel;
                    first_targetcoord.line[0] = line;
                    push (&targets_stack, first_targetcoord);

                    //keep going until all elements are off the stack

                    while (1) {

                        a_targetcoord = pop(&targets_stack);

                        if (a_targetcoord.pixel[0] == -999)
                            break; //ends terminates the endless loop

                        //label the seed pixel from the stack
                        numlinesdownb = a_targetcoord.line[0] * num_pixels * 2;
                        numpixsintolineb = a_targetcoord.pixel[0] * 2;
                        *(image_buffrCharactr + numlinesdownb + numpixsintolineb + 1) =
targetlabel;

                        //now for each direction from the particular stack element go until no more
boat adding to stack if still boat
                        for (direct_cntr = 0; direct_cntr < 8; direct_cntr++){
                            //printf("direct_cntr = %d\n",direct_cntr);
                            //using the direction counter move along a direction until edge of boat
                            pixsindirectn = 1;
                            while (1){
                                // don't go farther than the edge of the image
                                if ( (a_targetcoord.line[0] + (pixsindirectn *
dir_list[direct_cntr].R[0])) < 0 ||
                                    ( a_targetcoord.line[0] + (pixsindirectn *
dir_list[direct_cntr].R[0])) > (num_lines - 1) )
                                    break; //past the image edge

                                if ( (a_targetcoord.pixel[0] +(pixsindirectn *
dir_list[direct_cntr].C[0])) < 0 ||
                                    (a_targetcoord.pixel[0] +(pixsindirectn *
dir_list[direct_cntr].C[0]) ) > (num_pixels - 1) )
                                    break; //past the image edge

                                //using the direction counter move along a direction until edge of boat
                                numlinesdownb = ( a_targetcoord.line[0] + (pixsindirectn *
dir_list[direct_cntr].R[0]) ) * num_pixels * 2;
                                numpixsintolineb = (a_targetcoord.pixel[0] +(pixsindirectn *
dir_list[direct_cntr].C[0]) ) * 2;



                                // test to see if it is still a boat OR if has previously been labelled
                                if (*(image_buffrCharactr + numlinesdownb + numpixsintolineb) <
thrshold ||
                                    *(image_buffrCharactr + numlinesdownb + numpixsintolineb + 1)
>= 100){
                                    //printf("heading = %d with pixsindirectn =
%d\n",direct_cntr,pixsindirectn);
                                    break; //stops looping along a particular direction

                                } //Edge of boat IF
                                //otherwise put that pixel on the stack to be checked and labelled
                                b_targetcoord.line[0] = a_targetcoord.line[0] + (pixsindirectn *
dir_list[direct_cntr].R[0]);
                                b_targetcoord.pixel[0] = a_targetcoord.pixel[0] +(pixsindirectn *
dir_list[direct_cntr].C[0]);
                                push (&targets_stack, b_targetcoord);
                                pixsindirectn = pixsindirectn + 1;
                                //printf("pixsindirection = %d\n",pixsindirectn);
                            }// bottom of while loop - contiuing along a particular direction
                        }// bottom of the direction loop
                    } //bottom of infinite while loop for the entire STACK
                    //call the Boat Size function and report on that particular target
                    //boatsizefnctn(first_targetcoord, heading, targetsfile,targetlabel);
                    targetlabel = targetlabel + 1;
                    printf("Number of targets = %d\n", (targetlabel - 100));
                }//frequency filter IF
            } //IF < 100

        } //end of pixel loop
}// end of line loop */

//loop again to  gather attributes for the found targets.
```

```
//loop through the 2 channel image



for ( i = 100; i < (targetlabel + 1); i++){

  j = 0;
    for (line = 0; line < num_lines; line++){
        //printf("second line loop # = %d \n",line);
        for ( pixel = 0; pixel < num_pixels; pixel++){
        //printf("second pixel loop # = %d \n",pixel);
            numlinesdown = line * num_pixels * 2;
            numpixsintoline = pixel * 2;
            //pixel is labelled with the label of interest
            if (*(image_buffrCharactr + numlinesdown + numpixsintoline + 1) == i) {
                j = j + 1;
                //printf("target label = %d\n", i);
                //printf("j = %d\n", j);
                //printf("second line loop # = %d \n",line);
                //printf("second pixel loop # = %d \n",pixel);
                //first pixel is always one end of the boat
                if (j==1){
                    //craps out in here
                the_target_attributes[i].smpixel[0] = pixel;
                the_target_attributes[i].smline[0] = line;
                the_target_attributes[i].numbrpixs[0] = j;
                }//bottom of first pixel

                //SEEK THE OTHER END OF THE BOAT
                //printf("Other end of boat \n");
                the_target_attributes[i].lrgpixel[0] = pixel;
                the_target_attributes[i].lrgline[0] = line;
                the_target_attributes[i].numbrpixs[0] = j;
                //printf("numbrpixs[%d] = %d\n",i, the_target_attributes[i].numbrpixs[0]);
                }// end of if testing it is a correctly labelled pixel

        } //pixel loop

    } // line loop

}//bottom of i target# loop


//printf("starting the BIG second looper\n");
//Now calculate the attributes
sigma_targets = 0;
for ( i = 101; i < targetlabel ; i++){
    //printf("i = %d\n", i);
    if ( the_target_attributes[i].numbrpixs[0] == 1){

        //SINGLE PIXEL TARGETS UNLABELLED

        //*(image_buffrCharactr + (the_target_attributes[i].smline[0] * num_pixels * 2) +
(the_target_attributes[i].smpixel[0] * 2) + 1) = 0;
        the_target_attributes[i].length[0] = 1;
        //printf("numbrpixs[%d] = %d\n",i, the_target_attributes[i].numbrpixs[0]);
        the_target_attributes[i].width[0] = 1;
        the_target_attributes[i].cntrpixel[0] = the_target_attributes[i].smpixel[0];
        the_target_attributes[i].cntrline[0] = the_target_attributes[i].smline[0];
        //of course a single pixel target doesn't have orientation so this is a default value.
        //ptr_orient = "N/A";
        sprintf(orient,"N/A");
    }//end of one pixel boat
    //printf("past the IF\n");
    if ( the_target_attributes[i].numbrpixs[0] > 1){

        //calculate the length note: the addition of one to include the entire last pixel to the lenght
        deltapix = pow (the_target_attributes[i].lrgpixel[0] - the_target_attributes[i].smpixel[0],
2);
        deltaline = pow (the_target_attributes[i].lrgline[0] - the_target_attributes[i].smline[0], 2);
        the_target_attributes[i].length[0] = sqrt (deltapix + deltaline);

        //kind of a cludgie width calculation but demonstrates what is doable and is robust
        the_target_attributes[i].width[0] = the_target_attributes[i].numbrpixs[0] /
the_target_attributes[i].length[0];

