12-17-2004

Algorithms and Data Structures for Automated Change Detection and Classification of Sidescan Sonar Imagery

Marlin Gendron
University of New Orleans

ALGORITHMS AND DATA STRUCTURES FOR AUTOMATED CHANGE DETECTION AND CLASSIFICATION OF SIDESCAN SONAR IMAGERY

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Engineering & Applied Science

by

Marlin Lee Gendron

M.S., University of New Orleans, 1999

December 2004

Acknowledgements

I would like to thank all the people who made this dissertation possible. If it were not for the guidance of my committee, the patience of my family, and the willingness of my co-authors to listen to my off-the-wall ideas, this document would not exist. My committee members took the time out of their busy schedules to help. I am especially grateful to my main advisor Dr. Juliette Ioup and my advisors, Dr. Maria Kalcic, Dr. George Ioup, Dr. Greg Seab, and Dr. Brian and Edit Bourgeois for their many suggestions that kept me up late at night, but made this document much better.

I would like to say sorry to my eleven-year-old son Blake, who I could not take fishing many times. Now that this document is done, I think Blake and I will hang the "Gone Fishing" sign out. We will certainly take my girlfriend Megin. I appreciate her support more than she will ever know. I also wish to thank my Dad, and I regret that my Mom is not here to see me graduate.

I have to find some way to thank my co-authors. Without Maura's clear thinking and her excellent editing skills, this dissertation would be a very confusing document for sure. I especially would like to thank Geary Layne. His "completion" algorithm rocks! He almost has enough material for his own dissertation.

And finally, backing up even further, none of this would have been possible if I had not passed my general exams. I could have not passed without help from my study partner. I will never know how to repay Pam McDowell.

Table of Contents

List of Figures vi
List of Tables ix
Abstract x
Chapter 1 – Introduction 1
Acronyms 1
Change Detection 1
Background 2
Sidescan Sonar System (SSS) 4
Sidescan Theory 6
Klein SSS 9
Problems with Sidescan 10
ACDC System 11
Dissertation Chapters 13
Chapter 2 – The Geospatial Bitmap (GB) 13
Chapter 3 – The Geospatial Bitmap (GB) Clustering Algorithm 13
Chapter 4 – A Real-Time Computer-Aided Detection (CAD) Algorithm 13
Chapter 5 – A Computer-Aided Search (CAS) Algorithm 14
Chapter 6 - Conclusion 14
References 15
Chapter 2 - The Geospatial Bitmap (GB) 18
Abstract 18
Introduction 18
Terminology 19
GB Evolution 21
First Generation GBs 22
Second Generation GBs 24
Third Generation GBs 26
Fourth Generation GBs 28
GB Operations 31
Experimental Runs 33
Euclidean Geometry Method 33
Sparse Matrices (SM) Data Structure 34
Methods 35
Results 36
Conclusion 38
Acknowledgements 38
References 38
Chapter 3 - The Geospatial Bitmap (GB) Clustering Algorithm 41
Abstract 41
Introduction 41
Geospatial Bitmaps 42

The GB Clustering Algorithm 43
2D Example in Cartesian Coordinate Space 45
Experimental Runs 49
Methods 49
Results 49
Clustering Demonstration 55
Conclusion 55
Acknowledgements 56
References 56
Chapter 4 - A Real-time Computer-Aided Detection (CAD) Algorithm 58
Abstract 58
Acronyms 58
Introduction 59
Sidescan Sonar Imagery (SSI) 61
Real-Time CAD Algorithm 63
Across-Track Processing 63
Scan line Shadow/Bright Thresholding 65
Detecting Objects 68
Generating Snippet in the Mini-UNISIPS (MU) Format 70
Testing the CAD Algorithm 70
Examples of Detected MILECs 74
Conclusion 76
Acknowledgements 77
References 77
Chapter 5 - A Computer-Aided Search (CAS) Algorithm 80
Abstract 80
Acronyms 80
Introduction 81
Position Error in Sidescan Sonar Imagery (SSI) 83
Data Structures 86
Geospatial Bitmaps (GBs) 88
Quadtrees 88
Modified Quadtrees (MTREEs) 90
Data Loading 92
Searching for Spatially Close Features 93
Experimental Runs 95
Methods 95
Results 96
Conclusions 97
Acknowledgements 98
References 98
Chapter 6 – Conclusion 100
Acronyms 100
Introduction 101
The ACDC System 102
Detection Stage 102

Classification Stage 103
Search Stage 106
Graphic Information System (GIS) Display 108
Future Work 108
References 109
Vita 114

List of Figures

Figure 1-1. Sidescan Sonar System (Blondel and Murton, 1997) 5
Figure 1-2. Depiction showing a sidescan sonar system (SSS) imaging over the sea bottom. 7
Figure 1-3. Sidescan Sonar Imagery (SSI) 8
Figure 1-4. Snippet of a MILEC in SSI 9
Figure 1-5. Klein 5000 Sidescan Sonar 9
Figure 1-6. Tow Cable 10
Figure 1-7. Cable Layback Model 11
Figure 1-8. ACDC System 12
Figure 1-9 The Author's Real-Time Computer-Aided Detection (CAD) Algorithm 13
Figure 1-10. Computer-Aided Search Query 14
Figure 1-11. Results from the Completion Algorithm 15
Figure 2-1. Sample bitmap with 1 bit per pixel. 21
Figure 2-2. First generation geospatial bitmap. 23
Figure 2-3. Wasted Memory Example 24
Figure 2-4. Bit-masking (reverse endian) 25
Figure 2-5. Second Generation GB 26
Figure 2-6 . GB broken into strip-like submaps. In this example, NR = 29, NC = 30, p = 22, n = 5, e = 0, and overall savings = 42.75 bytes. 27
Figure 2-7. Tiled GB (Note: There are also 4 bits of wasted space per tile that are not shown in the figure) 29
Figure 2-8. Polyline Smoothing Example 32
Figure 2-9. ANDing in GB Space 32
Figure 2-10 . Sparse Matrix (SM) Data Structure 34
Figure 2-11. Time (in seconds) required by all algorithms to smooth polygons resulting from the four data sets with a clustering radius of 50 meters. 37
Figure 2-12. Time (in seconds) required by all algorithms to smooth polygons resulting from the four data sets with a clustering radius of 125 meters. 37
Figure 3-1. Example of a 2D bitmap. 44
Figure 3-2. Collection of points in 2D geographic space. 45
Figure 3-3. Points mapped to GB: each bit holding a point is set; other bits are cleared. 45
Figure 3-4. Example of expansion shape that is formed around each element in the data set. 46
Figure 3-5. Size of GB1 is determined by the MBR of the expansion of all points to be clustered. 46
Figure 3-6. Points that are geographically close to each other will cluster together. 46
Figure 3-7. Starting at upper-left corner bit, traverse each cluster (in a consistent direction) to obtain vertices (in blue). 46
Figure 3-8. Testing whether there are two adjacent clusters or a single cluster: a) two clusters with 4 vertices each; b) two clusters with 4 vertices each (one vertex is shared between the two clusters); c) one cluster with 8 vertices. 48
Figure 3-9. Convert bitmaps to vector boundaries 48

Figure 3-10. Smooth and simplify cluster boundaries by dropping vertices if the resulting polygon still contains all the original points and has an area equal to or less than the unsmoothed boundary. In this example, a recursive polyline-smoothing algorithm drops points and achieves maximum simplification of the clusters after four iterations (a through d). 48
Figure 3-11. Bivariate fit of time (**t**) by #elements to be clustered (**n**). 50
Figure 3-12. Bivariate fit of time (**t**) by expansion area (**a**). 51
Figure 3-13. Bivariate fit of time (**t**) by data set resolution (**r**) in bits/element. 51
Figure 3-14. One-way analysis of time (**t**) by expansion shape (**s**): C = circle, S = square..... 52
Figure3- 15. One-way analysis of time (**t**) by element distribution (**d**): D=distinct, U=uniform. 52
Figure 3-16. One-way analysis of time (**t**) by ordering (**o**) of elements to algorithm: A=alternating, R=random, S=sequential. 53
Figure 3-17. Model of processing time (**t**) as a function of #elements (**n**), expansion area (**a**), data set resolution (**r)**, expansion shape (**s**), and significant interactions among these variables. Note that shape (**s**) on its own does not have a significant effect on **t**. 54
Figure 3-18. Manual Clustering 55
Figure 3-19. Automated Clustering 55
Figure 4-1. Illustration of a towed sidescan sonars system 60
with resulting sidescan sonar imagery 60
Figure 4-2. Sidescan sonar imagery 61
Figure 4-3. Mine-like Echo detected in sidescan imagery 62
Figure 4-4. Gamma as a function of β 66
Figure 4-5. GBs are used to facilitate computer-aided detection of objects in SSI. Each row of bits in both GBs corresponds to a single scan line in the image. All pixels in the image with an intensity value greater than an upper threshold are considered “brights” and the appropriate bit in the bright GB is set. Likewise, all pixels in the image with an intensity value less than a lower threshold are considered “shadows” and the appropriate bit in the shadow GB is set. 67
Figure 4-6. Intensity thresholds for brights and shadows. 68
Figure 4-7. Momentum is used to determine how many cleared bits to include in a shadow or bright GB. 70
Figure 4-8. Dobeck’s Fusion method increases correct detections and reduces false alarms, compared with each individual algorithm (figure 5 from Dobeck, 2004). 71
Figure 4-9. The author’s real-time CAD algorithm produced detection and false alarm rates similar to the individual algorithms tested in Dobeck (2004). Combining the real-time CAD with the LCM resulted in greater correct detection rates and fewer false alarms, with results similar to Dobeck’s Fusion method. 72
Figure 4-10. Snippet of cylindrical MILEC. 75
Figure 4-12. MILEC with rectangular shadow. 75
Figure 4-11. MILEC partially buried in sand. 75
Figure 4-13. MILEC with quadrilateral shadow. 75
Figure 4-14. False detection: schools of fish. 76
Figure 4-16. False detection: rock 76
Figure 4-15. False detection: objects too small. 76
Figure 4-17. False detection: surface return. 76

Figure 5-1. Snippet of Mine-like Echo 82
Figure 5-2. Klein 5000 towfish 83
Figure 5-3. Sidescan Sonar Imagery (SSI) 84
Figure 5-4. Estimated lat/lon position of an object on a scan line. 85
Figure 5-5. Location of the towfish (Klein Associates, Incorporated, 2004). 86
Figure 5-6. Geospatial Bitmap (GB) 88
Figure 5-7. This is a 6-Level quadtree where each level contains four children nodes except for level 0. Level 0 is the root node which contains only a memory pointer to the four children nodes of level 1. 89
Figure 5-8. Four levels of a sample quadtree. 89
Figure 5-9. Look-Ahead GB 91
Figure 5-10. Step 1 searching returns the location of one historical MILCO, H_{10}, because it falls within the uncertainty error region of the recently detected MILCO, N_1. 94
Figure 5-11. If the FM process did not determine that H_{10} was the same as N_1, then step 2 searching is performed, determining that both H_3 and H_{10} are spatially close to N_1 because their error regions overlap the error region of N_1. 94
Figure 5-12. All MTREE leaf nodes falling within error ellipse for the feature of interest. 95
Figure 5-13. Average time (in seconds) that each method took to spatially search the data sets. Results were averaged for error radii of 5m, 15m and 25m. 97
Figure 6-1. Detection Stage 103
Figure 6-2. Sample echo snippets (left) along with their filtered echo images (right). 105
Figure 6-3. Classification Stage 106
Figure 6-4. Search stage of ACDC 106
Figure 6-5. Identification Stage of ACDC 107
Figure 6-6. All components of ACDC are tied together in GIS. 108

List of Tables

Table 2-1. Acronyms, Terms and Symbols 20
Table 2-2. Test results of smoothing with GB, Euclidean1, Euclidean2, SM1, and SM2 (Cluster Radius of 50 meters). 36
Table 2-3. Test results of smoothing with GB, Euclidean1, Euclidean2, SM1, and SM2 (Cluster Radius of 125 meters). 36
Table 3-1. GB clustering variables tested for their impact on processing time. 49
Table 4-1. Results from 25 runs of the CAD algorithm using the same UNISIPS files while adjusting parameters A and B (which control how the lower and upper shadow/bright thresholds are computed). 73
Table 4-2. Results after running LCM to reduce the false alarm rate. 74
Table 5-1. Results from 5 meter error region 96
Table 5-2. Results from 15 meter error region 96
Table 5-3. Results from 25 meter error region 96

Abstract

During Mine Warfare (MIW) operations, MIW analysts perform change detection by visually comparing historical sidescan sonar imagery (SSI) collected by a sidescan sonar with recently collected SSI in an attempt to identify objects (which might be explosive mines) placed at sea since the last time the area was surveyed. This dissertation presents a data structure and three algorithms, developed by the author, that are part of an automated change detection and classification (ACDC) system. MIW analysts at the Naval Oceanographic Office, to reduce the amount of time to perform change detection, are currently using ACDC**.** The dissertation introductory chapter gives background information on change detection, ACDC, and describes how SSI is produced from raw sonar data. Chapter 2 presents the author's Geospatial Bitmap (GB) data structure, which is capable of storing information geographically and is utilized by the three algorithms. This chapter shows that a GB data structure used in a polygon-smoothing algorithm ran between 1.3 – 48.4x faster than a sparse matrix data structure. Chapter 3 describes the GB clustering algorithm, which is the author's repeatable, order-independent method for clustering. Results from tests performed in this chapter show that the time to cluster a set of points is not affected by the distribution or the order of the points. In Chapter 4, the author presents his real-time computer-aided detection (CAD) algorithm that automatically detects mine-like objects on the seafloor in SSI. The author ran his GB-based CAD algorithm on real SSI data, and results of these tests indicate that his real-time CAD algorithm performs comparably to or better than other non-real-time CAD algorithms. The author presents his computer-aided search (CAS) algorithm in Chapter 5. CAS helps MIW analysts locate mine-like features that are geospatially close to previously detected features. A comparison between the CAS and a great circle distance algorithm shows that the CAS performs geospatial searching 1.75x faster on large data sets. Finally, the concluding chapter of this dissertation gives important details on how the completed ACDC system will function, and discusses the author's future research to develop additional algorithms and data structures for ACDC.

Chapter 1 – Introduction

Marlin L. Gendron, Maura C. Lohrenz and Geary J. Layne
Naval Research Laboratory (NRL) Code 7440.1, Stennis Space Center, MS

Juliette Ioup
University of New Orleans (UNO), New Orleans, LA

Acronyms

ACDC	automatic change detection and classification
ADC	analog to digital converter
AM	area matching
AUV	autonomous underwater vehicles
CAC	computer-aided classification
CAD	computer-aided detection
CAS	computer-aided search
FM	feature matching
GB	geospatial bitmap
GPS	global positioning systems
INS	inertial navigation systems
MCDB	master contact database
MILEC	mine-like echo
MLO	mine-like object
MIT	Massachusetts Institute of Technology
MIW	Mine Warfare
NAVOCEANO	Naval Oceanographic Office
NOMBOS	nonmine, minelike bottom object
NRL	Naval Research Laboratory
ONR	Office of Naval Research
PNT	positioning, navigation and timing
RAC	Research Advisory Committee
SPAWAR	Space and Naval Warfare System Command
TDA	Tactical Decision Aid
SSI	sidescan sonar imagery
SSS	sidescan sonar system
UNISIPS	Unified Sonar Image Processing System
UNO	University of New Orleans

Change Detection

Change detection is the process of identifying differences in the state of an object or phenomenon by observing it at different times (Deer, 1995). The term "digital change detection"

applies when computer algorithms are used to perform related change detection tasks commonly on data collected by a remote sensing device (Singh, 1989). Digital change detection is a major application of remote-sensed data (Rosin, 1994).