        //ORIENTATION OF THE TARGET

        if (the_target_attributes[i].lrgpixel[0] == the_target_attributes[i].smpixel[0]  &&
            the_target_attributes[i].lrgline[0] > the_target_attributes[i].smline[0]){
            //orient = "N/S";
            sprintf(orient,"N/S");
            the_target_attributes[i].cntrline[0] = the_target_attributes[i].smline[0] +
                (floor(the_target_attributes[i].lrgline[0] - the_target_attributes[i].smline[0])/2);
            the_target_attributes[i].cntrpixel[0] = the_target_attributes[i].smpixel[0];
        }
        if (the_target_attributes[i].lrgline[0] == the_target_attributes[i].smline[0]  &&
            the_target_attributes[i].lrgpixel[0] > the_target_attributes[i].smpixel[0]){
            //orient = "E/W";
            sprintf(orient,"E/W");
            the_target_attributes[i].cntrpixel[0] = the_target_attributes[i].smpixel[0] +
                (floor(the_target_attributes[i].lrgpixel[0] - the_target_attributes[i].smpixel[0])/2);
            the_target_attributes[i].cntrline[0] = the_target_attributes[i].smline[0];
        }
        if (the_target_attributes[i].lrgline[0] > the_target_attributes[i].smline[0]  &&
            the_target_attributes[i].lrgpixel[0] > the_target_attributes[i].smpixel[0]){
            //orient = "NW/SE";
            sprintf(orient,"NW/SE");
            the_target_attributes[i].cntrpixel[0] = the_target_attributes[i].smpixel[0] +
                (floor(the_target_attributes[i].lrgpixel[0] - the_target_attributes[i].smpixel[0])/2);
            the_target_attributes[i].cntrline[0] = the_target_attributes[i].smline[0] +
                (floor(the_target_attributes[i].lrgline[0] - the_target_attributes[i].smline[0])/2);
        }
        if (the_target_attributes[i].lrgline[0] > the_target_attributes[i].smline[0]  &&
            the_target_attributes[i].lrgpixel[0] < the_target_attributes[i].smpixel[0]){
            //orient = "NE/SW";
            sprintf(orient,"NE/SW");
            the_target_attributes[i].cntrpixel[0] = the_target_attributes[i].smpixel[0] -
                (floor(the_target_attributes[i].smpixel[0] - the_target_attributes[i].lrgpixel[0])/2);
            the_target_attributes[i].cntrline[0] = the_target_attributes[i].smline[0] +
                (floor(the_target_attributes[i].lrgline[0] - the_target_attributes[i].smline[0])/2);
        }
    //regardless of number of pixels comprising a boat, print out the results

    }//end of boats of more than one pixel

    fprintf(targetsfile,"    %d    \t%d\t%d\t%f\t%f\t%s\n",
        i,
        the_target_attributes[i].cntrpixel[0],
        the_target_attributes[i].cntrline[0],
        the_target_attributes[i].length[0],
        the_target_attributes[i].width[0],
        orient);
    sigma_targets++;


    //printf("bottom of the targetlabel looper\n");
}//bottom of the target label loop
    fprintf(targetsfile,"\nTotal number of targets = %d\n",sigma_targets);
    //printf("bottom of characterization\n");
} //end of Characterization function


/*########################################################################*/
/* ------------------------------------------------------------------ */
/*     Boat size                                                      */
/*      Function to calculate target attributes                       */
/* ------------------------------------------------------------------ */
/*########################################################################*/

void boatsizefnctn(element first_targetcoord, int heading[], FILE *targetsfile, int targetlabel){

int totalnum_pixels,i,finalheading[4], length, width, Azimuth;
totalnum_pixels = 0;
Azimuth = 0;
width = 0;
length = 0;

for (i = 0; i<8; i++){
    //printf("heading[%d] = %d\n",i,heading[i]);
    totalnum_pixels = totalnum_pixels + heading[i];
}// bottom of for loop
    length = heading[0] + heading[4];

        for (i = 0; i<4; i++){
        finalheading[i] = heading[i] + heading[i + 4];
        //printf("finalheading[%d] = %d\n",i,finalheading[i]);
        // figure out the length
        if (finalheading[i] > length){
            length = finalheading[i];
            Azimuth = i;
        }
        }// end of length/azimuth loop

//figure of the width of the boat
        // Az => e/w
        if (Azimuth == 0){
        width = finalheading[2];
        }
        // Az => NW/SE
        if (Azimuth == 1){
        width = finalheading[3];
        }
        // Az => N/S
        if (Azimuth == 2){
        width = finalheading[0];

        }
        // Az => NE/SW
        if (Azimuth == 3){
        width = finalheading[1];
        }

// Generate Target Report


    fprintf(targetsfile,"\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\n",targetlabel,(first_targetcoord.pixel[
0] + 1),(first_targetcoord.line[0] + 1),length,width, Azimuth,totalnum_pixels);




}//end of boat size function
```

APPENDIX II
SOFTWARE DESIGN 1

This appendix contains information regarding the programming issues surrounding the initial implementation of MRV Recon.

The initial design of MRV Recon duplicated the spatio-spectral template developed by Subramanian and Gat [1998]. Once duplicated, the spatio-spectral template was tested using IKONOS imagery. Once the testing was completed, areas for enhancing MRV Recon could be identified.

This is the first of two appendices dedicated to software design. The goal of these appendixes is to highlight the major functions within MRV Recon. For the sake of clarity, many "housekeeping" aspects of the software are not included in these chapters. For example, several functions are called to handle the special case of pixels along the edges of the image. They are relatively simple, but not discussing them makes it easier for the reader to wade through what is already quite tedious material. However, those interested in the details can find all the C code for MRV Recon in Appendix I.

II.I The PCI C/C++ Toolkit

Included with PCI V. 7.0 is a library of C functions that perform many standard image analysis functions, including: input and output, statistics, image queries, among others. By providing a toolkit, advanced users can develop complex image processing software without having to write code for many typical functions. In order to access the PCI C Toolkit, a header file must be included in the code:

```
#include "pci.h"
```

Once the header file is included, the library can be accessed. The following is an example of a commonly used C function that allows the reading into a buffer of a *.pix* file:

```
IDBRealChanIO (fp,IDB_READ, 0, line_pos, num_pixels,
kernel_size,image_buffr,num_pixels,kernel_size,numIkonosChan,dbic_list);
```

The C function automatically reads the imagery header file and then places the required data into a buffer. In this case, MRV Recon has a memory buffer allocated called *image_buffr*. The function only needs the offsets (*0, line_pos, num_pixels, kernel_size)* to the portion of the imagery required, and the channels (*dbic_list*) to be copied. Writing out from a buffer to a file is done in exactly the same manner except for changing the parameter *IDB_READ* to *IDB_WRITE*.

There is no intent to create a programmer's guide for the PCI C Toolkit. However, beyond this brief introduction there will be interspersed comments on specific issues surrounding some toolkit functions used in creating MRV Recon.

II.II Design Criteria

From the outset, MRV Recon was envisioned to have the following criteria:

- Automatic
- Simple to use
- Flexible
- Accurate
- Robust

The primary design criterion is that the software must be usable for non-remote sensing experts. Ultimately, when MARIS and MRV Recon become operational, CCG personnel will use them. It is important to provide a system that is automatic, simple to use, only requiring minimal instruction.

MRV Recon is designed to be flexible. It is anticipated that imagery from different vendors would be processed so the design allows for any size of imagery, with any number of channels, and differing resolutions.

Finally, MRV Recon is designed to be accurate and robust. The focus of the entire work is predicated on how accurately MRV Recon can detect small recreational vessels. Even though it is designed to be simple to use, requiring minimal user interaction, ideally the software should be designed to trap any errors that occurred.

II.III MRV Recon Memory Handling

A major decision in designing MRV Recon – even prior to duplicating the algorithm from Subramanian and Gat - concerned the handling of the memory buffers for loading the image. Therefore, the memory buffers are designed for the required flexibility. MRV Recon was tested using IKONOS imagery. However, it is designed to handle imagery of any resolution, any size, and with a varying number of channels. Therefore, the size of the buffers to hold the imagery accommodates both different imagery, and different sizes of user-defined *nxn* processing kernels.

The following is the fast memory allocation PCI function call used in MRV Recon:

```
image_buffr = (float *) malloc(sizeof(float) * num_pixels * numIkonosChan *
kernel_size);
```

The function *malloc* reserves a block of memory, in this case reserving elements of memory being equal to size of the data type *float*. The dimensions of the block being reserved is set to the width of the image in pixels and having a length equal to the size of the processing kernel. So, for example, if a 3x3 kernel is used to process an image of 1000 pixels wide by 2000 pixels long, that contains 5 channels, then a buffer of 1000 x 3 x 5 (of size *float*) is reserved. Utilizing a loop, the buffer calls for each line in the image.

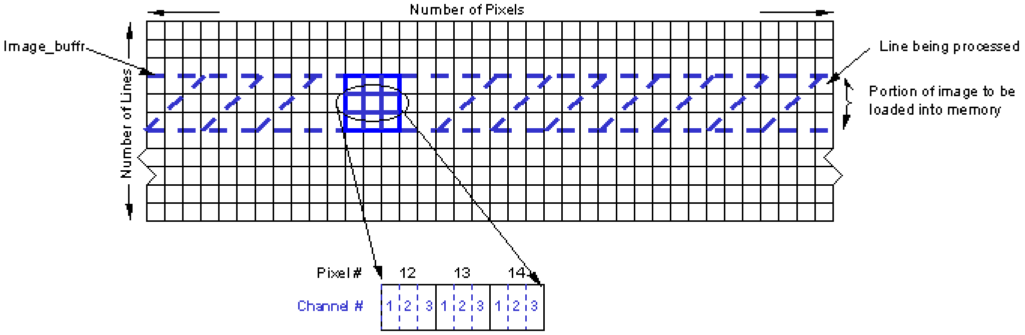


Figure II.I Illustration of MRV Recon memory buffer loading

Figure II.I conceptually illustrates how the imagery file is loaded into memory. The matrix of cells represents the entire image being processed by MRV Recon. As stated above, the program loops through each line of the image. At each line a portion of the image is loaded into memory. The rectangle is surrounded with dashed lines and contains the dashed diagonal lines. This represents the portion of the imagery loaded into a reserved memory buffer. Figure II.I shows the line being operated on was the fifth line of the image.

MRV Recon then passes a kernel, in this case a 3x3 kernel, across this portion of memory. A loop is used to investigate each of the 9 pixels of the kernel. Shown in the inset of Figure II.I are pixel numbers 12, 13, and 14. In the example shown in Figure II.I, each pixel has three channels. Each individual channel represents the smallest division of memory in a buffer. It is here where the digital numbers (DN) are recorded for each band of the satellite. The DN represents the radiance for each band recorded by the CCD in the satellite.

It is important to realize that Figure II.I is conceptual only. It is a way to understand how the imagery was stored and manipulated. In reality the computer physically stores the data in a different manner. For example, the block of memory reserved to store a portion of the imagery is not a two-dimensional array, as shown in Figure II.I, but rather a one-dimensional array of memory. Nonetheless, each individual piece of memory can be accessed as if it is stored in two dimensions and sometimes it is easier to think of it in a two-dimensional sense.

From the example shown in Figure II.I, how is the recorded DN for channel 3 of pixel number 13 (see inset) accessed? As described above, a block of memory of sufficient size is reserved and, for the example in Figure II.I named *image_buffr*. In the C language, using only the name, the first memory address for this block can be accessed. Using pointer arithmetic the remaining "portions" of data can be accessed. A pointer in C is a programming device that is used to find a specific piece of memory. The DN for band k for a particular pixel (having a pixel number p and line number l) is;

$$DN(p,l,k) = N_p * l * N_k + p * N_k + k \quad \text{(II.I)}$$

where

Np = width of image in pixels

Nk = number of channels

For example, accessing channel 3 of pixel 13 would be done in the following manner:

1. The data in the beginning of the memory block is accessed using the following statement:

   ```
   *(image_buffr)
   ```

2. Pixel 13 is towards the middle of the second line of the image buffer (refer to Figure II.I). Therefore, the number of whole lines along the memory buffer is 1. Each line comprises, in this example, 38 pixels. Each pixel contains three channels. So the pointer is moved to the beginning of the second line by:

   ```
   *(image_buffr +  numlinesdown * num_pixels*num_channels):
   ```

   Where;

   ```
   numlinesdown = 1;
   num_pixels = 38;
   num_channels = 3;
   ```

3. The final step is to move the pointer to channel 3, pixel 13, of the second line of the buffer. Therefore, 12 whole pixels having three channels each must be added to the pointer. Lastly, now that the pointer is within pixel 13, it must be moved to the third channel. So the entire pointer arithmetic is:

   ```
   *(image_buffr +  numlinesdown * num_pixels*num_channels +
       numpixsintoline * num_channels +  channel);
   ```

   Where;

   ```
   numlinesdown = 1;
   num_pixels = 38;
   num_channels = 3;
   numpixsintoline = 12;
   channel = 3;
   ```

Although there are other common ways of handling pointers for imagery such as using a two dimensional pointer addressing, the above is conceptually simple and has worked well. Although the above is a fictitious example, the variable names are consistent with the variable names found in MRV Recon. The C code for MRV Recon is attached in Appendix I.

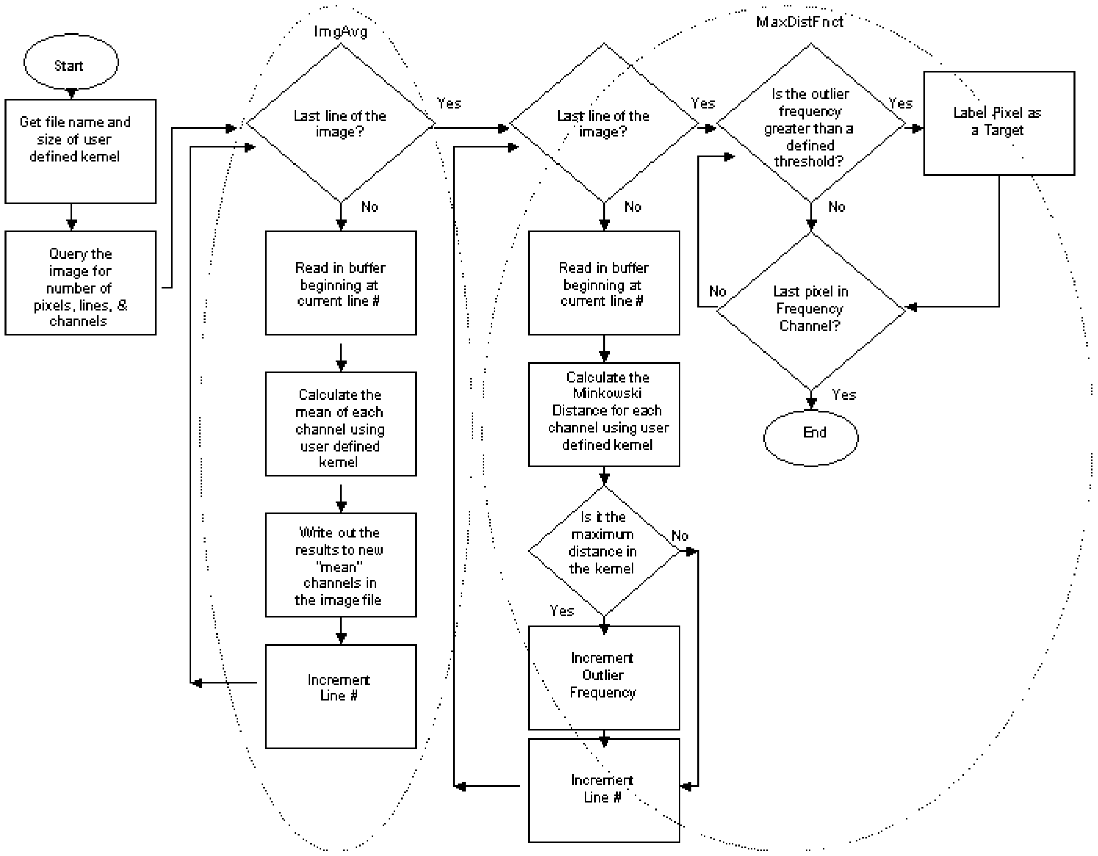

Start
Get file name and size of user defined kernel
Query the image for number of pixels, lines, & channels
ImgAvg
Last line of the image?
Yes
No
Read in buffer beginning at current line #
Calculate the mean of each channel using user defined kernel
Write out the results to new "mean" channels in the image file
Increment Line #
MaxDistFnct
Last line of the image?
Yes
No
Read in buffer beginning at current line #
Calculate the Minkowski Distance for each channel using user defined kernel
Is it the maximum distance in the kernel
No
Yes
Increment Outlier Frequency
Increment Line #
Is the outlier frequency greater than a defined threshold?
Yes
No
Label Pixel as a Target
Last pixel in Frequency Channel?
No
Yes
End

Figure II.II MRV Recon initial design I flowchart.

In retrospect, the initial design of handling the memory buffers was poor in terms of the processing speed of MRV Recon. It certainly addressed the criterion of flexibility especially in handling very large datasets, but this was at the expense of considerable input/output (I/O) to the hard drive. Any access to the hard drive is inherently slow. The small size of the initial test datasets hid the impact of this flaw. However, it is a problem that could be overcome in future versions of MRV Recon by increasing the buffer size and thus reducing the I/O to the hard drive.

II.IV MRV Recon Design

The initial design of MRV Recon is shown in Figure II.II. Again, this initial design duplicated Subramanian and Gat's spatio-spectral template. Once the filename and kernel size are retrieved from the user, and the image file is queried for its size and number of channels, the software design can be broken down into two functions: 1) *ImgAvg* function that performed image averaging and 2) the *MaxDistFnct* function that calculated the maximum Minkowski distance and increments the outlier frequency.

II.IV.I Image Averaging

The *ImgAvg* function began with a loop over the entire image. For each line of the image, a buffer the width of the image having a length of the size of the user-defined kernel is read into the memory reserved. A second loop moves a kernel "across" the buffer for each pixel. Next a loop is made for every pixel in the kernel. Finally, a loop is made for each channel. The values for each channel are summed and the average is calculated for each channel for the centre pixel of a kernel. These averages are then placed into another buffer of the same size of the first. Once the entire line is processed this "average" buffer is then written out to empty channels in the image.