An area that benefits from change detection on data from a remote sensing device is mine warfare. During military Mine Warfare (MIW) operations to clear assault lanes for naval ships to pass through safely, analysts perform change detection by visually comparing historical high resolution imagery collected by a sidescan sonar system (SSS) with recently collected sidescan sonar imagery (SSI) in an attempt to identify objects (that might be explosive mines) placed at sea since the last time the area was surveyed. The objective of change detection in MIW is to match new features detected in SSI with historical features that analysts have placed in a database and divers, or autonomous machines, have visually investigated and confirmed. Change detection using SSI is potentially a significant time saving tool, but problems such as large data volume, navigational errors, sonar towfish instabilities, differences in look-angle and differences in environmental conditions can hinder the time savings (Lingsch and Lingsch, 2001).

There is an abundance of literature showing the success of the digital change detection on satellite imagery, so only a few are listed here (Howarth and Wickware, 1981; Griffiths, 1988; Stow et al. 1990; Banner, 1991; Lambin and Strahler, 1994). Literature is sparse on digital change detection techniques for SSI, although ground-breaking research by the author at the Naval Research Laboratory in collaboration with the University of New Orleans (UNO) is being conducted (McDowell et al., 2003; Ioup et al., 2003, Ioup et al., 2004). Results from this research, and this dissertation, show that automated digital change detection techniques applied to SSI could greatly reduce the time it takes analysts to perform manual change detection. Faster change detection will reduce mine clearance timelines and allow MIW operators to accurately assess the risk to follow-on naval forces.

Furthermore, it is believed that several digital change detection algorithms and data structures can be combined into one Automated Change Detection and Classification (ACDC) system. ACDC, being developed by the author, will be capable of aiding analysts to detect seafloor features in SSI collected by SSS, classify, and catalog said features. These features are then compared with ones “seen” in the past (in stored historical data) to determine if the features have moved or are new.

When implemented, ACDC will not be fully autonomous. ACDC is not meant to perform change detection without humans, but rather, enable analysts to perform change detection on SSI more efficiently. Digital change detection techniques are useful tools to assist human analysts, and can be useful as a 'cueing system' to attract the attention of human analysts to 'interesting' images, but further considerable effort is required to produce fully autonomous change detection systems (Deer, 1995).

Background

This dissertation contains this introductory chapter, a conclusion chapter, and four main chapters. The main chapters are written as separate journal articles and each describes one of four

components, previously developed by the author, which are critical in the development of an ACDC system for SSI. The four components presented are: 1) The Geospatial Bitmap (GB), 2) The GB Clustering Algorithm, 3) A Real-time Computer-Aided Detection (CAD) Algorithm, and 4) A Computer-Aided Search (CAS) Algorithm.

The remainder of this introductory chapter gives some background on ACDC, some basic sidescan sonar theory, and describes how SSI is produced from raw sonar data. The difficulties of using SSI to perform change detection will be described, and finally, a detailed look at the ACDC system and a short breakdown of the main chapters will be presented. The conclusion chapter will give some additional results and discuss future research by the author needed to complete and improve ACDC.

The author of this dissertation is the first author of each chapter and has contributed to the majority of the writing and research. Co-authors are listed in the chapters and a letter from each co-author has been submitted to the Graduate School Dean for approval.

Portions of this introductory chapter have been published in the Proceedings of the Sixth International Symposium on Technology and the Mine Problem (Gendron et al., 2004b). The work presented in Chapter 2, "The Geospatial Bitmap (GB)", was patented in 2001 (Gendron et al., 2001). Chapter 3, "The Geospatial Bitmap (GB) Clustering Algorithm", has been submitted to the Journal of Discrete Algorithms (Gendron et al., *submitted*), and the Naval Research Laboratory (NRL) patent attorney prepared and submitted a patent application to the United States Patent Office (Gendron et al., 2004a). The remaining chapters have not been submitted to journals for publication to date, but it is the full intention of the author to submit them in the near future.

Work in this research area began at NRL by the author in 1999 with funding from the Space and Naval Warfare System Command (SPAWAR) to automatically detect and cluster seafloor objects surveyed with SSS. An autonomous clustering algorithm and a fully autonomous, real-time clutter detection algorithm were completed and transitioned to the Naval Oceanographic Office (NAVOCEANO) in 2001. Both of these algorithms have been enhanced and modified to work in the ACDC system and are described in Chapters 3 and 4, respectively.

Late in 2001, Dr. Brian Bourgeois from NRL Stennis Space Center briefed the Research Advisory Committee (RAC) at NRL in Washington D.C. on a proposal for 6.2 Applied Research funding to research an underwater positioning, navigation, and timing (PNT) system for multiple autonomous underwater vehicles (AUVs). The objective of the research is to provide enabling technologies to improve the task force reference position and allow inter-AUV positioning and navigation. The objectives for reference position improvement include feature-based, or terrain matching, positioning using SSS.

The 6.2 funding was approved in 2002, and the author began researching methods to "Improve AUV Navigation by Extracting and Matching Bottom Features in Sidescan Imagery". The premise is that an AUV would have an on-board feature database populated with features detected in historical SSI. The autonomous algorithms aboard the AUV would detect new features in real-time, match individual features with features in the on-board database, and use

this information to correct its position. To date, under this funding, a feature database for fast spatial searching has been researched and designed. The author's CAS algorithm is fully described in Chapter 5 of this dissertation.

In 2003, the author presented the 6.2 research to NAVOCEANO as a stepping-stone for an automated change detection system. NAVOCEANO realized that the steps required for AUV autonomous navigation using SSS are almost identical to those required for automated change detection. NAVOCEANO was specifically interested in using ACDC to identify newly placed objects on the seafloor that could be explosive mines. NAVOCEANO helped the author obtain additional funds from SPAWAR and the Office of Naval Research (ONR) to accelerate the 6.2 project with the long-term goal of implementing and transitioning a fully functional ACDC system by the year 2006.

Sidescan Sonar System (SSS)

Dr. Harold Edgerton at the Massachusetts Institute of Technology (MIT) first developed sidescan sonar in the 1960's. The SSS transmits an acoustical beam on each side of a transducer, sometimes called the "fish". The beams are sent in a wide angular pattern down to the bottom in swaths 100-500 meters wide, and the echoes are received back creating a narrow strip below and to the sides of the transducer track (Blondel and Murton, 1997).

Figure 1-1 depicts a SSS "flying" through the ocean at a given altitude above the seafloor. When sound is reflected back, details about the local morphology of the seafloor can be extracted (Blondel and Murton, 1997).

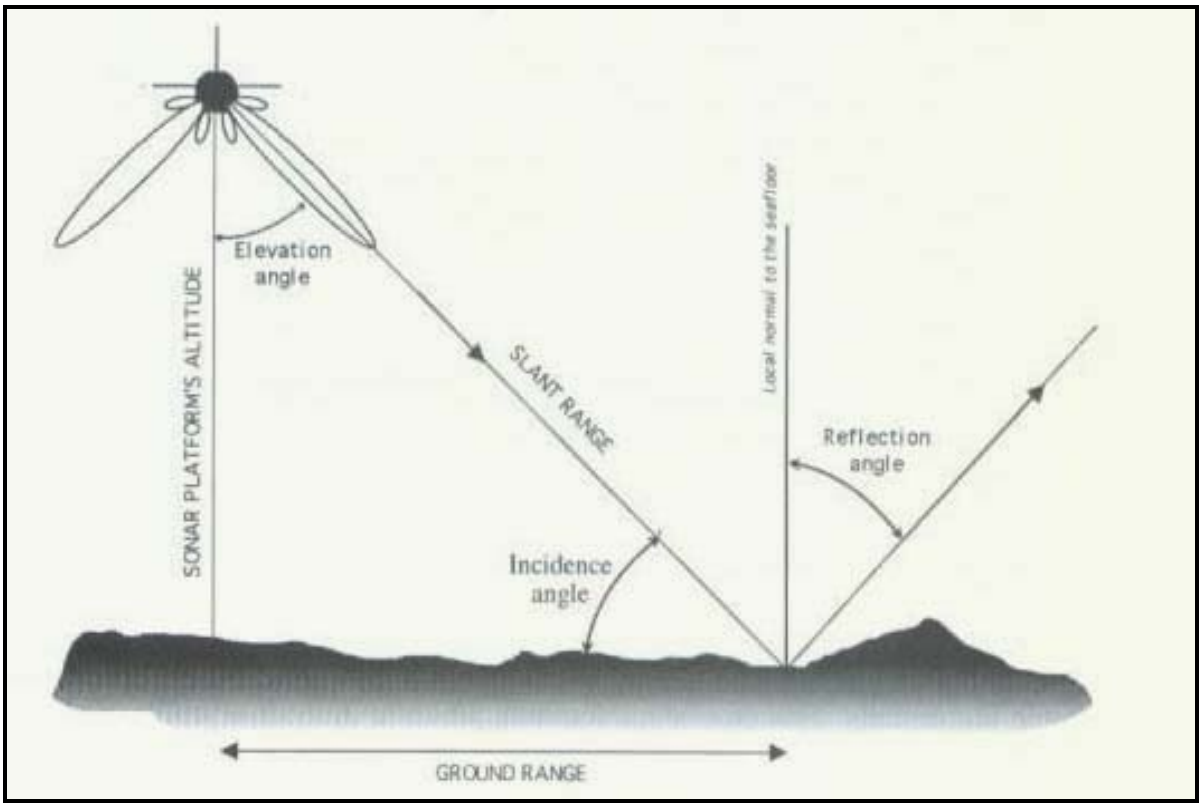


Figure 1-1. Sidescan Sonar System (Blondel and Murton, 1997)

Dynamically focused SSS, like the Klein 5000, use five phase-shift banks to form five adjacent beams (Klein Associates, Incorporated, 2004). For the Klein 5000, the adjacent beams are formed giving either 10 or 20cm resolution in the direction the sonar is traveling, called along-track resolution (Huff et al., 1991). The beams spread or fan out perpendicular to the direction of travel, called across-track. Across-track resolution is determined by the bandwidth. The bandwidth for the Klein 5000 is 20kHz (Klein Associates, Incorporated, 2004). The amount of across-track spreading is measured by the beam angle (Huff et al., 1991). The top of Figure 1-1 shows a typical beam pattern for a SSS. The main lobes point to the side making the sonar effectively "blind" directly below the sonar, called nadir.

The SSS can be hull-mounted, towed from a platform, such as a ship or helicopter, or carried on-board an AUV. The fish is usually equipped with a pressure or altimeter sensor that allows it to follow the bottom while maintaining a constant height above the sea floor or alternatively "fly" at a constant depth below the surface. Important measurements such as heading, pitch, and roll are recorded on-board the fish or, in the case of towed transducers, are often transmitted up the towing cable and recorded separately on the towing vessel (Fish and Carr, 1990).

Estimating the exact location of the fish, relative to a point on Earth, underwater is always a problem. The position of cable-towed transducers is often deduced from the global positioning systems (GPS) location of the surface tow-body by using a cable layback model (Schwab et al., 1991; Kenny et al., 2001). A free swimming AUV sometimes has sophisticated on-board inertial

navigation systems (INS) in combination with GPS. Because GPS does not function underwater, the AUV must routinely surface to get a navigational fix.

Sidescan Theory

The pressure of an acoustic wave in a material is given by (Urick, 1983),

$$p = \rho c u \ , \tag{1-1}$$

where ρ is the fluid density of the medium, c is the propagation velocity of the wave, and u is the velocity of the fluid particles. Similarly to Ohm's Law for electricity, the particle velocity, u, acts like current and ρc acts like resistance and is called the specific acoustic resistance or (when complex) impedance. The pressure of the wave, p, behaves like voltage and the energy per second, or power, is found by:

$$Power = \frac{p^2}{\rho c}, \tag{1-2}$$

(Urick, 1983). The symbol p represents the instantaneous pressure, but due to the integration time inherent in the sonar, it is more useful to look at the squared pressure as an average over an interval of time. A more general approach is to write the equation in terms of *energy flux density*, or the acoustic energy per unit area of wave front:

$$E = \int_0^\infty I \, dt = \frac{1}{\rho c} \int_0^\infty p^2{}_2 \, dt \, . \tag{1-3}$$

The intensity is the mean-square pressure of the wave divided by ρc and averaged over an integral of time length T, or

$$I = \frac{1}{T} \int_0^T \frac{p^2(t)}{\rho c} \, dt \ , \tag{1-4}$$

so that over the time interval T,

$$I = \frac{E}{T} \ . \tag{1-5}$$

The quantity T is the time interval over which the energy flux density of an acoustic wave is to be averaged to form the intensity (Urick, 1983).

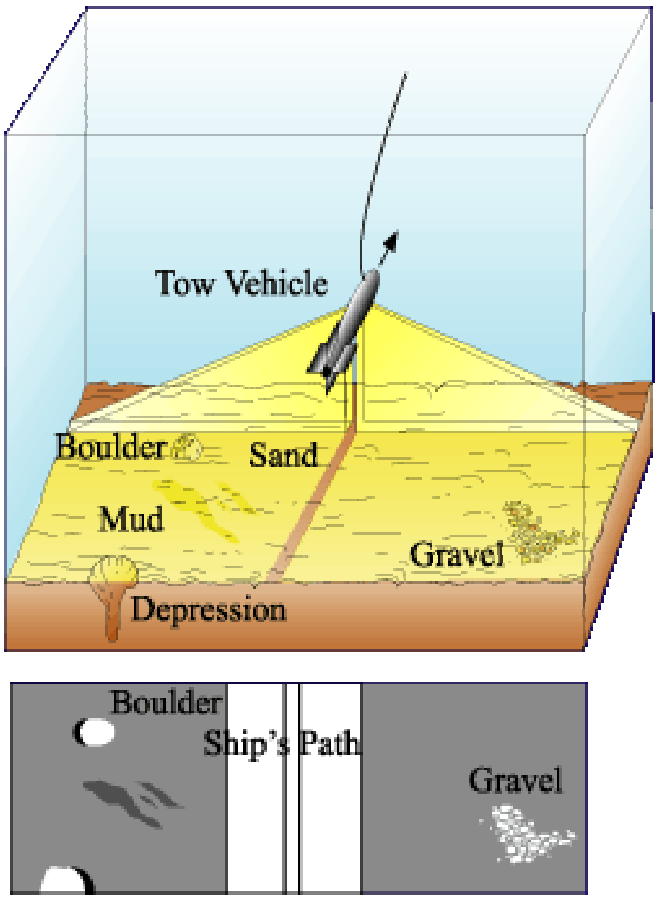


Figure 1-2. Depiction showing a sidescan sonar system (SSS) imaging over the sea bottom.

Figure 1-2 depicts a SSS being towed. The beams strike the seafloor and are reflected back to the fish. For the Klein 5000, the received sonar signal is then bandpass-filtered to a 20-kHz bandwidth between 445 kHz and 465 kHz (Huff et al., 1991). The signal, 5 V peak to peak, is basebanded and sampled at 22.75 kHz with an analog to digital converter (ADC). The ADC quantizes the signal to 12-bit signed integers (Klein Associates, Incorporated, 2004). The 4 least significant bits are discarded from each integer, and the resulting 8-bit values range from 0 to 255 (low to high intensity) to form scan lines that make up a grayscale SSI as depicted in Figure 1-3 (Klein Associates, Incorporated, 2004).