The PCI toolkit function *IDBRealChanIO* will not allow the overwriting of an existing channel. This is an inconvenience in the design of MRV Recon which meant that there must be new empty channels existing in the imagery file prior to processing, with MRV Recon ready to receive the output from the *ImgAvg* function.

II.IV.II Maximum Distance Calculation

The *MaxDistFnct* function, utilized a similar cascading buffer but in this function it had to be larger to accommodate the larger number of channels – the original channels in addition to the average channels produced by the *ImgAvg* function. The Minkowski uses the raw digital number (DN) values and the newly calculated average DN values.

$$d = \sqrt[r]{\sum_{i=1}^{p} \left| x_i - y_i \right|^r} \qquad \text{(II.II)}$$

Equation (II.II) is the general Minkowski equation [Hartigan, 1975]. The Minkowski distance is commonly known as the “Euclidean distance” when r = 2. When r = 1 it is commonly known as the “Manhattan distance”. The user is prompted by MRV Recon as to whether the Euclidean or Manhattan distance should be used. The value of the exponent variable (i.e. *r*) is set depending on which metric was selected. The variable *p* is the number of channels.

Similar to the image average function described above, the distance is calculated for every pixel in the kernel as it moves across a line. Mathematically (for a Euclidean distance):

for a *N*x*N* kernel, and for all *p*,*l*:

$$f(p,l) = f(p,l) + 1, \text{ if}$$

$$dist(p,l) = \max \sqrt[2]{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N_k} \left| x_{i,j,k} - \bar{x}_{i,j,k} \right|^2}$$

where:

p = pixel

l = line

i = pixel in NxN kernel

j = line in NxN kernel

k = channel

N_k = number of channels

The distance is calculated between the original channel values and the values calculated for the mean channels. The maximum distance within a kernel is then selected. The program then increments that pixel's outlier frequency by 1 and stores this frequency in a separate channel. The pixels having a high outlier frequency f, are then labeled as targets.

II.V Summary

The goal of this initial design of MRV Recon is to reproduce the spatio-spectral template. The design criteria for MRV Recon are as follows: automatic, simple, flexible, accurate, and robust.

The memory buffers reserved for MRV Recon must be large enough to store a number of lines of the image equaling the size of the user-defined processing kernel. The program reads in and writes out a portion of the image corresponding to the buffer size for each line in the image. While this is I/O intensive, it allows virtually any sized image to be processed.

This initial design of MRV Recon is composed of two major functions. The first function calculates the average of the original channels for a user-defined kernel size. The second function selects the maximum Minkowski distance within each location of the convolved kernel. Each time a pixel is found to be a maximum its outlier frequency is incremented by one. Those pixels with the highest frequency of being an outlier, or exceeding a predefined threshold, are labeled as targets.

APPENDIX III
SOFTWARE DESIGN 2

This appendix describes the coding of the enhancements made to the early work of Subramanian and Gat [1998]. Further, it outlines the programming issues surrounding other improvements to MRV Recon and provides a link to the discussions and testing of the third journal paper included in Chapter 6. As in Appendix II, only the most important software design issues are discussed. Again, key problems encountered will be analyzed.

III.I Integrating the Mahalanobis/WED Distance

In order to test if the new distance metric is an improvement, MRV Recon had to be redesigned to be able to allow the user to select either the Minkowski or Mahalanobis distance. While seemingly simple, it became more and more complex as a selection of Mahalanobis distance metric required loading buffers of different sizes and channels than those used for the Minkowski distance metric.

Equation (III.I) is the Mahalanobis distance metric. Most of the quantities for this have already been calculated. The **($\mathbf{x}$-$\mathbf{m}_x$)** vector is the difference between the original vector and the mean vector. The means are calculated for use in the Minkowski distances and used for this function.

$$\boldsymbol{d}^2 = \left(\boldsymbol{x} - \boldsymbol{m}_x\right)^T \boldsymbol{C}_X^{-1} (\boldsymbol{x} - \boldsymbol{m}_x) \tag{III.I}$$

The covariance matrix ($\mathbf{C_x}$) is a new quantity to be calculated. The calculation of the $\mathbf{C_x}$ matrix requires a larger window surrounding the user-defined kernel. The user-defined kernel defines the number of pixels surrounding the pixel currently being processed to see which has the maximum distance. However, the window surrounding the pixel being processed for deriving the $\mathbf{C_x}$ matrix must be large enough so that a sufficient sample size is gathered to avoid a non-singular $\mathbf{C_x}$ matrix. A non-singular matrix cannot be inverted using the Gauss-Jordan inverse. Later, the problem of inversion would not be important because the WED metric does not require inversion of the $\mathbf{C_x}$ matrix.

To accommodate the calculation of the $\mathbf{C_x}$ matrix, a two-tiered memory buffer is developed. Figure III.I shows the conceptual representation of the memory buffer used in the Mahalanobis Distance Function (*MahalDistFnct*)..

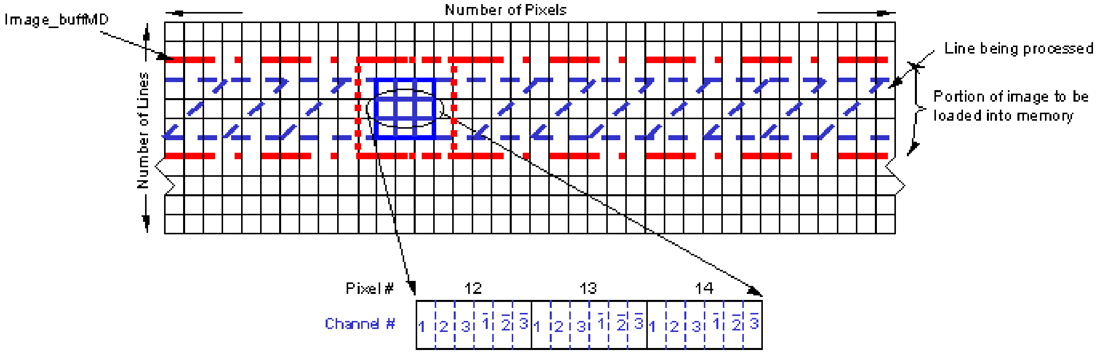

Number of Pixels
Image_buffMD
Number of Lines
Line being processed
Portion of image to be loaded into memory
Pixel #
12
13
14
Channel #
1 2 3 $\bar{1}$ $\bar{2}$ $\bar{3}$ 1 2 3 $\bar{1}$ $\bar{2}$ $\bar{3}$ 1 2 3 $\bar{1}$ $\bar{2}$ $\bar{3}$

Figure III.I Two-tiered memory buffer used in *MahalDistFnct*.

The *MahalDistFnct* is called for each line of the image. In order to accommodate a larger window for the $\mathbf{C_x}$ matrix, a larger buffer must be loaded into memory. In the case shown in Figure III.I, the user-defined processing kernel is 3x3 in size surrounding pixel 13 shown in the inset. A larger window surrounding each kernel location is sampled for the generation of the $\mathbf{C_x}$ matrix. For simplicity's sake, Figure III.I has a 5x5 kernel for sampling of the $\mathbf{C_x}$ matrix. In reality, MRV Recon allows a variable size window for generation of the $\mathbf{C_x}$ matrix. All the initial testing of Mahalanobis metric is done using a $\mathbf{C_x}$ matrix generated from a 10x10 window.

Care had to be taken to design a way to handle the varying kernel sizes. The following code fragment populates an array that contains the sample for generating the $\mathbf{C_x}$ matrix:

```
Int wsize = 5;
for ( channel = 0; channel < numIkonosChan; channel++){
     for ( j = 0; j < wsize; j++){ //whole lines down
         for ( k = 0; k < wsize; k++){ //pixels in
                         IntoCoVar[cntr] = *(image_buffrMD  + (j *
                         TwonumIkonosChan1 * num_pixels) + ( (cell + k - 2) *
                         TwonumIkonosChan1 ) + channel);
               cntr++;
         } //end of pixel loop
       }// end of line
  }//end of IntoCoVar Channel loop
```

The code fragment populates the *IntoCoVar* array with a 5x5 window. The buffer is sized within *Main* to match this window size. Other than a series of nested loops to manage the individual channels and picking out the elements for a particular kernel location, it is quite straightforward.

In contrast to straightforward handling of the memory for generating the $\mathbf{C_x}$ matrix, accessing the original and mean vectors for the user-defined kernel requires the use of an offset for the buffer:

```
buffoffset = 1 * num_pixels * TwonumIkonosChan1;
for(channel = 0; channel < numIkonosChan  ; channel++){
```

```
                XminusM[channel] =  (*(image_buffrMD + buffoffset +
                numpixintoline  + channel) - *(image_buffrMD + buffoffset +
                numpixintoline  + channel + numIkonosChan));
    }/*end of channel loop*/
```

The fragment of code above illustrates the use of the offset for the example in Figure III.I. To access the portion of code containing the data within the kernel, the pointer must be moved one whole line of data past the beginning address for *image_buffrMD*. So in the example above, the pointer address is moved, 1 line times the number of pixels per line by 6 channels per pixel (3 original channels and 3 mean channels). A loop is used to fill an array containing the difference between the original DN value and the mean DN value for a pixel. This array is the (x-m_x) vector used in calculating the Mahalanobis distance.

The PCI Toolkit provides a C function to calculate statistics for an image or a portion of an image:

```
    pixs_sampled = ImageStats(fp, numIkonosChan, dbic_list,
    bitmap, window, CoVar, Corr, means, std_dev);
```

This particular function caused a great deal of problems in getting MRV Recon running. First, the User's Guide provided for the toolkit states that if some of the statistical values are not required then just replace the vector name used to contain the statistic with a NULL [PCI, 2000, pg. 340]. This is incorrect. If a NULL value is used, the *ImageStats* function behaves erratically. In this case, the values put in the *CoVar* vector were erroneous – sometimes. This behavior made it difficult to find the problem.

In addition, the *ImageStats* function contains a small memory leak. In other words, it doesn't return to the *heap* all the memory it requested when the function is called. This caused no difficulties with smaller files used for testing MRV Recon. However, when larger files were run, “Out of Memory” errors occurred. Again, this was a troublesome bug to locate because: (1) one tends to trust the toolkit function more than their own code because presumably it has been thoroughly checked; and (2) a small memory leak is difficult to find.

Once found, another function to calculate the necessary covariance values replaced the faulty *ImageStats* function:

```
singular = gjmatinv(CoVarDbl,numIkonosChan);
mul_mtm(XminusMDbl,CoVarDbl,X_MCovar,n1,
numIkonosChan,numIkonosChan);
mul_mm(X_MCovar, XminusMDbl, MahalSqrd, n1, numIkonosChan,n1);
dist = sqrt(fabs(MahalSqrd[0])/10000);
```

The above code completes the necessary steps for calculating the Mahalanobis distance. The function *gjmatinv* inverts the variance - covariance matrix and the *mul_mtm* and the *mul_mm* multiplies a matrix by the transpose of another and multiplies two matrices together, respectively. The final Mahalanobis distance is scaled by 10,000 so as to reduce its size to fit the float data type imagery where it will be stored.

As described in Chapter 6, the Mahalanobis distance metric proved to be inadequate. The WED distance was implemented. The C code remained the same except for two minor changes. First, the function *gjmatinv* was not needed because the WED metric does not require the inversion of the variance – covariance matrix. Secondly, because the variance – covariance matrix is not being inverted, the size of the window used to generate the variance – covariance matrix can be reduced because the danger of inverting a non-singular matrix is removed.

III.II The Maximum WED

The resulting WED distances are stored in a new channel. However, because of the greater complexity in calculating the WED distance than the Minkowski, the maximum distance and outlier frequency is handled by a separate function called *MahalMax* - note the function name reflects its origins as the Mahalanobis Max function. MRV Recon only calls *MahalMax* from *Main* once, after all the WED distances are calculated.

The *MahalMax* function operates in a similar manner to the *MaxDist* function. It loops through the entire image and for each pixel in each line it selects the maximum distance. However, it differs from the *MaxDist* function in that it doesn't always increase the frequency of a particular maximum distance. Subramanian and Gat (1998) in their work tested the use of a "threshold ratio"(TR). Preliminary test results revealed that targets were being falsely identified where there was obviously no target. The TR is implemented to help reduce the false alarm rate.

$$TR = \frac{dist_{mahal} - dist_{mean}}{\sigma} \qquad \text{(III.II)}$$

where

$dist_{mahal}$ = individual distance

$dist_{mean}$ = mean of a WED distances

σ = standard deviation of pixels in the kernel

Equation (III.II) is for the threshold ratio. For a particular pixel within a user-defined kernel, the mean of all the WED distances is subtracted from the individual WED distance for that pixel. That difference is then divided by the standard deviation (σ) of the distances within the kernel.