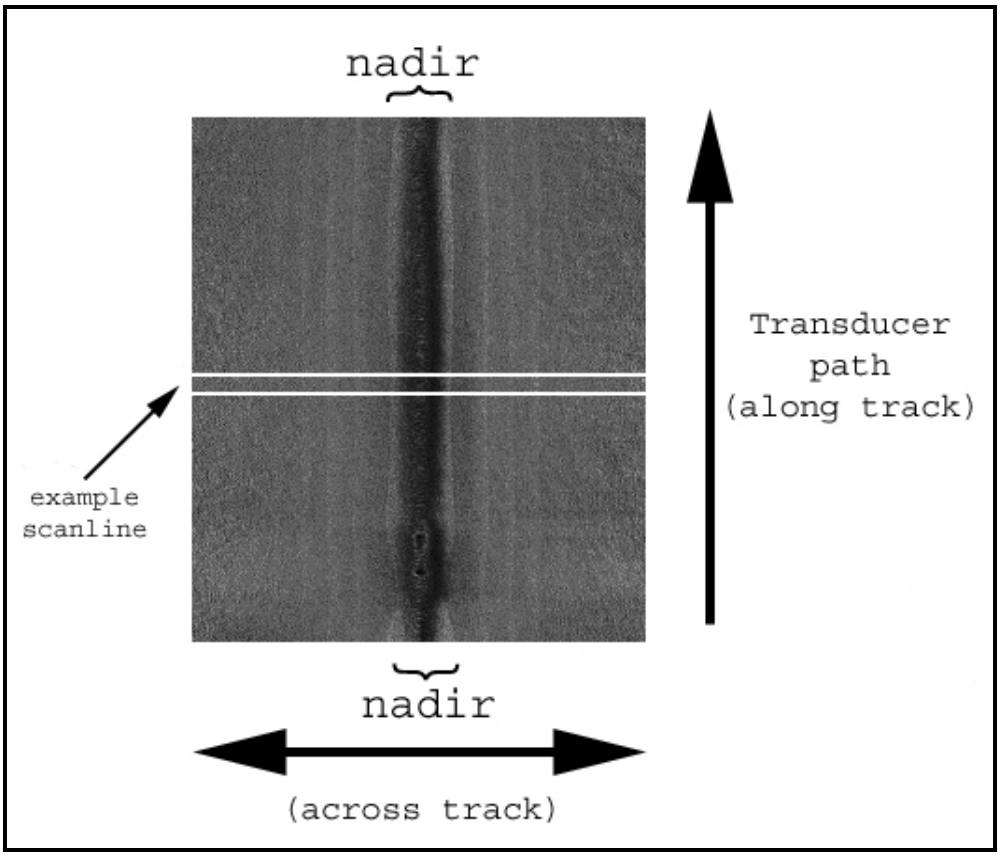


Figure 1-3. Sidescan Sonar Imagery (SSI)

Objects close to or on the seafloor, such as mines, are detected with SSI. Such objects, also called mine-like echoes (MILECs), show up in the SSI as bright spots with adjacent shadows that face perpendicular away from nadir. Features of various shapes and sizes can be detected by the shadows (Collet et al., 1996), and the size of the shadow varies as a function of beam angle and feature dimensions (Fish and Carr, 1990). Figure 1-4 shows an example of a small image extracted from SSI, called a contact snippet that contains a MILEC. NAVOCEANO maintains a Master Contact Database (MCDB) containing thousands of features detected from SSI worldwide since the early 1990's.

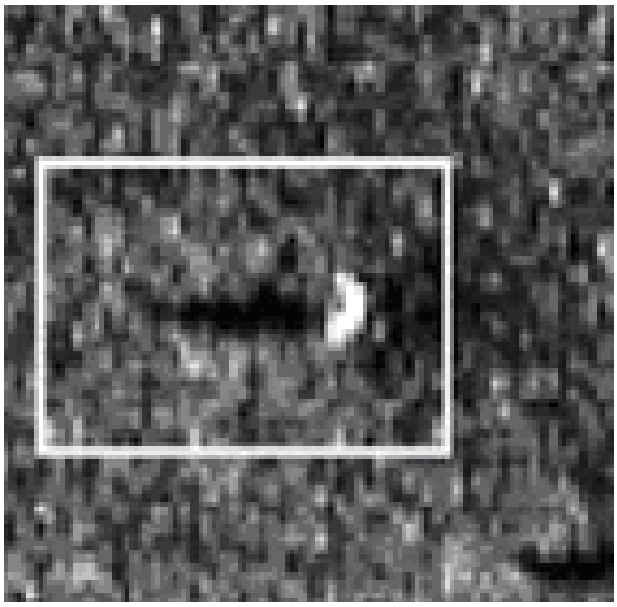


Figure 1-4. Snippet of a MILEC in SSI

Klein SSS

The imagery used for this research comes from the Klein 5000 SSS. The center frequency of the sonar is 455 kHz with a pulse length of 50 to 200 μsec that gives 10 or 20 cm along-track resolution. The across track resolution is 36 cm at a maximum of 150 m for an array 120 cm long.

The Klein 5000 (Figure 1-5) is usually towed at 10 knots with a 300 m swath at an altitude of 25 m. The sonar tow fish is equipped with standard nominal heading, pressure, pitch, altimeter and roll sensors. The sonar is compact and relatively light and easy to use. The body length is 194 cm and the diameter 15.2 cm. It only weighs 70 kg in air (Klein Associates, Incorporated, 2004).

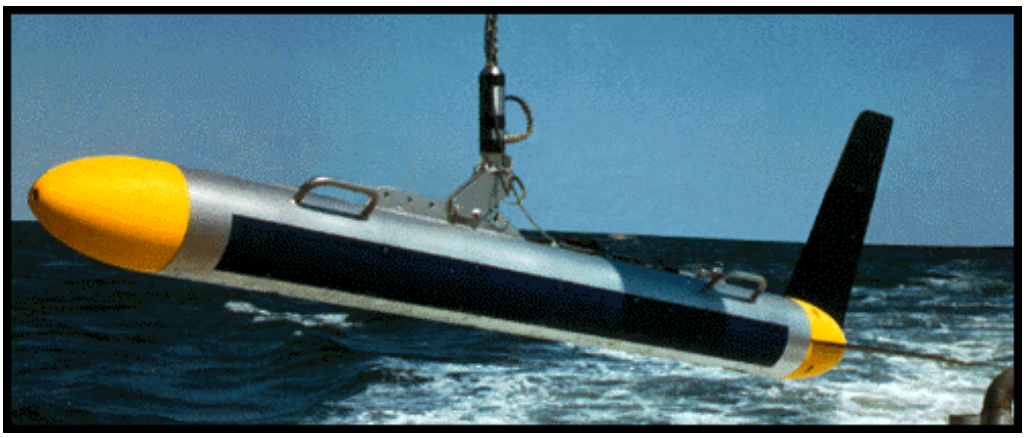


Figure 1-5. Klein 5000 Sidescan Sonar

The 8-bit SSI is produced from the Klein 5000 sonar by NAVOCEANO and is stored in the Unified Sonar Image Processing System (UNISIPS) format. Each scan line of the imagery is

stored in a separate record and the latitude and longitude coordinate, the sonar heading, speed and depth above the seafloor are given for the sample at nadir.

Problems with Sidescan

By its nature, SSI is non-linear (Reed et al., 2002). Correlation between pixels in SSI is affected by "speckle" noise (Blondel and Murton, 1997). This speckle noise also hampers the extraction of targets and features in sidescan imagery (Reed et al., 2002). The imagery can contain phantom features created by surface return when the sonar is in shallow water or when the fish is pitched (Kenny et al., 2001). GPS signal dropout at the surface can also cause the imagery processing software to lose scan lines or to duplicate others (Fish and Carr, 1990). Any automated feature detection method that is applied to the imagery must be robust and able to cope with all these issues.

Bottom objects and features can also change and migrate over time due to ocean currents and burial. Even when the objects are stationary, another problem, probably one of the biggest issues with sidescan, is position error observed to be 15 m or greater during actual surveys (Schwab et al., 1991). The center latitude/longitude position of each scan line in the UNISIPS file contains position error due to GPS error and error from the cable layback model. Because the Klein 5000 is usually towed through the water with a cable attached to a tow platform, sometimes-large positioning errors are introduced (Schwab et al., 1991, Kenny et al., 2001). GPS will not work underwater, so the position of the tow platform, determined by GPS, is first measured and then the position of the fish is computed by taking into account the length of the cable. If the GPS antenna is not mounted at the same location where the tow cable attaches to the platform, the "antenna offset" must be included in the calculation. If one assumes that the cable is a straight line, the cable length can be computed by using the Pythagorean theorem (Figure 1-6). In reality, the cable is not straight, and a more complicated equation, called a cable layback model must be used (Figure 1-7).

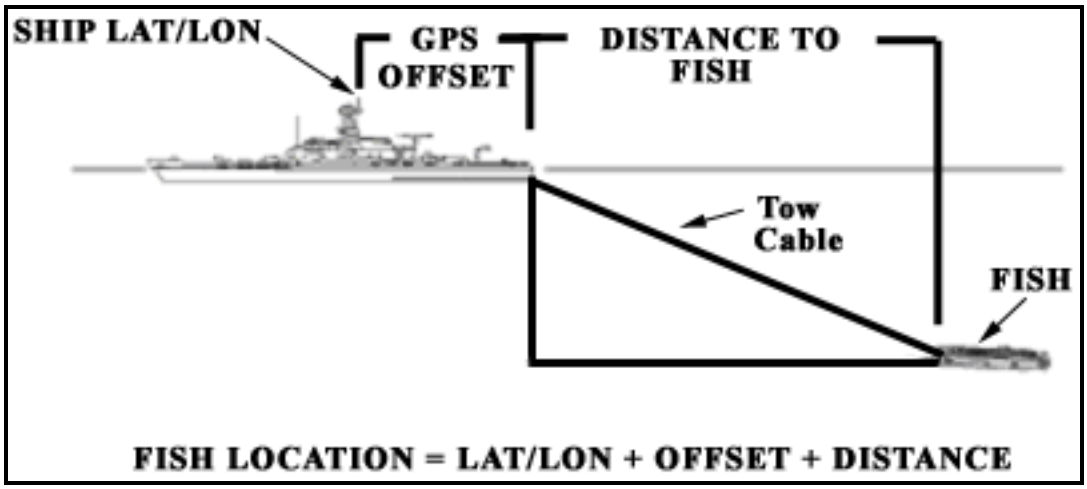


Figure 1-6. Tow Cable

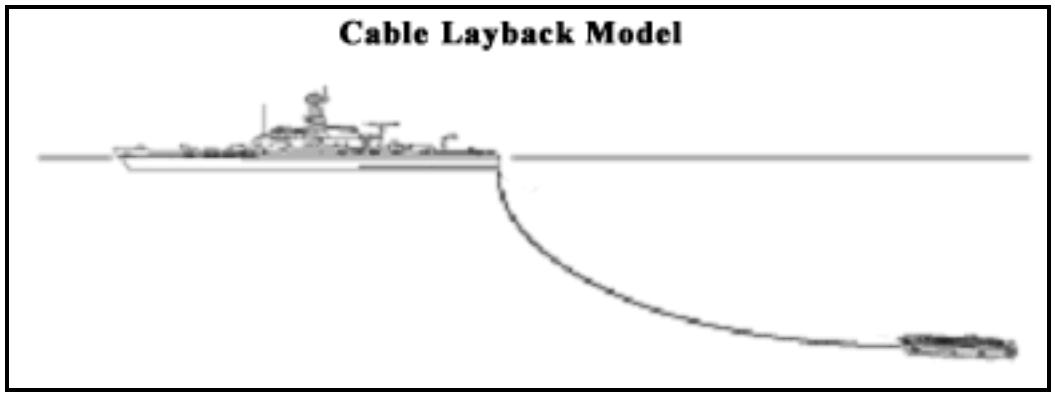


Figure 1-7. Cable Layback Model

ACDC System

The purpose of this dissertation is to present digital change detection algorithms and data structures, developed by the author, necessary to build an ACDC system. The enormous increase in the volume of SSI produced by modern SSS (Reed, 2001; Cronin et al., 2003) reduces the benefits of manual change detection, large navigational position errors between recent and historical SSI, and distortions in the SSI (Lingsch and Lingsch, 2001). Analysts desperately need software algorithms to 1) automatically detect objects of interest, 2) cluster those objects so large areas of uninteresting data can be discarded, 3) search large databases containing historical data while dealing with position error, and 4) automatically match new objects with historical objects from different look-angles.

Chapter 2 presents a critical data structure developed by the author that is central to these algorithms. Chapters 3, 4 and 5 of this dissertation introduce the algorithms that perform the above-mentioned items 1, 2 and 3, respectively. Finally, the conclusion chapter describes the on-going work to produce tools necessary to perform item 4.

Figure 1-8 is a block diagram of the ACDC system. ACDC requires two inputs, 1) new collected SSI in the UNISIPS, and 2) historical snippets from the NAVOCEANO MCDB. The newly collected UNISIPS files are first processed by the author's real-time CAD algorithm ("Detection" block – Figure 1-8) that detects MILECs in real-time.

The output of the first stage is image snippets of MILECs. The locations of the MILECs are then grouped geographically (Type I Clustering – Figure 1-8) by the author's unique clustering algorithm and a density computed so analysts can make crucial decisions about which areas should be examined more closely or altogether eliminated.

The snippets are then sent to a classification stage ("Classification" block - Figure 1-8) where a computer-aided classification (CAC) algorithm attempts to classify the detections as mine-like objects (MLO) or non-mine, mine-like bottom object (NOMBOS). The CAC algorithm attempts to determine attributes such as size and shape, and the algorithm then computes a confidence measure. The completion process is not fully automated and prompts the operator to make a final judgment when the confidence measure falls below a set threshold.

The MLO snippets, with associated attributes, can then be clustered again (Type II Clustering – Figure 1-8) to produce improved clusters to further aid the analysts in decision-making. The location of each MLO is then passed to the author's CAS algorithm ("Searching" block – Figure 1-8) that queries the MCDB and finds all the historical MLOs that are "spatially close" based on an estimate of position error.

ACDC then tries to automatically "match" one of the historical MLOs with the new MLO by using a feature-matching (FM) algorithm ("Identification" block – Figure 1-8). When a match is found, an area-matching (AM) algorithm ("Identification" block – Figure 1-8) attempts to match spatially close features located around the new MLOs with the same features in the historical imagery. New objects that are not matched, i.e., not in the database, are identified as "new objects" (change detection). The results are sent to the display ("Data Fusion" block – Figure 1-8). The center LAT/LON positions of features not observed in the past, i.e. not in the MCDB, can be clustered (Type III Clustering – Figure 1-8).

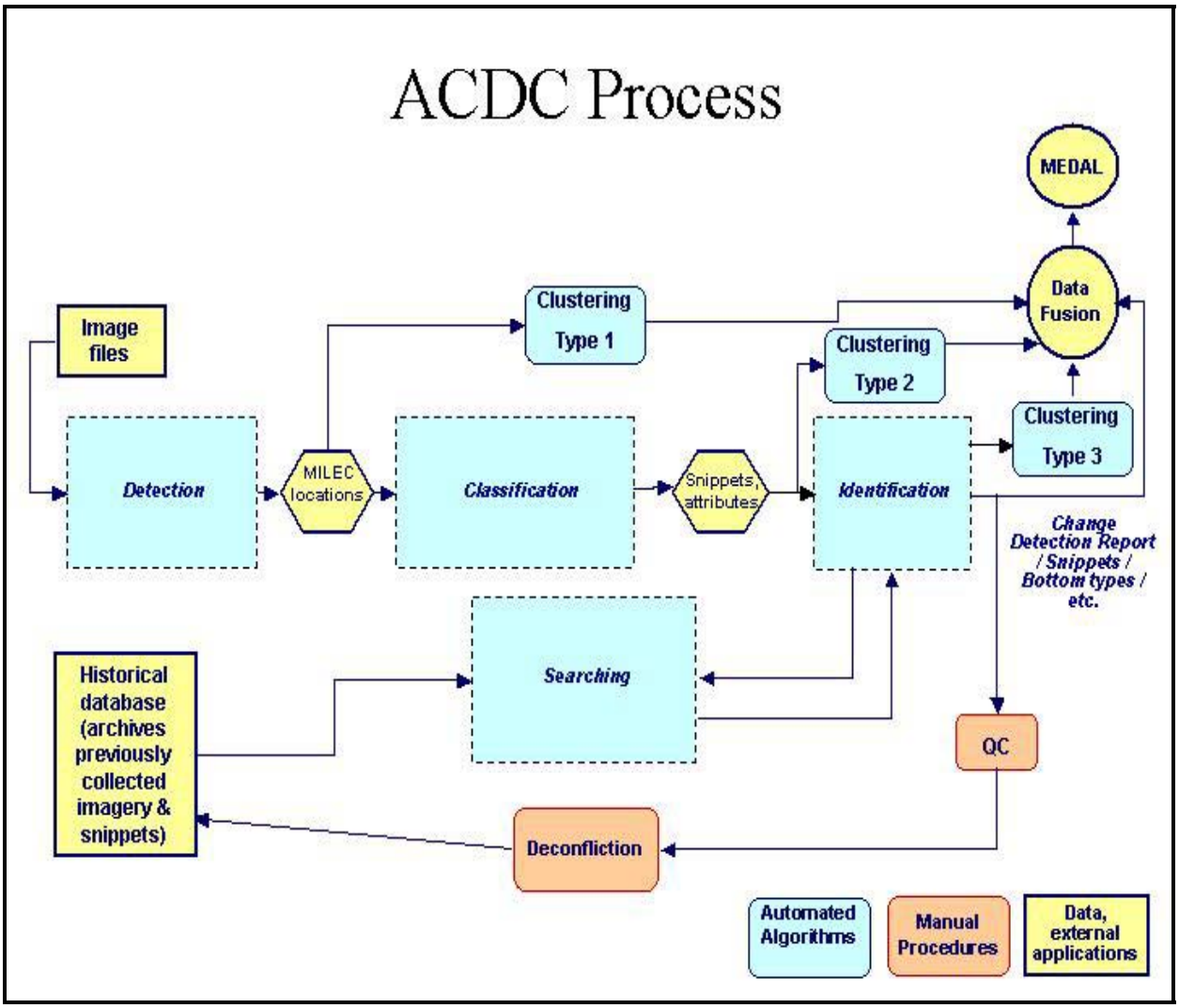


Figure 1-8. ACDC System

Dissertation Chapters

Chapter 2 – The Geospatial Bitmap (GB)

Digital change detection algorithms should be able to cope with high volumes of historical and newly collected SSI during MIW operations. Choosing the correct data structure when implementing such algorithms is paramount. This chapter presents the history, development and evolution of the GB (Gendron et al. 2001). Three digital change-detection algorithms, presented in the next three chapters, utilize GBs in a variety of ways to improve efficiency.

Chapter 3 – The Geospatial Bitmap (GB) Clustering Algorithm

Onboard naval mine-hunting vessels, change detection can be used to determine the presence or absence of mines, thus reducing mine clearance timelines and developing an accurate assessment of risk to follow-on naval forces. Geographical areas of high clutter, or areas where there is a high density of mine-like objects, are avoided altogether. Naval analysts need a software tool that will allow them to quickly cluster MLOs geographically to decide which areas to avoid. This chapter presents a clustering algorithm that is fast, efficient, and produces repeatable results.

Chapter 4 – A Real-Time Computer-Aided Detection (CAD) Algorithm

There are a variety of digital change detection techniques including image differencing (Weismiller et al., 1977; Miller et al., 1978; Williams and Stauffer, 1978), image regression (Singh, 1986; Jha and Unni, 1994) and post-classification comparison. The post-classification change detection technique compares two images that were classified independently (Howarth and Wickware, 1981). The author has applied this technique to SSI, and Chapter 4 describes a new and unique CAD algorithm (Figure 1-9), developed by the author, that is capable of detecting and extracting snippets from historical and recently collected SSI so comparisons can be made later. The CAD algorithm is able to run in real-time (one scan line at a time) by utilizing GBs.

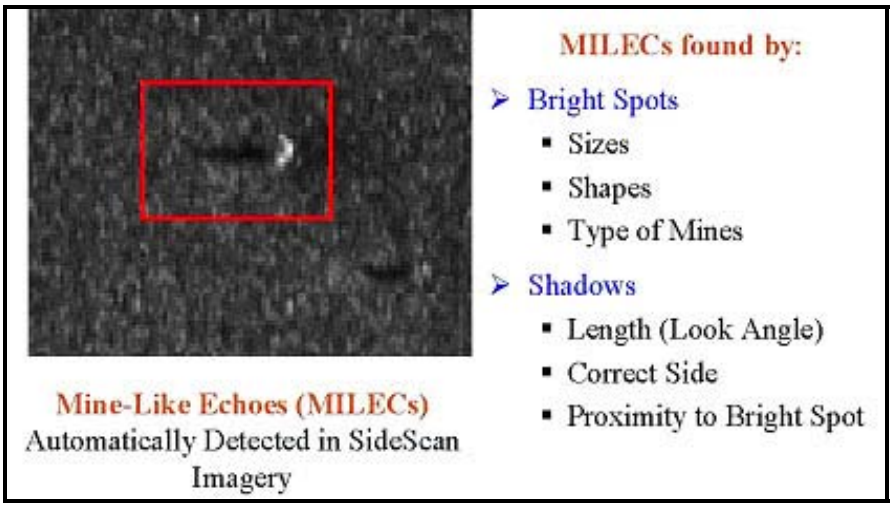


Figure 1-9 The Author's Real-Time Computer-Aided Detection (CAD) Algorithm

Chapter 5 – A Computer-Aided Search (CAS) Algorithm

One factor that can greatly reduce the accuracy of post-classification change detection is the inaccurate geometric registration between the two images (Howarth and Wickware, 1981; Mas, 1999; Singh, 1989). In most cases, the accurate geo-registration of SSI is impossible due to inherent position error. This chapter describes a CAS algorithm developed by the author, which uses GBs and a modified quadtree structure to accurately and efficiently perform geospatial searches (Figure 1-10).

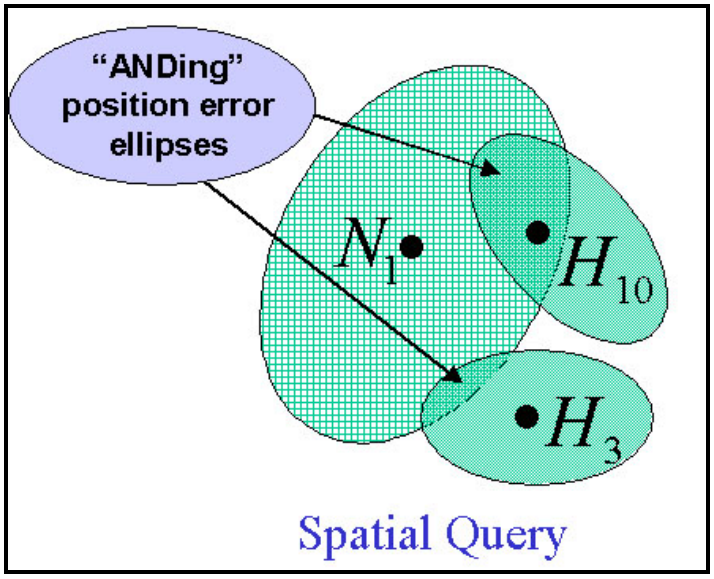


Figure 1-10. Computer-Aided Search Query

Chapter 6 - Conclusion

This chapter discusses the ACDC system in more detail and gives more information about each subsystem. Additional information on a "completion" algorithm used for automated classification is discussed (Figure 1-11). The reader will be brought up-to-date on current research to develop robust FM and AM algorithms. Finally, recent work by the author to develop a FM algorithm is discussed. The author has concentrated his research on wavelet networks, which use wavelets coefficients as input to a neural network. Wavelet networks in the past have been proven to work well at matching features and are used extensively in face recognition, for example (Krueger and Sommer, 2000).

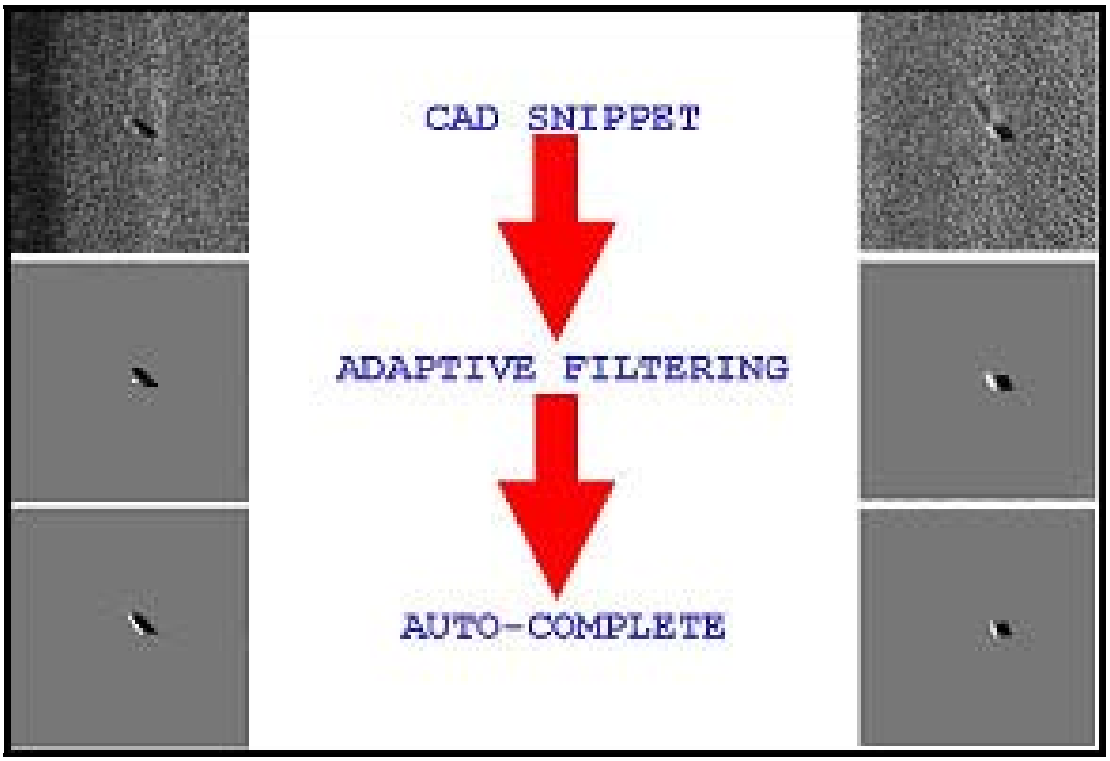


Figure 1-11. Results from the Completion Algorithm

References

Banner, A.V. (1991). Change Detection Using Commercial Satellite Imagery to Monitor Large-Scale Withdrawals of Conventional Forces. *Overhead Imaging for Verification and Peacekeeping: Three Studies*. The Arms Control and Disarmament Division, External Affairs and International Trade, Canada.

Blondel, P. and B Murton (1997). *Handbook of Seafloor Sonar Imagery*. New York: John Wiley & Sons.

Collet, C., P. Thourel, P. Perez, and P. Bouthemy (1996). Hierarchical MRF modeling for sonar picture segmentation. Proceedings of the *3rd IEEE International Conference on Image Processing*, 3, Lausanne, Switzerland.

Cronin, D., M. Broadus, B. Reed, S. Byrne, W. Simmons, and L. Gee (2003). Hydrographic Work Flow – From Planning to Products. Proceedings of the *U.S. Hydro 2003 Conference*, March 24-27, Biloxi, Mississippi.

Deer, P.J. (1995). Digital Change Detection Techniques: Civilian and Military Applications. Abstract for the *International Symposium on Spectral Sensing*, Reseach '95 Reports.

Fish, J. P. and Carr, H. A. Carr (1990). *Sound Underwater Images: A guide to the Generation and Interpretation of Sonar*. American Underwater Search and Survey, Lower Cape Publishing: Massachusetts.

Gendron, M., G. Layne and M. Lohrenz (2004a). The Geospatial Bitmap (GB) Clustering Algorithm. Patent Pending, Navy Case Number 76,458.

Gendron, M., M. Lohrenz, G.J. Layne and J.W. Ioup (2004b). Automatic Change Detection and Classification (ACDC) System. Proceedings of the *Sixth International Symposium on Technology and the Mine Problem*, Naval Postgraduate School, Monterey, CA. May 9-13.

Gendron, M., G. Layne and M. Lohrenz, (*submitted*). The Geospatial Bitmap (GB) Clustering Algorithm. Naval Research Laboratory, Stennis Space Center, Mississippi, submitted to the *Journal of Discrete Algorithms*.

Gendron, M., P. Wischow, M. Trenchard, M. Lohrenz, L. Riedlinger, M. Mehaffey (2001), *Moving Map Composer*. Naval Research Laboratory, US Patent No. 6,218,965.

Griffiths, G.H., 1988. Monitoring Urban Change from Landsat TM and SPOT satellite imagery by image differencing. Proceedings of the *1988 International Geoscience and Remote Sensing Symposium on Remote Sensing*, 1.

Howarth P.J. and G. M. Wickware (1981). Procedures for Change Detection Using Landsat Digital Data. *Int. J. of Remote Sensing,* 2.

Huff, L., N. Langhorne, R. Quiqley, J. Weintroub, R. Kuwahara, J. Preston, and A. Hart (1991). Multibeam foucused sonar (MBFS) experiments, Plymouth, UK, June-July 91. Klein Associates, Inc., Tech. Rep.

Ioup, J.W., M.L. Gendron, M.C. Lohrenz, G.J. Layne and G.E. Ioup (2004). Classifying sidescan sonar images using self organizing maps. Abstract of the *J. Acoust. Soc. Am*., 116.

Ioup, J.W., M.L. Gendron, P.J. McDowell, B.S. Bourgeois and G.E. Ioup (2003). Wavelets and Neural Networks for Change Detection in Simulated Sidescan Sonar Data. Abstract of the *J. Acoust. Soc. Am*., 114.

Jha C.S. and N.V.M. Unni (1994). Digital Change Detection of Forest Conversion of a Dry Tropical Forest Region. *Int. J. of Remote Sensing,* 15.

Kenny, A.J., B.J. Todd, and R. Cooke (2001*). Procedual Guidelines No. 1-4 – The Application of Sidescan Sonar for Seabed Habitat Mapping, Marine Monitoring Handbook*. The Joint Nature Conservation Committee, Peterborough, United Kingdom.
http:// www.jncc.gov.uk/marine/mmh/ MMH_0601.pdf

Klein Associates, Incorporated (2004). Klein Associates, Incorporated. Website: http://www.kleinsonar.com.

Krueger, V. and G. Sommer (2000). Affine Real-Time Face Tracking using Gabor Wavelet Networks. *Internation Conference on Pattern Recognition*, Barcelona, Spain, September, 3-8.

Lambin E.F. and A.H. Strahler (1994). Change Vector Analysis in Multitemporal Space: a Tool to Detect and Categorise Land-Cover Change Processes Using High Temporal Resolution Satellite Data. *Remote Sensing of Environment*, 48.

Lingsch, W.C. and S.C. Lingsch (2001). Sonar Image Change Detection as a Mine Counter Measures Tactical Tool. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

Mas, J.F. (1999). Monitoring land-cover changes: A comparison of change detection techniques. *Int. J. Remote Sensing*, 20.

McDowell, P.J., M.L. Gendron, P.M. McDowell, J.W. Ioup and G.E. Ioup (2003). Kalman Filtering With Neural Networks for Change Detection in Simulated Sidescan Sonar Data. Abstract of the *J. Acoust. Soc. Am*., 114.

Miller L.D., K. Nualchawee and C. Tom (1978). Analysis of the Dynamics of Shifting Cultivation in the Tropical Forests of Northern Thailand Using Landscape Modelling and

Classification of Landsat Imagery. *NASA Technical Memorandum 79545*, Goddard Space Flight Center, Maryland.