```
for ( i = (line - half_kernel); i < ( line + half_kernel + 1); i++){
  for ( j = ( pixel - half_kernel); j < (pixel + half_kernel + 1); j++){
      numlinesdown = i * num_pixels * 2;
      numpixsintoline = j * 2;
          dist = *(image_buffrMahMax + numlinesdown + numpixsintoline);
      //THRESHOLD RATIO TEST
      if (dist > maxmahdist) {
      maxmahdist = dist;
      maxthresratio = (fabs(dist - mean) ) / stnd_dev;
      MaxLine = i;
      MaxPixel = j;
      MaxFlag = 1;
      } // end of if statement
  } // end of pixel in kernel loop
} //end of line in kernel loop
if (MaxFlag == 1 && maxthresratio > 0.5){
        numlinesdown = MaxLine * num_pixels * 2;
        numpixsintoline = MaxPixel * 2;
        //INCREMENT OUTLIER FREQUENCY
      *(image_buffrMahMax + numlinesdown + numpixsintoline + 1) =
      (*(image_buffrMahMax + numlinesdown + numpixsintoline + 1) + 1);
      } //end of MaxFlag if statement
```

The above code fragment implements the TR within MRV Recon. The code loops through a particular kernel testing each distance with all the others until the maximum WED distance is found. If a maximum distance is found, the location of the maximum distance is recorded in the *MaxLine* and *MaxPixel*. If the TR value (assigned to the variable *maxthresratio)* is greater than 0.5, only then will the outlier frequency be implemented at the location of the maximum distance. The outlier frequencies are stored within a new channel in the image file. Although, Subramaniam and Gat [1998] reported using a value of 1.0 for the TR value, it was found to be too high and excluded many targets. After many attempts, a value of 0.5 for the TR was found to achieve the best results.

III.III Outlier Controls

At the outset of this project, the idea was not just to implement the work of Iverson [1997]) and Subramanian and Gat [1998] but to adapt it for use with imagery such as IKONOS and enhance its ability to detect small targets. The enhancement is not limited to the use of the WED distance metric, but also the implementation of various thresholds, weightings, frequency limits, and threshold ratios. Collectively, these are known as “outlier controls” indicating their purpose of improving detection rate of MRV Recon while limiting the number of false positives.

Figure III.III illustrates the various outlier controls used in MRV Recon beginning with the user-defined kernel size. The user selects a kernel size keeping in mind that the larger the kernel, the greater the number of pixels that the currently investigated pixel will be compared to when determining if it is an outlier. This makes it harder to stand out as distinct from its neighbours. The converse is also true; the smaller the kernel size, the easier it is to stand out. However, this means possibly allowing more false positives.

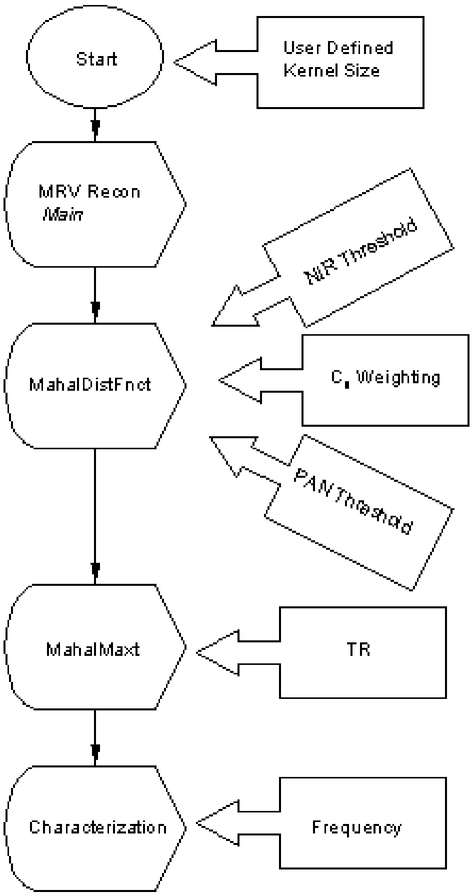


Figure III.II MRV Recon Outlier Controls

As shown in Figure III.II, the *MahalDistFnct* function contains three outlier controls. The thresholding of the near infrared channels (NIR) and the panchromatic (PAN) channels are very similar. Once a WED distance is calculated for a pixel, it is only retained for use if the corresponding values of the NIR and PAN are above imperically determined thresholds.

$$T_{NIR} = 1.65 * m_{NIR} \tag{III.III}$$

$$T_{PAN} = 2.70 * m_{PAN} \qquad \text{(III.IV)}$$

where:

m_{NIR} = mean of the NIR channel (excluding the background pixels)

m_{PAN} = mean of the PAN channel (excluding the background pixels)

The PAN and NIR thresholds are shown in the code below:

```
*(nirthrhld + 2) = *(nirthrhld)/(*(nirthrhld + 1)) * 1.65;
*(panthrhld + 2) = *(panthrhld)/(*(nirthrhld + 1)) * 2.7;
if( *(image_buffrMD + buffoffset + numpixintoline  + 4) > (*(nirthrhld + 2))
&& (*(image_buffrMD + buffoffset + numpixintoline) > (*(panthrhld + 2))))
*(image_buffrMD + buffoffset + numpixintoline  + TwonumIkonosChan1 - 1 )
= dist;
```

As previously stated, water in general exhibits low NIR values. The NIR threshold is used to minimize the number of background water pixels selected as an outlier. Similarly, the PAN threshold is used to restrict only those pixels that have a very high NIR value as being labeled an outlier. It was found that a pixel could be a local maximum driven by slightly higher NIR and PAN values, but not high enough to be a target. Invoking the PAN threshold lowers the chances of allowing a non-target pixel to be falsely labeled. The threshold values are generated from the mean values of the NIR and PAN channels and are weighted by 1.65 and 2.7, respectfully. These weighting were developed when testing the St. John River dataset.

The following fragment of C code invokes the third outlier control from within the *MahalDistFnct*.

```
CoVarDbl[0] = CoVarDbl[0] * 3; //pan
CoVarDbl[24] = CoVarDbl[24] * 2;//NIR
```

This third control weights the variance values generated for the PAN and NIR channels within the $\mathbf{C_x}$ matrix.

$$\boldsymbol{W_{PAN} = 3\sigma_{PAN}^2} \tag{III.V}$$

$$\boldsymbol{W_{NIR} = 2\sigma_{NIR}^2} \tag{III.VI}$$

where

W is the weight

$\boldsymbol{\sigma}$ = standard deviation

PAN = panchromatic band

NIR = near infrared band

The testing has shown the NIR and PAN channels carry the greatest variance or information. By further accentuating the weighting of these two channels, pixels with high PAN and NIR values will produce a higher WED distance due to the weighting. This ensures that these pixels stand out amongst their neighbours and are labeled as outliers.

In describing the *MahalMax* function, the next outlier control shown in Figure III.II, was outlined. The TR or threshold ratio as was described in section III.2 also controls outliers.

The final control resides within the *CharacterFnct*. The characterization function investigates the outliers having a high frequency and then labels them as targets. Once labeled, target attributes are gathered automatically within the CharacterFnct.

```
//frequency tests
if ( *(image_buffrCharactr + numlinesdown + numpixsintoline + 1) >
((kernel_size * kernel_size) - 1)){
    // any pixel making it to here is to be labelled
    // begin search
    //push the pixel onto the stack
```

```
first_targetcoord.pixel[0] = pixel;
first_targetcoord.line[0] = line;
push (&targets_stack, first_targetcoord);
```

The above code fragment ensures that only pixels having a very high outlier frequency namely ((*kernel_size * kernel_size) - 1*)) are labeled as a target. Lowering the minimum allowable frequency makes it easier for false positives to be labeled as targets. Conversely, targets that are difficult to detect can be positively detected by lowering the minimal allowable frequency. Again, this is at the expense of increasing the number of false positives. A balance must be struck between ability to detect small targets and the number of allowable false positives.

III.IV Characterization Function

After a vessel has been detected, data must be gathered regarding its "characteristics". To clarify the distinction between gathering characteristics and traditional spectral-based classification techniques, the function that performs this task is referred to as the *characterization* function. Central to the *characterization* function is the use of a "stack" structure. The stack is an array of structures. Each element of the structure is composed of a pixel's attributes and its element number in the stack array is its target label.

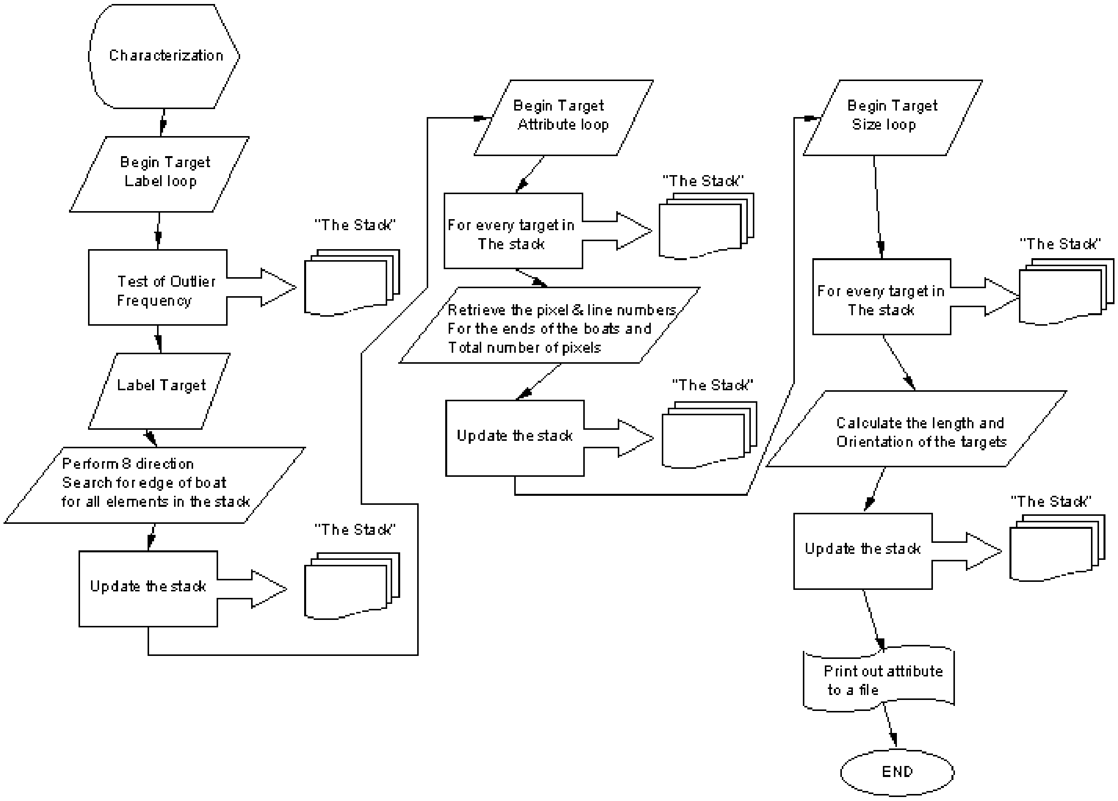

Characterization
Begin Target Label loop
Test of Outlier Frequency
"The Stack"
Label Target
Perform 8 direction Search for edge of boat for all elements in the stack
"The Stack"
Update the stack
Begin Target Attribute loop
For every target in The stack
"The Stack"
Retrieve the pixel & line numbers For the ends of the boats and Total number of pixels
Update the stack
"The Stack"
Begin Target Size loop
For every target in The stack
"The Stack"
Calculate the length and Orientation of the targets
Update the stack
"The Stack"
Print out attribute to a file
END

Figure III.III The Characterization Function.

Figure III.III illustrates the major components of the *CharacterFnct.* It is composed of three primary loops: the *Target Label Loop*, the *Target Attribute Loop*, and the *Target Size Loop*.

The *Target Label Loop* begins, as previously described, with a test of the outlier frequency. If the frequency is higher than the minimal allowable frequency, the pixel is labeled as a target. Next the pixel and line coordinates, along with its label, are put onto the stack.

A suitable analogy for the stack would be a spring-loaded dinner plate dispenser commonly found in cafeterias. As plates are added the spring compresses due to the weight. As a patron removes a plate, the spring lifts the next plate upwards making retrieval convenient for the next patron.

The stack works much in the same way. Instead of plates however, there are linked memory storage addresses containing the attribute data for each labeled target.

```
        first_targetcoord.pixel[0] = pixel;
        first_targetcoord.line[0] = line;
        push (&targets_stack, first_targetcoord);
```

The above portion of the *CharacterFnct* illustrates the use of the stack. The current pixel and line values are stored in the defined C structure called *first_targetcoord*. The function *push* then "pushes" the structure (pixel and line values) onto the top of the stack. A similar function *pop* takes the top element off the stack. These functions are used to add and remove elements to and from the stack so that it is always kept current with respect to the labeled targets.

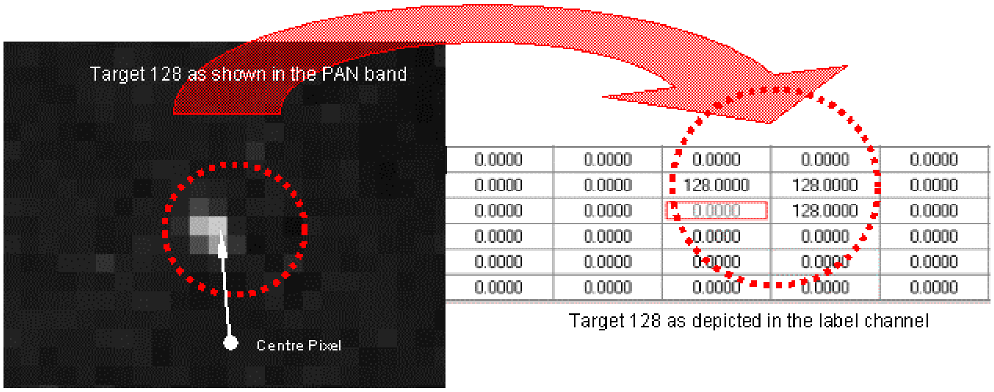
Once the target is labeled and put onto the stack, a continuous loop is constructed to interrogate each element of the stack. The loop is only broken when the bottom of the stack is reached. The bottom is marked with a special end of data value. This way, every element is certain to be processed.

Beginning the actual investigation for a target, an 8-way search is performed from the centre of each target. The function keeps labeling adjacent pixels with the centre pixel's target number, until the edge of the boat is found. The boat's edge is deemed to be found when the panchromatic value of the pixel falls below a threshold:

```
    thrshold = meen[0] + (4 * sigma[0]);
        if (*(image_buffrCharactr + numlinesdownb + numpixsintolineb) < thrshold
```

The above code fragment shows that the edge of boat threshold is set at $\mu+4\sigma$. If the PAN value falls below this threshold then the labeling ends and the search begins for the edge of the boat in the next of the eight directions. If all eight directions are done, the next target is “popped” off the stack.

The second of the three loops shown in Figure III.III gathers the target attributes. For each target on the stack, the previous loop essentially performs a region growing function. The next loop seeks out and records the minimum and maximum pixel and line numbers for a target.


Target 128 as shown in the PAN band
Centre Pixel
Target 128 as depicted in the label channel

0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	128.0000	128.0000	0.0000
0.0000	0.0000	0.0000	128.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000

Figure III.IV Example of Labeling Target # 128.

For example, Figure III.IV illustrates the target labeling for a small target. On the left hand side of Figure III.IV is the panchromatic band for target number 128. Beginning at the centre pixel the eight-way search is performed and the region representing the boat is grown. On the right hand side of Figure III.IV the results of the labeling routine can be seen. The second loop finds the extreme pixel and line coordinates for the target region.

```
//SEEK THE OTHER END OF THE BOAT
the_target_attributes[i].lrgpixel[0] = pixel;
the_target_attributes[i].lrgline[0] = line;
the_target_attributes[i].numbrpixs[0] = j;
```

The above code fragment places the largest pixel and line coordinate, as well as the total number of pixels within the region, into *the_target_attributes[]* array. Again, the array element number corresponds to the target number.

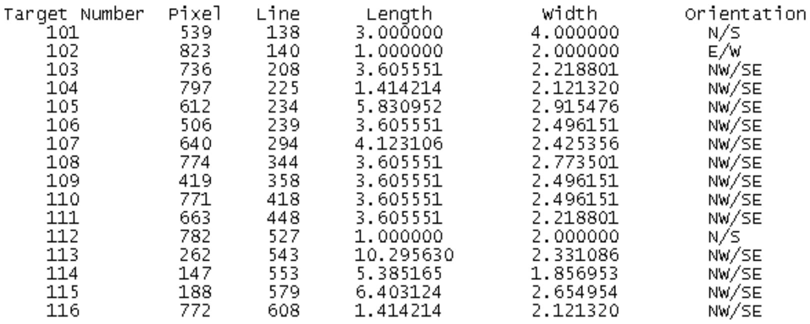
The final loop in the *characterization* function, shown in Figure III.III, calculates the dimensions of the target using the values in the *the_target_attributes[]* array.