Reed, B. (2001). Data, Data Everywhere and Nary a Bit to Drop. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

Reed, S., E. Dura, D.M.Lane, Y. Petillot, J. (2002). Extraction and Classification of Objects from Sidescan Sonar. *IEEE Workshop on Nonlinear and Non-Gaussian Signal Processing* 8-9th July.

Rosin, P.L., (1994). Parcel-Based Change Detection. *European Symposium on Satellite Remote Sensing*, Institute for Remote Sensing Applications, Joint Research Centre, Ispra, Italy.

Schwab, W.C., Webb, R.M.T., Danforth, W.W., O'Brien, T.F., and Irwin, B.J. (1991). High-Resolution Sidescan-Sonar Imagery of the Manchas Interiores - Manchas Exteriores Coral Reef Complex, Mayagüez, Puerto Rico. *U.S. Geological Survey Open-File Report*.

Singh A. (1986). Change Detection in the Tropical Forest Environment of Northeastern India Using Landsat. *Remote Sensing and Tropical Land Management*, Chichester: John Wiley & Son.

Singh A. (1989). Digital Change Detection Techniques Using Remotely-Sensed Data. *Int. J. of Remote Sensing*, 10.

Stow D.A., D. Collins and D. McKinsey (1990). Land Use Change Detection Based on Multi-date Imagery from Different Satellite Sensor Systems. *Geocarto International*, 5.

Weismiller R.A., S.J. Kristof, D.K. Scholz, P.E. Anuta and S.A. Momin (1977). Change Detection in Coastal Zone Environments. *Photogrammetric Engineering and Remote Sensing,* 43.

Williams D.L. and M.L. Stauffer (1978). Monitoring Gypsy Moth Defoliation by Applying Change Detection Techniques to Landsat Imagery. Proceedings of the *Symposium for Vegetation Damage Assessment*, American Society for Photogrammetry.

Urick, R.J. (1983). *Principles of Underwater Sound*. 3rd Edition, New York: McGraw-Hill.

Chapter 2 - The Geospatial Bitmap (GB)

Marlin L. Gendron, Geary J. Layne and Maura C. Lohrenz
Naval Research Laboratory (NRL), Code 7440.1, Stennis Space Center, MS

Abstract

This paper presents the geospatial bitmap (GB) data structure, developed by the author to solve map data processing issues in polar regions and later improved for use in data clustering, computer-aided detection in sidescan sonar imagery, and spatial database searching. A GB is comprised of a matrix of rows and columns of bits, in which each bit represents some measurable quantity at some unique location in space, accurate to within the resolution of the bitmap. A "set" bit (=1) in the GB indicates that an object of interest exists at that geospatial location. A "cleared" bit (=0) indicates the absence of any object at that location. GBs are demonstrated in a polyline-smoothing algorithm. Processing time and results are compared for running the polygon-smoothing algorithm using 1) GBs, 2) Euclidean geometry, and 3) Sparse Matrices (SM). Results show that the GB implementation ran between 1.3 – 48.4x faster than either of the other methods, depending on the size and number of the polygons to be smoothed.

Introduction

In the preface of the book, *Computational Ocean Acoustics*, Jensen et al. (2000) emphasize the key role that computers play in modern scientific research and state that computers have already changed our approach to doing experimental and theoretical science. In *Algorithms and Data Structures: The Science of Computing*, Baldwin and Scragg (2004) stress the importance of choosing the correct data structure when developing computer algorithms to solve some complex scientific problems.

In the early 1990's, the Naval Research Laboratory (NRL), with the aid of the author, designed and developed the Compressed Aeronautical Chart database (Wischow and Lohrenz, 1998; Lohrenz et al., 1992; Lohrenz et al., 1990). To produce this database, NRL converted equal Arc-second Raster Chart (ARC) Digitized Raster Graphics (ADRG), produced by the National Geospatial-Intelligence Agency (NGA) (DMA, 1989), from the ARC system frame of reference into the Tessellated Spheroid (TS) projection (Lohrenz et al., 1993) at several geographic scales ranging from 1:50,000 to 1:5,000,000. This conversion is accomplished by applying 1) a neighborhood averaging function (downsampling), 2) a color-compression from 24 bits-per-pixel to 8 bits-per-pixel (Myrick et al., 1992) and 3) a vector quantization (VQ) spatial compression (Miller et al., 1994). These steps compress the ADRG tiles by 48:1 and produce a seamless database of rectangular digitized raster map files called segments that exist at distinct geographic locations.

Because the Earth is spheroidal, it is geometrically impossible to represent it with a single mesh of uniform rectangles (Dutton, 1983). To minimize distortion, the TS projection divides the

supported map scales into five distinct regions, or zones: 1) south polar (below 50°S latitude), 2) south temperate (between 50°S and 30°S latitude), 3) equatorial (between 30°S and 30°N latitude), 4) north temperate (between 30°N and 50°N latitude), and 5) north polar (above 50°N latitude. The processing of non-polar map segments was easily accomplished in the two-dimensional (2D) Cartesian coordinate system; however, processing of polar map segments had to correct for projection distortion.

The author developed a new approach that would consistently process all segments in all zones, in which the data structure linked each cell (called a bit) of a simple binary data structure (bitmap) with a unique geospatial location. Specifically, a 2D bitmap (e.g., Figure 2-1) is a structure in which each bit is indexed by a column (x) and row (y). A third dimension is indexed by depth (z).

Gendron et al. (2001) patented a process that extended the simple bitmap concept to construct GBs, in which every bit represents some measurable quantity at some unique location in n-D space, accurate to within the resolution of the bitmap. For example, a set bit in the GB could indicate that an object of interest exists at that geospatial location. Cleared bits could indicate the absence of objects at other geospatial locations. In certain cases, it is desirable to reverse this logic for the purpose of compression.

The GB greatly decreased the complexity of processing map data in polar zones, and NRL successfully transitioned the Compressed Aeronautical Chart processing system to NGA in the late 1990's. The GB data structure was then used in many components of a graphical information system (GIS) called the Moving-Map Composer (MMC) system (Myrick et al., 2003). Both the Compressed Aeronautical Chart and MMC systems are efficient, robust and widely used by the U.S. Navy and foreign allies.

This chapter describes improvements made by the author to the GB data structure. The purpose of this chapter is to demonstrate the improvement to the GB data structure over several generations. Note that memory usage comparisons between generations are made by the author to show that the GB's memory requirements go down in each new generation. Ultimately, the author wants to utilize GBs in real-time processing algorithms and, thus, is primarily interested in determining and reducing the processing time of GBs. In the testing section of this chapter, the GB data structure and a Euclidean geometry method are used separately in a polygon-smoothing algorithm and processing time compared.

Also, improvements to the GB support software components of author's Automatic Change Detection and Classification (ACDC) system (Gendron et al., 2004). ACDC algorithms use GBs to perform data clustering, computer-aided searching (CAS), detection (CAD), classification (CAC), identification (CAI), and change detection.

Terminology

A bitmap is a three-dimensional data structure in which two of the dimensions are made up of a grid of regular rectangles called cells of a specified height and width, and the remaining dimension corresponds to depth. The depth, or bits-per-pixel, is usually set to 8, which allows

the bitmap to store up to 256 different values in each cell. The standard 8-bit bitmap is ideal for storing grayscale images. A "pixel map" – or pixmap – is a special type of bitmap with greater depth values (Howe, 2004). For example, a 24-bit pixmap is used to store RGB (red, green, blue) images.

In image processing, especially when discussing mathematical morphology where sets represent objects within the images, the term bitmap is rarely used and is replaced with names such as binary, gray-scale, and color images. These names are based on the dimension of the integer Z space that is needed to represent these sets. For example, sets in binary images are members of Z^2 integer space and sets in gray-scale images are members of Z^3 integer space (Gonzalez and Woods, 2002).

In this paper, the term bitmap is more in keeping with the definition stated above and thought of as a computer data structure or class. The author is primarily interested in developing efficient classes and methods for computer algorithms that operate on 2D geospatial data rather than static raster images, although the bitmap class and accompanying methods discussed here work equally well when there is no geospatial component. The author has also modified the GB class to support higher dimensions for real applications (addressed in future papers), and the bitmap methods support morphological operations (discussed later in this paper).

Throughout this paper, C code terminology (e.g., "malloc") is mixed with mathematical symbology (e.g., $\lceil\ \rceil$ for "ceiling") in the equations. Table 2-1 presents a list of all terms and symbols used.

Table 2-1. Acronyms, Terms and Symbols

ACDC	Automatic Change Detection and Classification system
ADRG	ARC Digitized Raster Graphics
AOI	Area Of Interest
ARC	equal Arc-second Raster Chart
$\lceil\ \rceil$	Ceiling (i.e., rounding the contained value up to the next largest integer)
CAC	Computer-Aided Classification
CAD	Computer-Aided Detection
CAI	Computer-Aided Identification
CAS	Computer-Aided Searching
DMA	Defense Mapping Agency (now NGA)
$\lfloor\ \rfloor$	Floor (i.e., rounding the value down to the next smaller integer)
GB	Geospatial Bitmap
GIS	Graphical Information System
LAT/LON	Latitude and Longitude coordinate
malloc	Memory allocation function in C
MIW	Mine Warfare
MMC	Moving-Map Composer
NC	Number of Columns in a bitmap
NGA	National Geospatial-Intelligence Agency (formerly DMA and NIMA)
NIMA	National Imagery and Mapping Agency (now NGA)

NR	Number of Rows in a bitmap
NRL	Naval Research Laboratory
NAVOCEANO	Naval Oceanographic Office
POS	Points Of Significance
ROW/COL	Row and Column coordinate
SM	Sparse Matrix
TS	Tessellated Spheroid
VQ	Vector Quantization
WGS84	World Geodetic System 1984

GB Evolution

The use of a bitmap data structure within a software algorithm can reduce the number of I/O (input/output) operations and increase performance (Yoon et al., 2001; Edelstein, 1995). This is especially important for real-time applications. Also, if the depth is small, bitmaps are very space-efficient (Ozbutun, 1997). A bitmap with a depth of one is called an "ideal bitmap": the value in each cell can be represented by one bit, which indicates whether the cell is empty or occupied (Gambino et al. 1996). The term bit is shorthand for "binary digit". There are only two possible binary digits: 0 and 1, often referred to as "cleared" or "set" bits, respectively (Downing and Covington, 1986). Figure 2-1 shows a sample bitmap with 29 rows and 30 columns, in which set bits are black and cleared bits are white.

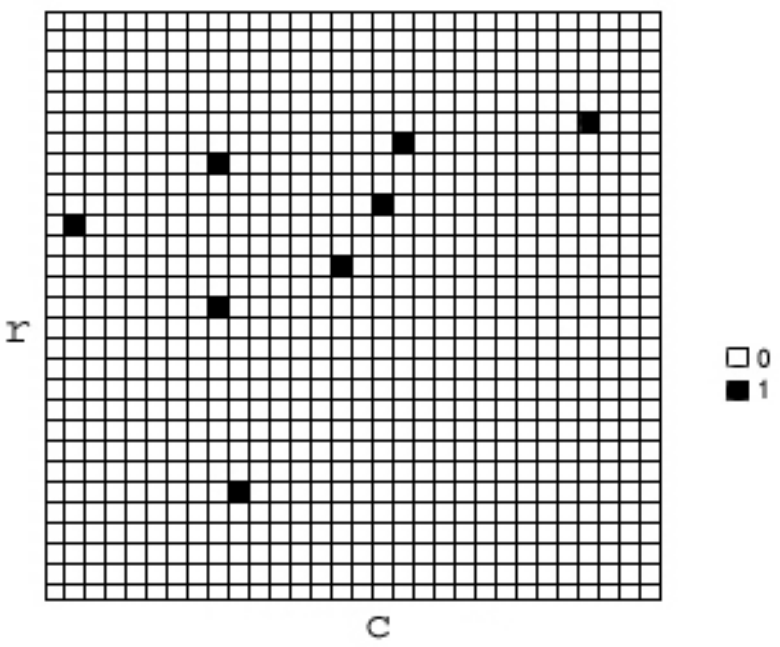


Figure 2-1. Sample bitmap with 1 bit per pixel.

Each bit in the bitmap has a unique row and column (ROW/COL), or index, which denotes its location. Like seats in a movie theater, the bitmap can show which seats are occupied and which are empty, but the bitmap, by itself, does not indicate, for example, in which town the theater is located. The bitmap also does not show what part of town, on which street, or on which block

the theater is located. To accomplish this, a bitmap has to be linked to a geospatial coordinate system, by some mechanism. The resulting GB is able to store this kind of information at different spatial resolutions or (in cartography) different map scales.

The next sections of this paper describe how the author has developed, modified, and improved the GB data structure.

First Generation GBs

This section describes the development of 2D GBs for the latitude and longitude coordinate system, using the World Geodetic System 1984 (WGS84) datum and Mercator projections (Snyder, 1987). This first generation GB described the geospatial location and resolution of bits within the bitmap. The fields contained in the data structure are as follows:

1. Minimum longitude x_0
2. Maximum longitude x_1
3. Minimum latitude y_0
4. Maximum latitude y_1
5. Longitude resolution s_x based on the map scale of the data being represented
6. Latitude resolution s_y based on the map scale of the data being represented
7. Bitmap *B* of depth 1.

From these values, the number of rows *NR* and columns *NC* of the bitmap are calculated:

$$NR = \lceil \Delta Y / s_y \rceil \text{ where } \Delta Y = | y_1 - y_0 | \quad (2\text{-}1)$$

$$NC = \lceil \Delta X / s_x \rceil \text{ where } \Delta X = | x_1 - x_0 | \quad (2\text{-}2)$$

The definition of ΔX is more complicated when the bitmap crosses from -180° to 180° longitude. Logic was added to the code to address this issue and is not discussed here.

In the first generation of the GB software, a bitmap *B* is created as a 1D array of sequential bytes. The reader probably recognizes that using a byte to represent one bit in the bitmap is wasteful. The first generation bitmaps used bytes to represent bit cell because 1) the software was easier to implement, 2) the source code contained less lines of code and, therefore, was easier to debug and ultimately contained less errors, and 3) the bit cells (bytes) can be directly accessed by a row and column index.

In high-level programming languages, such as C, the byte is the smallest addressable chunk of memory: There are no logical operations that allow software to operation on individual bits. For example, the software cannot directly "AND" one bit with other because the bits cannot be stored separately. The technique of bit-masking allows bit cells to be represented by individual bits and allows operation like "ANDing" of bits, but the software can be more complicated. The next generation will show how the GB data structure is improved by the use of bit-masking.

The bitmap B is created by allocating memory for the number of rows and columns:

$$B = \text{(unsigned char*)malloc(} NR \text{ * } NC \text{);} \tag{2-3}$$

The mapping from x, y to r, c is simply:

$$r = \lfloor (y - y_0) / s_y \rfloor \tag{2-4}$$

$$c = \lfloor (x - x_0) / s_x \rfloor \tag{2-5}$$

Note that the resulting row and column are valid (within the bitmap) as long as x, y represents a [latitude, longitude] location within the bounds of the bitmap described by the geospatial structure. From r, c a 1D index into the 2D bitmap is computed by:

$$i = rNC + c\,. \tag{2-6}$$

This index is used to access the corresponding bit in the bitmap (in C notation) by $B[\,i\,]$. Figure 2-2 depicts the resulting geospatial bitmap. The inverse mapping is defined as follows:

$$x = x_0 + (s_x c) \tag{2-7}$$

$$y = y_0 + (s_y r) \tag{2-8}$$

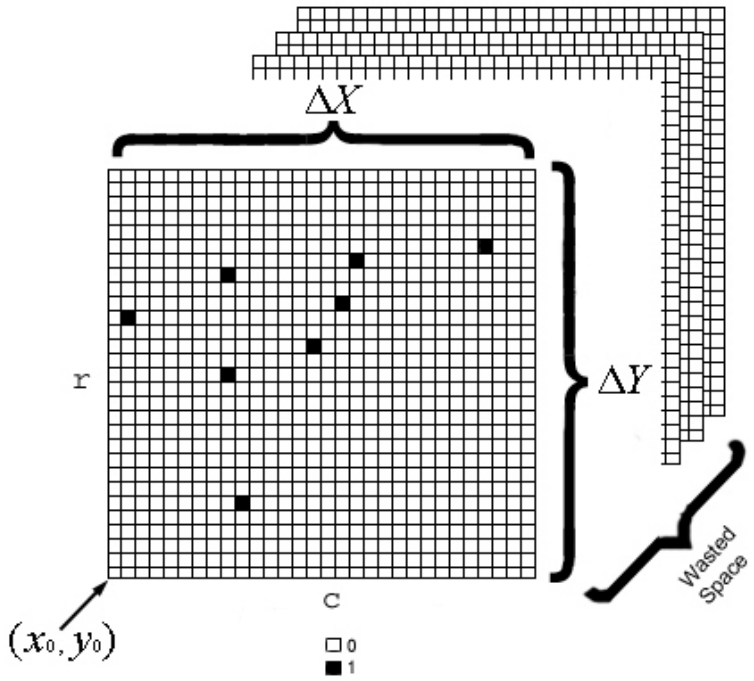


Figure 2-2. First generation geospatial bitmap.

This GB scheme is adequate for some applications, but has two serious drawbacks. First, the memory allocated for the GB is larger than required (the "wasted space" illustrated in Figure 2-2). Because an entire byte is used to store a single 1-bit cell value, the GB is eight times larger than necessary. Second, the GB (when set bits are used to indicate that something exists) can become very large over a geographic area when, in fact, bits are set only in localized areas. This can leave large areas of wasted space (cleared bits) in the bitmap. Figure 2-3 shows an example where a GB is draped over a geographic area where bits are set only over the landmasses (e.g., locations of airports). In some cases, large expanses of water (which do not include any set bits) require considerable memory to represent the AOI. At higher resolutions, increasingly more memory is wasted on cells where bits will never be set. The next two generations of GB solve both problems.

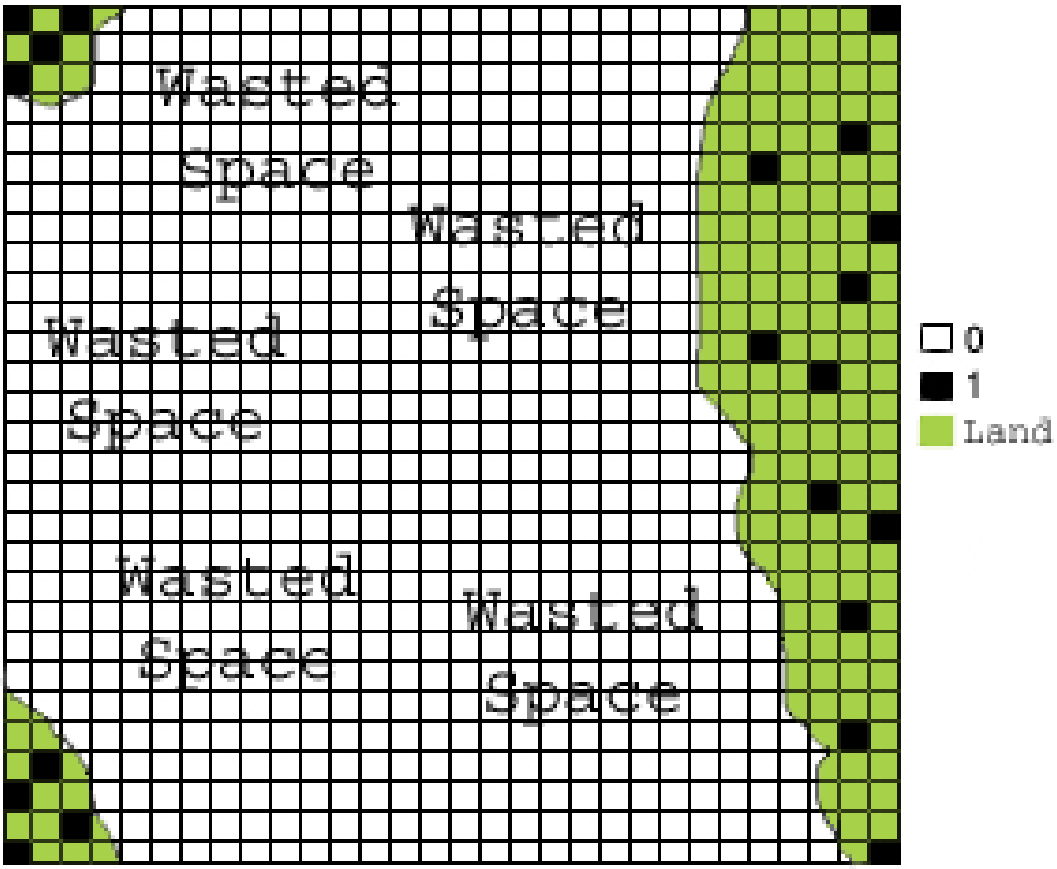


Figure 2-3. Wasted Memory Example

Second Generation GBs

To solve the wasted memory issue in the first generation, a technique called "bit-masking" is used to allow one byte to represent 8 bits within the GB. This is ideal and reduces the overall size of the GB by a factor of 8. An insignificant amount of space (a maximum of 7 bits per GB) is wasted at the end of the allocated memory when the area of the GB (in bits) is not divisible by 8.

The first generation GB array *B* is replaced by the more efficient GB array *B'*:

$$B' = \text{(unsigned char*)malloc}\left(\left\lceil \frac{(NR)(NC)}{8} \right\rceil\right) \tag{2-9}$$

A bit-mask is created for each bit within a byte. By logically ANDing the appropriate mask with the byte, the state (set or clear) of any bit can be determined. The masks are declared in C as an array of hexadecimal values as follows:

```
char mask[] = {0x01, 0x02, 0x04, 0x08,
               0x10, 0x20, 0x40, 0x80};
```
(2-10)

and Figure 2-4 shows all 8 masks and how an individual bit is extracted.

.

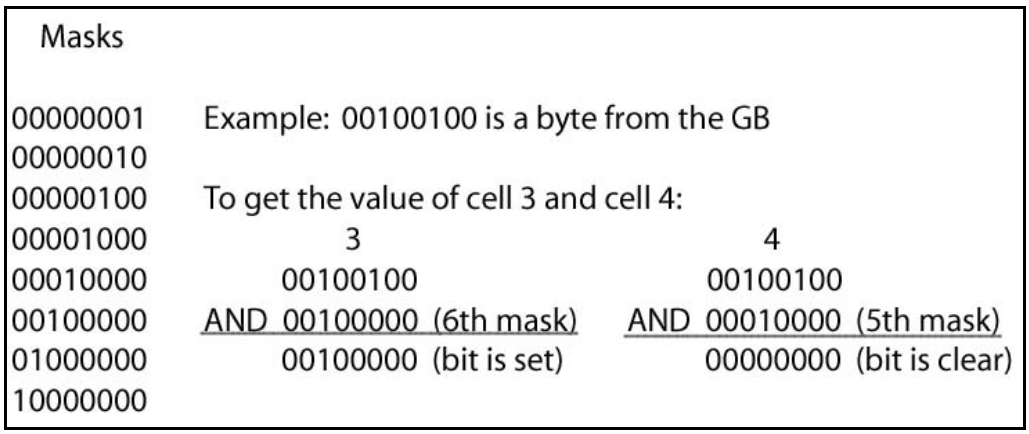


Figure 2-4. Bit-masking (reverse endian)

This bit masking technique reduces the overall size of the GB by a factor of 8. Figure 2-5 shows the new GB.

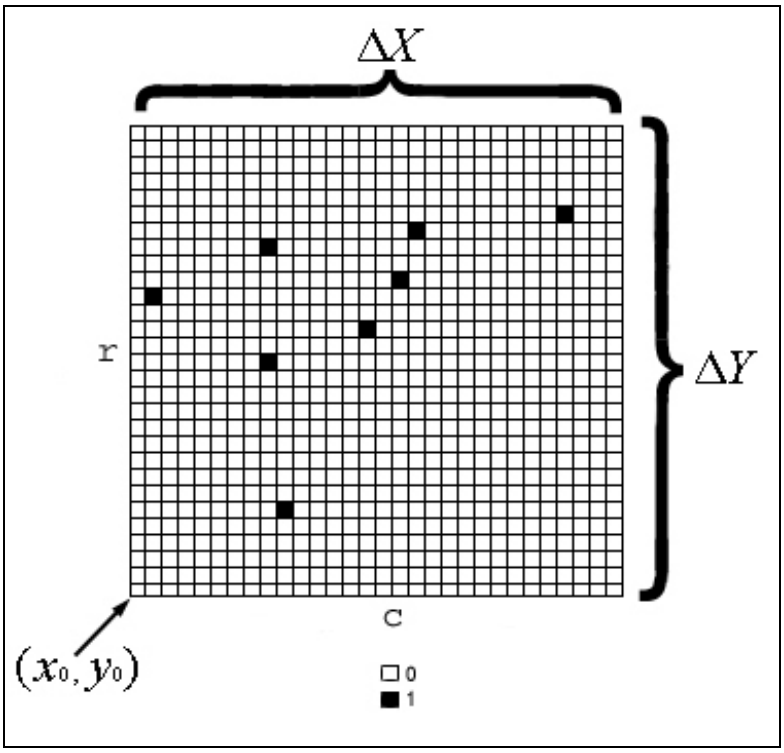


Figure 2-5. Second Generation GB

Third Generation GBs

The third generation of GB code attempts to solve the problem of wasted space when the ratio of set bits to cleared bits is low (Figure 2-3). The fields contained in the data structure (listed in section 3.1) are revised as follows:

1. Minimum longitude x_0
2. Minimum latitude y_0
3. Longitude resolution s_x based of the map scale of the data being represented
4. Latitude resolution s_y based of the map scale of the data being represented
5. Number of rows (*NR*)
6. Number of columns (*NC*)
7. Number of submaps (n)
8. Number of bytes per submap (p)
9. Array of unsigned char pointers (*B"*)

During processing, *NR* and *NC* are required more often than x_1 and y_1. Thus, the values of *NR* and *NC* are initially calculated by equations (2-1) and (2-2) and stored permanently in the data structure; x_1 and y_1 are inversely calculated from *NR* and *NC*, as needed.

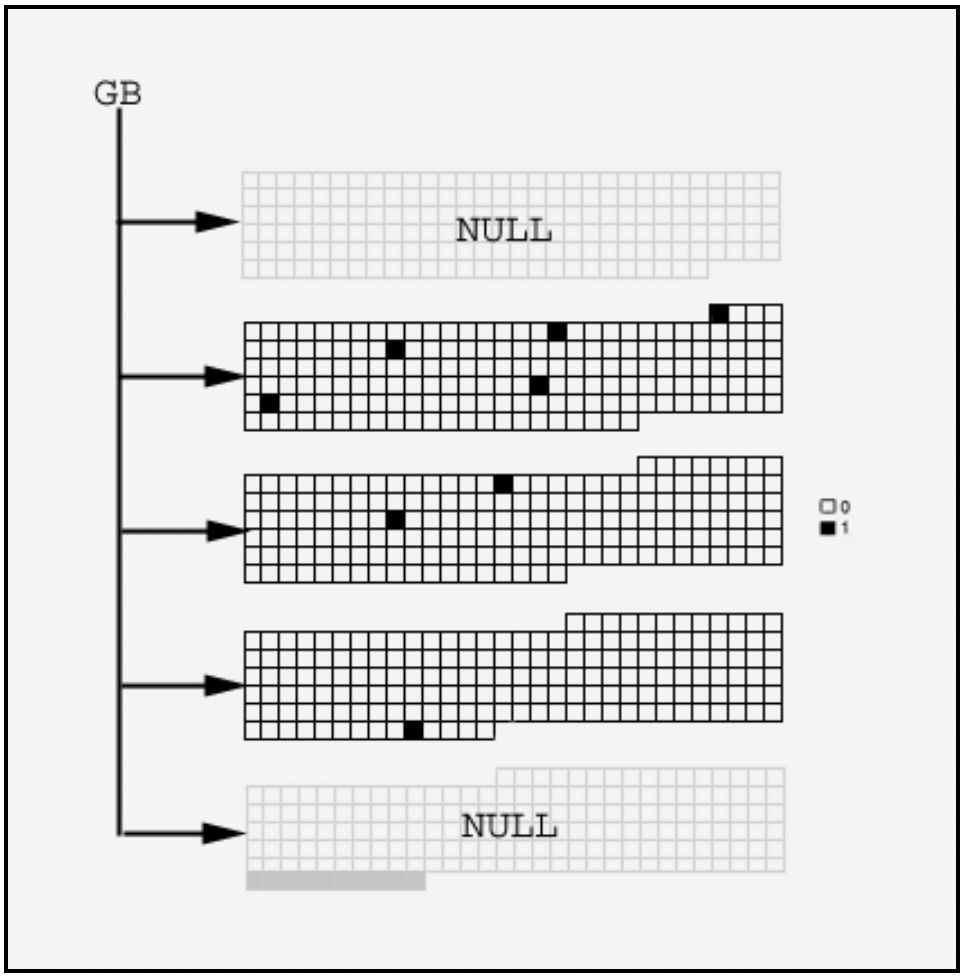


Figure 2-6 . GB broken into strip-like submaps. In this example, NR = 29, NC = 30, p = 22, n = 5, e = 0, and overall savings = 42.75 bytes.

In an effort to minimize wasted space, the bitmap is broken into strip-like submaps (see Figure 2-6), where m_i represents the i^{th} submap in the GB. Memory for a submap is only allocated the first time one of its bits is set. In Figure 2-6, for example, memory is not allocated for submaps m_1 or m_5 because no set bits fall within the first or fifth submap.

Letting $N = (NR)(NC)$, and having no *a priori* knowledge of the data set distribution, we estimate an appropriate number of submaps, n, to be:

$$n = \left\lfloor \frac{NR}{\sqrt{NC}} \right\rfloor \tag{2-11}$$

Note that if the GB is square, then NR = NC and n = the square root of one side of the GB. If the GB is not square, the number of strips will increase linearly with the number of rows (for a given number of columns), and the number of strips will decrease inversely to the square root of the number of columns (for a given number of rows).

The number of bytes per submap, p, is:

$$p = \left\lceil \frac{N}{8n} \right\rceil \tag{2-12}$$

Some bytes may be wasted in the last submap if two conditions are met: 1) $pn \neq \frac{N}{8}$ and 2) memory is allocated for the last submap m_n. The number of potentially wasted bytes, e, is calculated by:

$$e = U(m_n)(pn - \left\lceil \frac{N}{8} \right\rceil), \tag{2-13}$$

where U(m) is equal to 0 if memory for the submap is not allocated. U(m) is equal to 1 otherwise.

For example, in the GB shown in Figure 2-6, 1.25 bytes would have been wasted if the last submap m_5 had been used. However, since the last submap was not used, U(m_n) = 0 and, therefore, $e = 0$.

The net memory savings is calculated by:

$$savings = \frac{N}{8} - p\sum_{i=1}^{n} U(s_i), \tag{2-14}$$

which comes to 42.75 bytes in this example, a significant savings compared to the last generation.

This solution is an improvement, but not optimal when the GB represents a disjoint geographic area, as in Figure 2-3. Because the submaps are strips that run the entire width of the GB, there would be no space saving in this case. Also, once a submap's memory is allocated, there is no easy mechanism to free the memory when all the bits are cleared by logical operations, such as ANDing. Finally, this calculation is inefficient for small GBs, for which it would be more efficient to eliminate submaps and simply use one bitmap.