```
deltapix = pow (the_target_attributes[i].lrgpixel[0] -
the_target_attributes[i].smpixel[0], 2);
deltaline = pow (the_target_attributes[i].lrgline[0] -
the_target_attributes[i].smline[0], 2);
the_target_attributes[i].length[0] = sqrt (deltapix + deltaline);
```

The code fragment above shows how the largest and smallest pixel values are retrieved from *the_target_attributes[]* array. Using Pythagorean theorem, the length is then calculated and put into *the_target_attributes[].length[0]* structure.

The *characterization* function also makes use of the *the_target_attributes[] array* to determine the orientation of the target by finding the longest axis of the target region and determining its aspect. The orientation is distinct from heading. Heading indicates a direction of travel. Orientation only reveals the direction the longest axis of the vessel.

The final step in the c*haracterization* function is the printing out to a text file each of the targets and its corresponding attributes. Figure III.V shows the text output of MRV Recon.



```
Target Number  Pixel   Line     Length          Width        Orientation
     101        539     138    3.000000        4.000000         N/S
     102        823     140    1.000000        2.000000         E/W
     103        736     208    3.605551        2.218801         NW/SE
     104        797     225    1.414214        2.121320         NW/SE
     105        612     234    5.830952        2.915476         NW/SE
     106        506     239    3.605551        2.496151         NW/SE
     107        640     294    4.123106        2.425356         NW/SE
     108        774     344    3.605551        2.773501         NW/SE
     109        419     358    3.605551        2.496151         NW/SE
     110        771     418    3.605551        2.496151         NW/SE
     111        663     448    3.605551        2.218801         NW/SE
     112        782     527    1.000000        2.000000         N/S
     113        262     543    10.295630       2.331086         NW/SE
     114        147     553    5.385165        1.856953         NW/SE
     115        188     579    6.403124        2.654954         NW/SE
     116        772     608    1.414214        2.121320         NW/SE
```

Figure III.V MRV Recon Characterization function output.

III.V Summary

This chapter describes the programming of enhancements made to the spatio-spectral template. The implementation of first the Mahalanobis and then the WED metrics are described. In particular, problems surrounding the erratic behavior of the *ImageStats* function from the PCI C Toolkit and its replacement with another function are described.

The chapter explains the function that selects the maximum WED distance within a kernel. In addition, various outlier controls are described. These controls help to minimize the number of false positives generated by MRV Recon. Finally, the characterization function that automatically generates the attributes of the targets is explained.

Vita

Kevin Huntly Pegler, P.Eng.

Place and Date of Birth:
King City, Ontario
Canada
19 April 1961

Home Address:
175 Topcliffe Crescent
Fredericton, N.B.
Canada E3B 4P8

EDUCATION

Diploma, University Teaching
Faculty of Education
University of New Brunswick, 2002.

Master of Engineering,
Dept. Geodesy and Geomatics Engineering
University of New Brunswick, 1999.

Graduate Diploma, Remote Sensing
College of Geographic Sciences (COGS), 1996.

Diploma, Geographic Information Systems, (Honours)
Algonquin College, 1995.

Bachelor of Technology, Surveying Engineering
Dept. of Civil Engineering
Ryerson Polytechnic University, 1994.

PUBLICATIONS

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang, [2004]. *Comparison of maximum distance metrics for use in the remote sensing of small targets*, Journal of Surveying Engineering, in press.

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang, [2003]. *The potential for using very high spatial resolution imagery for marine search and rescue surveillance*, GeoCarto International, 18(3):35-39.

Rancourt, M. Captain, K. Pegler, D.J. Coleman, R. Pelot [2002]. “Development of an IKONOS Coverage Prediction Application”. Proceedings of The 95th CIG Annual Geomatics Conference / Joint International Symposium on Geospatial Theory, Processing and Applications. Ottawa, Jul 8-12.

Pegler, K., D.J. Coleman, R. Pelot, and Y. Zhang [2002]. “IKONOS Sub-pixel Target Detection for Use in Marine Search and Rescue” *Proceedings of the XXII FIG International Congress and ACSM-ASPRS Conference and Technology Exhibition*, 2002,Washington, D.C.

Pegler, K. and D.J. Coleman [2002]. “Quality Control Issues Surrounding the Reprocessing of Digital Terrain Models”. *GIM*. Vol. 16, March, pp. 12-15.

Pegler, K. [2001]. “An Examination of Alternative Compensation Methods for the Removal of the Ridging Effect from Digital Terrain Model Data Files”. Department of Geodesy and Geomatics Engineering Technical Report No. 209, University of New Brunswick, Fredericton, N.B., Canada.

Pegler K. and D.J. Coleman [2001]. “New Brunswick Technical Specifications for Minimization of the Ridging Effect in Service New Brunswick Digital Terrain Models.” Contract Report prepared for Service New Brunswick, Fredericton, N.B. March.

Pegler, K., D.J. Coleman, R. Castonguay, and H. Nguyen [2000]. “Comparing TIN Random Densification with the Mean Profile Filter to Minimize the Ridging Phenomenon in Service New Brunswick DTMs”. *Geomatica,* Vol. 54, No.4, pp 433-440.

Pegler, K. and D.J. Coleman [1999]. “Removal of Ridging Effects from Digital Terrain Model Data Files: Examining the Alternatives”. *Proceedings of the 1999 Annual Conference of the Urban and Regional Information Systems Association*. Chicago.

Pegler K. and D.J. Coleman [1999]. "Investigation of Alternative Approaches to Systematic Removal of Ridging Effect in New Brunswick Digital Terrain Model Products". Contract Report prepared for Service New Brunswick, Fredericton, N.B., Canada. April.

Pegler, K. [1999]. “TIN Random Densification: A Process to Minimize the Ridging Phenomenon in DTMs.” Unpublished thesis. Department of Geodesy and Geomatics Engineering, University of New Brunswick, Fredericton, N.B., Canada.

Pegler, K. [1987]. “Volume Calculations on an Arc Basis.” Unpublished thesis. Department of Survey Engineering, Ryerson PolytechnicUniversity, Toronto, Ont., Canada.

PRESENTATIONS

Atlantic Institute, “A Marine Recreational Vessel Reconnaissance System”, Fredericton, New Brunswick. June, 2004.

Society of Optical Engineering, “Comparison of Distance Metrics for Use Within a Marine Recreational Vessel Reconnaissance System”. San Diego, California. August, 2003.

GEOIDE Annual Meeting, “A Marine Recreational Vessel Reconnaissance System.” Victoria, British Columbia. May, 2003.

Maritime Safety and Risk Analysis Conference, “A Marine Recreational Vessel Reconnaissance System.” Halifax, Nova Scotia. March, 2003.

GEOIDE Annual Meeting, “Sub-Pixel Target Detection for Marine Search and Rescue.” Toronto, Ontario. May, 2002.

XXII FIG & ASPRS Annual Conference, “IKONOS Sub-pixel Target Detection for Marine Search and Rescue”. Washington, D.C. April, 2002.

Annual Conference of the Urban and Regional Information Systems Association, Removal of Ridging Effects from Digital Terrain Model Data Files: Examining the Alternatives” Chicago, Illinois, August, 1999.

Atlantic Institute, “Alternative Approaches to Systematic Removal of Ridging Effect in New Brunswick Digital Terrain Model Products”, Quebec City, Quebec. June, 1999.