Fourth Generation GBs

The fourth and current GB scheme reduces the memory allocation further by breaking the GB into tile-like submaps (Figure 2-7) instead of strips. The underlying reason is based on Tobler's First Law of Geography, which states, "everything is related to everything else, but near things are more related than distant things" (Tobler, 1979). In the newest structure, submaps are 2D tiles of equal size. The total number of bits per submap is calculated from the number of submap rows and columns. The submaps are actually pointers to byte arrays, in which individual bits are masked from the bytes as previously described in this paper.

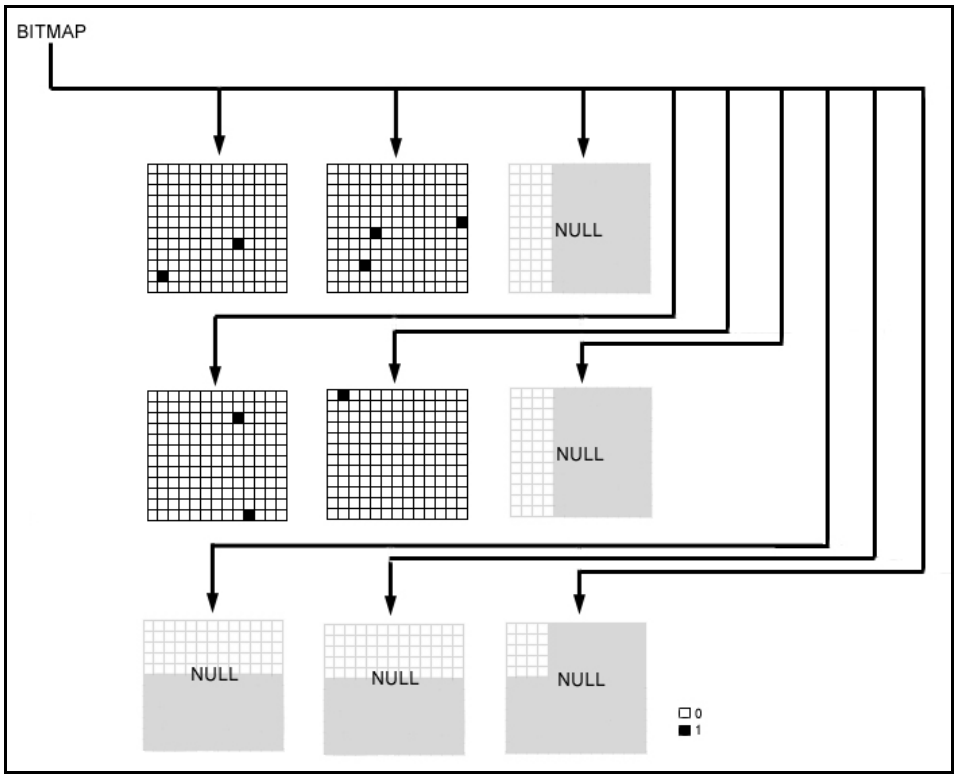


Figure 2-7. Tiled GB (Note: There are also 4 bits of wasted space per tile that are not shown in the figure)

The fields contained in the data structure (listed in section 3.3) are revised for this generation as follows:

1. Minimum longitude x_0
2. Minimum latitude y_0
3. Longitude resolution s_x based of the map scale of the data being represented
4. Latitude resolution s_y based of the map scale of the data being represented
5. Number of rows (*NR*)
6. Number of columns (*NC*)
7. Number of total set bits (*K*)
8. Number of submaps (*n*)
9. Number of submap rows (NR_S)
10. Number of submap columns (NC_S)
11. Array of integers holding the count per submap (*k*)
12. Array of unsigned char pointers (*B"*)

To determine the number and size of the submaps, a "best fit" algorithm is used. The algorithm attempts to optimize the number, size, and shape of the submaps, to minimize wasted space and processing time.

First, the total size of the data that the GB will represent is calculated. If the entire data set at the given resolution can be represented by a GB of 256 bits or less, then the GB is not tiled, and the following assignments are made:

$NR_S = NR$;
$NC_S = NC$;
$n = 1$;

For GBs larger than 256 bits, a "best-fit" algorithm is applied to determine the optimum number and size of submaps. If the submap size is too small, the algorithm will require considerable memory overhead (e.g., a pointer to each submap). If the submap size is too big, there is wasted space. The algorithm must pick a best-fit submap size, with no *a priori* knowledge of the data set except the assumption that the GB was chosen appropriately for the data set.

To determine this best-fit submap size, we first assume that the data has a positive spatial autocorrelation (Lu, 2001; Scott and Lloyd, 1997), and more specifically that the proximity of two geospatially close points tends to be approximately proportional to the proximity of two geospatially close sets of points (i.e., clusters). This theory seems to be supported by Miller (2004), who suggests that relationships and interactions among near entities (e.g., city dwellers) can produce regular geographic patterns or structures both locally (e.g., distribution of people within cities) and globally (e.g., distribution of cities within countries).

Given these theories, we assume that the number of bits per submap (N_S) is equivalent to the number of submaps (n) in the GB. Noting that $N_S = (NR_S)(NC_S)$ and $n = N / N_S$, where $N = (NR)(NC)$, it follows that:

$$N_S = N / N_S$$
$$N_S = \sqrt{N}$$

That is, a first estimate of the best-fit submap size, N_S, is the square root of the number of bits in the GB. Unfortunately, this submap size does not account for the fact that typical computer systems only allocate memory in 8-bit (1-byte) chunks. We describe an algorithmic approach to solving this problem.

Let $N_{S'}$ be a second estimate of N_S for a given GB. The algorithm first computes a good estimate of n, n':

$$n' = \sqrt{(NR)(NC)} \qquad (2\text{-}15)$$

The algorithm then iterates with the number of bits per submap, i, ranging from 1 to N, computes the number of submaps and the submap dimensions (NR_S, NC_S), and tests to determine if these dimensions produce the best fitting submap.

If $\sqrt{i}$ is an integer, then the submaps will be square (i.e., $NR_S = NC_S = \sqrt{i}$). Otherwise, two submap shapes are considered for the current value of i:

1) $NR_S = \left\lfloor \sqrt{i} \right\rfloor$ and $NC_S = \left\lfloor \frac{i}{NRs} \right\rfloor$

2) $NR_S = \left\lceil \sqrt{i} \right\rceil$ and $NC_S = \left\lfloor \frac{i}{NRs} \right\rfloor$

With each iteration of the algorithm, an error value is calculated for all possible submap dimensions. This error value is a weighted sum of the amount of wasted space, the resultant number of submaps, and the difference between the resultant number of submaps and *n'*. After all iterations, the submap dimensions that produced the lowest error are used for the number of submap columns and rows.

Repeated tests confirmed that the best-fit submap dimensions rarely occur at the extreme values of *i*. To reduce the number of iterations (resulting in a more efficient algorithm), *i* is iterated from i_0 to i_1, instead of from 1 to *N*, where:

$$i_0 = 0.5 * \sqrt{(NR)(NC)} \quad (2\text{-}16)$$

$$i_1 = 1.5 * \sqrt{(NR)(NC)} \quad (2\text{-}17)$$

The algorithm was tested on the same data set as before and a submap size of 12 rows by 13 columns was calculated (Figure 2-7), in which 4 submaps are necessary to represent the data, resulting in a savings of 31 bytes.

In this generation of the GB structure, an overall count of set bits is maintained to facilitate comparisons among two or more GBs. More importantly, the number of bits set within a submap is maintained such that if this "submap count" falls to zero (i.e., no set bits), the memory for that submap is freed. This allows the GB to grow and shrink dynamically and allows for bitmap compression (Goyal et al., 1999). It also enables internal software error checking to ensure the GB is not corrupt.

Extending GBs to three dimensions or higher is possible because the GB data structure contains a pointer field that points to itself. This allows link-lists of GB's to be dynamically allocated to form 3D or higher dimensions.

GB Operations

GBs are extremely useful in solving problems pictorially using Boolean operations, such as ANDing and XORing (Goyal, 1999), and morphological operations like shape filling, outlining, erosion and dilation (Russ, 2002). One example of how GBs are used to solve such problems is given in "POS Polyline Smoothing: Reduction of Polyline Vertices", by Layne et al. (2004) in which GBs are used to quickly determine if a vertex of a filled polygon should be dropped or "smoothed". The interior of the polygon to be smoothed contains points of significance (POS), and a vertex can only be removed if no POS lie outside the newly formed polygon. The decision

to remove a vertex reduces to triangles. Consider the following example, noting that the interior of the polygon is below the polyline (Figure 2-8).

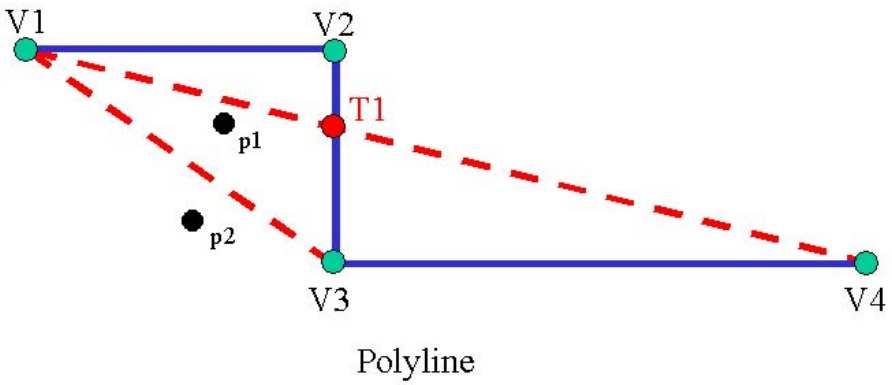


Figure 2-8. Polyline Smoothing Example

In the example in Figure 2-8, the algorithm would first test whether or not there are POS inside the triangle defined by V1, V2, and V3. If not, then vertex V2 is removed; otherwise, V2 is preserved. For example, if the point, p1, is a POS, V2 would not be removed because the resulting polygon would no longer contain all the POSs.

This problem can be represented pictorially, and logical operations can be performed between GB's of differing sizes and locations (Figure 2-9). All POS are represented by a GB (gray cells in Figure 2-9, where memory is only allocated for regions outlined in red). Each triangle under consideration is represented by a small GB (blue area of Figure 2-9), where all bits representing the triangle are set. The logical ANDing between the POS GB (gray) and the triangle GB (blue) is driven by the smaller triangle GB, which results in a very efficient operation. If the resulting GB contains any set bits, as in Figure 2-9, it is immediately known that the triangle contains at least one POS, and thus the vertex under consideration in this triangle (e.g., V2 in Figure 2-8) is not removed.

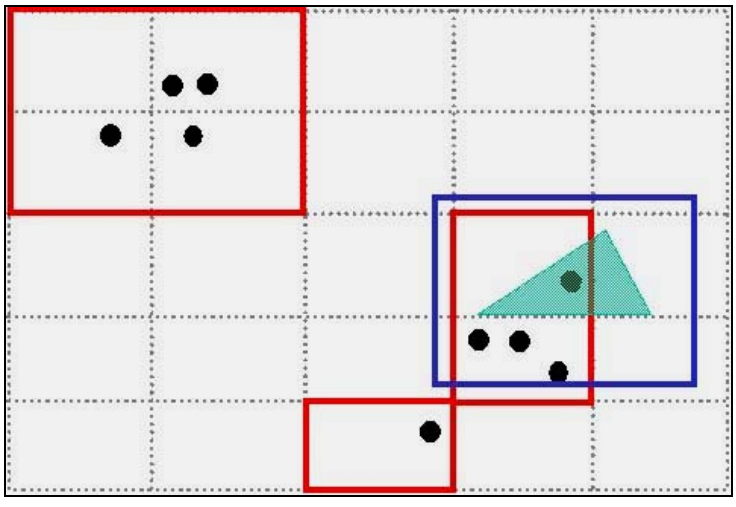


Figure 2-9. ANDing in GB Space

The use of GBs and associated logical operations results in a smoothing algorithm that is much faster and more memory efficient than traditional methods, which require geographical comparisons and Euclidian distance calculations.

Experimental Runs

Currently, the POS algorithm is being used by Mine Warfare (MIW) analysts at the Naval Oceanographic Office (NAVOCEANO) to smooth polygons produced by a unique clustering algorithm developed by the author (Gendron et al., *submitted*), in which spatially close data is clustered into bounding polygons. The clustering and smoothing algorithms are used in this paper to quantitatively illustrate how the use of GBs can improve processing time. The GB method was compared with two alternative methods of determining whether POS fall within a given triangle: Euclidean geometry and Sparse Matrix (SM) data structures.

Note that the author did not compare GBs with other compression methods, such as run-length encoding or Lempel-Ziv (Ziv and Lempel, 1977), because these algorithms require a decompression step to recover the original spatial information before performing other geospatial operations (e.g., POS smoothing). GBs, on the other hand, can be used to both compress the data and perform geospatial operations on the compressed subset of data (e.g., with many 0's missing).

The following sections discuss the Euclidean Geometry and SM data structures, which were compared with GBs for POS smoothing.

Euclidean Geometry Method

The Euclidean geometry method (O'Rourke, 1997) determines where a given POS is, relative to the triangle of interest, as follows:

1) The POS position is set as the origin.
2) The algorithm checks whether any of the triangle's vertices is the POS. If so, the POS is considered to be inside the triangle and the algorithm stops. If not, the algorithm continues with step 3.
3) The algorithm determines whether any edges of the triangle cross the x-axis. If not, the POS is outside the triangle and the algorithm stops. If an edge does cross the x-axis, the algorithm continues with step 4.
4) The algorithm keeps track of how many edges cross the x-axis, and whether the edges cross to the left or to the right of the POS:
    a. If the number of crossings to the left is an odd number and the number to the right is odd (or if both are even), the POS is on the edge of the triangle and is considered to be inside.
    b. Otherwise, if the number of crossings to the right is odd, the POS is inside the triangle.
    c. Otherwise, the POS is outside the triangle.

Fourth generation GBs (discussed earlier in this chapter) attempt to reduce storage and processing time by only operating on non-zero values (e.g., 1's), which minimizes the total number of operations required while still maintaining geospatial relationships among the data elements. Since the geographic data that the author typically processes is often characterized by

large expanses of empty space (e.g., 0's over ocean areas), GBs have the potential to significantly reduce storage and processing time. Another way to accomplish this is using SM data structures, which also store only non-zero values in a data set.

Sparse Matrices (SM) Data Structure

The SM data structure (Steward and Leyk, 1994) used for the comparison tests is depicted in Figure 2-10. SM contains two memory pointer arrays of length m, where m is the number of rows in an m x n matrix.

The first memory pointer array, or Value Array, has m pointers, each of which points to a row array representing a corresponding row in the matrix containing at least one non-zero element. If a given row in the matrix contains all zeroes, then the corresponding memory pointer, at that row location, will have a value of NULL. The number of elements in any given row array is equal to the number of non-zero values in the corresponding row of the matrix. For example, in Figure 2-10, the 2nd row of the matrix (shown on the right) contains 3 non-zero elements; therefore the 2nd row array in the Value Array of the SM structure has length 3 and contains all the non-zero elements for that row.

The second memory pointer array, or Location Array, also has m pointers and contains the same number of row arrays as the Value Array. The elements of these row arrays contain the indices, starting at 0, of the elements in the matrix they represent. For example, in Figure 2-10, the three non-zero elements in the 2nd row of the matrix are located at indices 1, 3, and 8; therefore, the 2nd row array in the Location Array of the SM structure has length 3 and contains these indices.

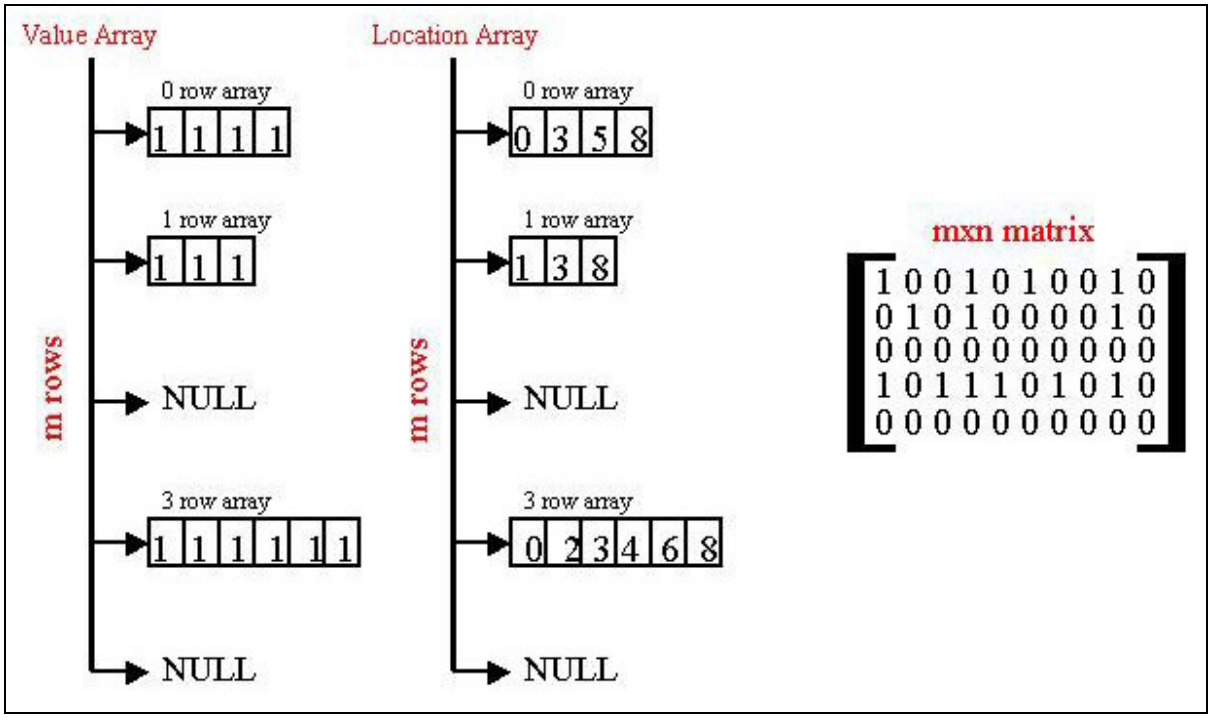


Figure 2-10 . Sparse Matrix (SM) Data Structure.

If an element in the matrix changes value, then the corresponding row arrays in both the Value and Location Arrays must be rearranged. To retrieve a value from the matrix by location (i.e., row, column), a binary search is performed on the corresponding row array in the Location Array to retrieve the value's column index, which is then used to retrieve the actual data value (stored in the Value Array). These operations require indirect indexing of the SM, which can be quite time-consuming, and can negate many of the benefits of SM data structures (Computational Science Education Project, 1995). Alternatively, the author has demonstrated that operations to change and retrieve information from GBs (for the POS smoothing example) are more efficient and faster than with SM structures, because GBs support direct indexing.

Methods

As described above, the author uses the GB method in the POS smoothing algorithm to determine if any POS lies within triangles formed by adjacent vertices. To compare the GB with alternate methods, the POS smoothing algorithm was modified to use two versions of the Euclidean geometry method and two versions of SM data structures in place of the GB method. C code to perform the Euclidean geometry methods was obtained from O'Rourke (1997). The Meschach SM C library was downloaded from Steward, D.E. and Z. Leyk (1994) and used to implement the SM data structure.

The first comparison (Euclidean1) checks every POS to determine whether it falls within the triangle. This information is used to decide if the current vertex can be removed (as described in the GB Operations section, above), and ultimately how many POS are interior to the triangle. This produces identical results to the GB method. By knowing how many POS are interior, the algorithm could remove a vertex only if, for example, less than 5% of the POS are interior to the triangle. The SM1 comparison uses SM data structures to find all the POS that are interior to the triangle (similar to the Euclidean1 method).

The Euclidean2 comparison stops checking POS as soon as the first interior POS is found. This will reduce processing time over Euclidean1, but the total number of interior POS is lost, which makes the algorithm less powerful. The SM2 comparison uses SM data structures to find the first interior POS (similar to Euclidean2).

Time trials were performed to compare smoothing using the GB method verses each of the four other methods. The tests were performed on a Pentium III 800 MHz PC with 256 MB of RAM running Linux.

Each method was tested five times on random data sets of 600, 3000, 10000, and 65000 POS with a clustering radius of 50 and 125 meters. The clustering algorithm generates a different number of polygons, with a different number of vertices, depending on the size of the clustering radius.

Results

Tables 2-2 and 2-3 list – for each trial – the number of POS input to the algorithm, the number of polygons generated, the total number of triangles that the POS smoothing algorithm had to process for each data set, and the average time (in seconds) of the five runs for each algorithm.

Table 2-2. Test results of smoothing with GB, Euclidean1, Euclidean2, SM1, and SM2 (Cluster Radius of 50 meters).

				Average time (s)				
# POS	# poly (r=50m)	# vertices	# triangles	GB	Euclidean 1	Euclidean 2	SM1	SM2
600	382	2,650	5,751	7.69	123.44	68.13	216.11	147.90
3,000	1,067	11,893	22,509	40.68	1822.16	1047.30	1967.24	1759.38
10,000	104	3,258	5,019	17.09	666.02	504.90	762.02	497.12
65,000	2	1,000	1,288	31.78	281.98	272.06	274.21	262.64

Table 2-3. Test results of smoothing with GB, Euclidean1, Euclidean2, SM1, and SM2 (Cluster Radius of 125 meters).

				Average time (s)				
# of POS	# poly (r=125m)	# vertices	# triangles	GB	Euclidean 1	Euclidean 2	SM1	SM2
600	80	1,536	2,832	30.09	75.57	49.25	177.62	161.45
3,000	4	563	787	32.36	74.48	62.26	102.77	98.57
10,000	1	465	567	48.85	91.97	85.27	124.96	121.74
65,000	1	335	425	104.81	142.74	133.92	180.92	160.56

Figures 2-11 and 2-12 are plots showing the time (in seconds) each algorithm required to smooth the resulting polygons produced by the clustering algorithm at two clustering radii. The GB method performed extremely well over both Euclidean methods and both SM methods when the polygons were small (50 m clustering radius) with many vertices (i.e., small triangles). In fact, the GB method processing times were extremely fast regardless of the number of points (Figure 2-11).

For a cluster radius of 125 m (Figure 2-12), the difference in processing time among the three methods was not as pronounced, but the GB method clearly took less time for all four data sets. However, the GB method was not as efficient with a clustering radius of 125 m as it was with a radius of 50 m, because the polygons in the 125 m case are usually much larger with fewer vertices (i.e., larger triangles). The GB method must create larger GBs and "fill" larger triangles prior to the ANDing operation.

As you can see in these plots, GB outperformed SMs in the POS smoothing algorithm. In many cases, the Euclidean geometry method also outperformed SMs. This is because SM data structures require more overhead (storing both indices and nonzero matrix entries) than simpler data structures (e.g., arrays), and data operations are usually slower with SM structures, due to the indirect indexing (Computational Science Education Project, 1995).

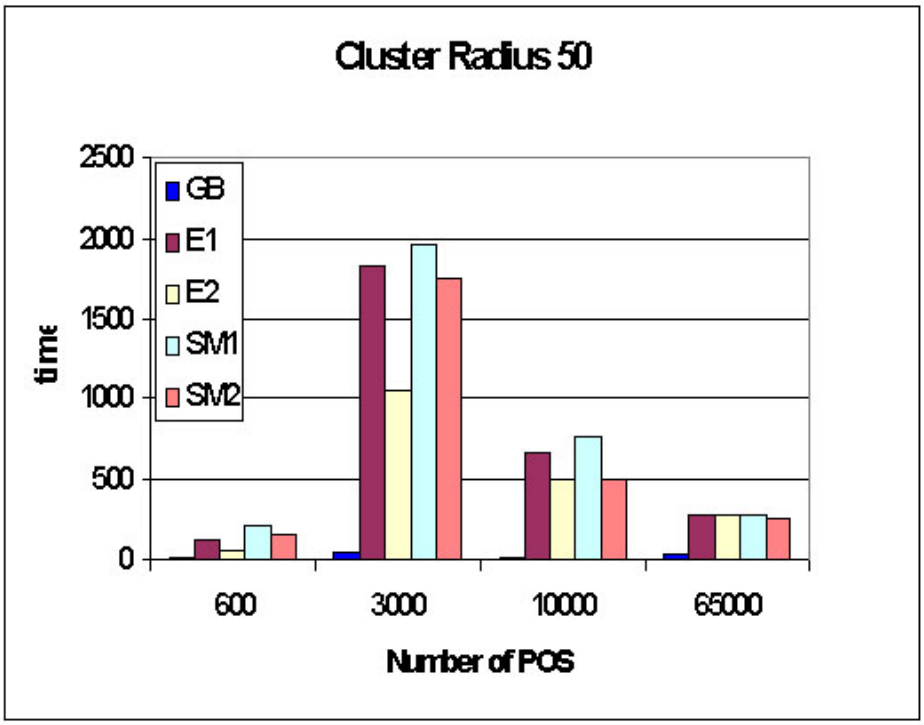


Figure 2-11. Time (in seconds) required by all algorithms to smooth polygons resulting from the four data sets with a clustering radius of 50 meters.

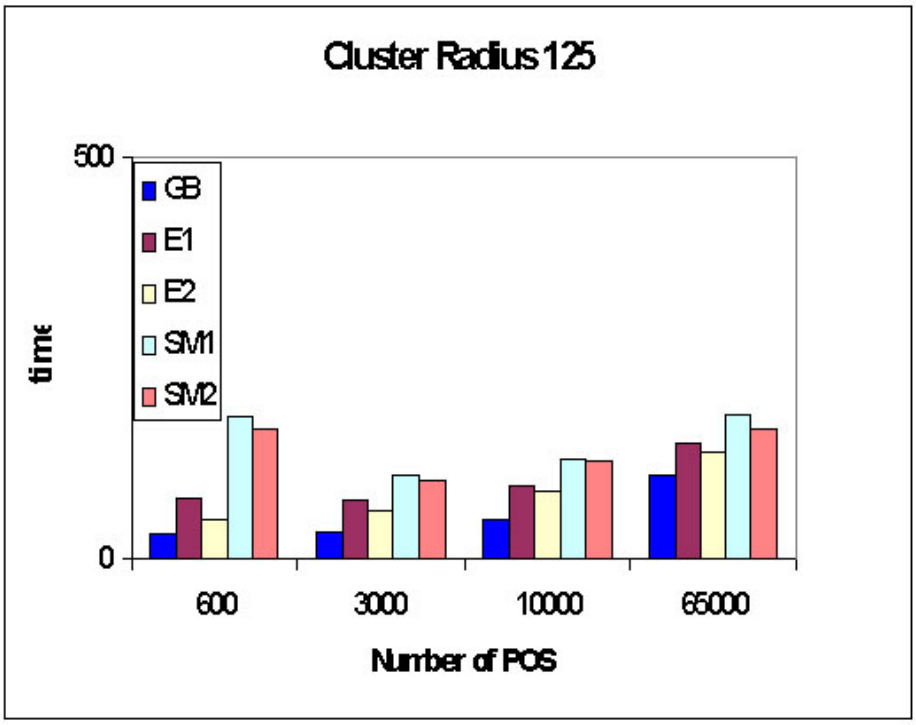


Figure 2-12. Time (in seconds) required by all algorithms to smooth polygons resulting from the four data sets with a clustering radius of 125 meters.

Conclusion

The author has taken the simple bitmap and linked each cell to a geospatial location to produce the GB data structure. The author has presented improvements to the design over many generations to produce a powerful, compact data structure. The fourth-generation tiled GB is excellent for solving certain complex computational problems at a low processor cost, and is a unique tool for solving problems graphically, such as those encountered during polygon smoothing and data clustering. Processing time and results for running a polygon-smoothing algorithm using 1) GBs, 2) Euclidean geometry, and 3) Sparse Matrices (SM) show that the GB implementation ran between 1.3 – 48.4x faster than either of the other methods, depending on the size and number of the polygons to be smoothed. GBs perform best when representing large geographical areas where there are a lot of clear bits. The processing speed advantages of the GB data structure is reduced when the represented data is denser as shown in the 50 m radius verses the 125 m radius examples presented during the polygon-smoothing tests. Future work with the GB includes further modifications to the structure to improve compression and reduce processing times. Also further comparisons between GBs and the SM data structure should be made.

Acknowledgements

This work was performed in support of the Naval Oceanographic Office (NAVOCEANO) and funded by the Space and Naval Warfare Systems Command (SPAWAR) through the Commander, Naval Meteorology and Oceanography Command (CNMOC) and ONR under Program Element 0603207N. The authors thank Captain Robert Clark (program manager, SPAWAR PMW 150), Dr. Edward Mozley (assistant program manager for data acquisition, SPAWAR PMW 150), Mr. Bruce Northridge (CNMOC) and Mr. Michael M. Harris (NRL) for their support of this project. We thank Mr. Jim Hammack (NAVOCEANO) for his recommendations concerning the application of this algorithm in support of NAVOCEANO requirements. Any mention of commercial products or the use of company names does not imply endorsement by the U.S. Navy. This paper is approved for public release; distribution is unlimited.

References

Baldwin, D. and G.W. Scragg (2004). *Algorithms and Data Structures: The Science of Computing*. Massachusetts: Charles River Media.

Computational Science Education Project (1995). *Numerical Linear Algebra*, Chapter 8.2 Sparse Matrices. http://csep1.phy.ornl.gov/la/la.html (accessed December 9, 2004).

Defense Mapping Agency (DMA, 1989). *Defense Mapping Agency Product Specifications for Arc Digitized Raster Graphics (ADRG)*. First Edition, DMA Aerospace Center, St. Louis, Missouri.

Downing, D. and M.A. Covington (1986). *Dictionary of Computer Terms*. Barron's Educational Series, New York: Woodbury.

Dutton, J. (1983). Geodesic Modelling of Planetary Relief. Proceedings of the *AutoCarto VI*, Ottawa.

Edelstein, H. (1995). Technology Analysis: Faster Data Warehouses. *Information Week*, December 4.

Gambino, F., G. Oriolo, and G. Ulivi (1996). A Comparison of Three Uncertainty Calculus Techniques for Ultrasonic Map Building. SPIE Proceedings on *Applications of Fuzzy Logic Technology III*, 2761.

Gendron, M., G. Layne and M. Lohrenz, (*submitted*). The Geospatial Bitmap (GB) Clustering Algorithm. Naval Research Laboratory, Stennis Space Center, Mississippi, submitted to the *Journal of Discrete Algorithms*.

Gendron, M., M. Lohrenz, Layne and Ioup (2004). Automatic Change Detection and Classification (ACDC) System. Proceedings of the *Sixth International Symposium on Technology and the Mine Problem*, Naval Postgraduate School, Monterey, CA. May 9-13.

Gendron, M., P. Wischow, M. Trenchard, M. Lohrenz, L. Riedlinger and M. Mehaffey (2001). *Moving Map Composer*. Navy Case Number 76,358. Patent Number 6218965, April 17.

Gonzalez, R.C. and R.E. Woods (2002). *Digital Image Processing*. Second Edition, New Jersey: Prentice-Hall.

Goyal, K.B, K. Ramamritham, A Datta and H. Thomas (1999). Indexing and Compression in Data Warehouses. *Design and Management of Data Warehouses, 11*.

Howe, D. (2004). The Free On-line Dictionary of Computing, http://wombat.doc.ic.ac.uk/foldoc/index.html

Jensen, F.B., W.A. Kuperman, M.B. Porter and H. Schmidt (2000). *Computational Ocean Acoustics*. New York: Springer-Verlag.

Layne, G., M. Gendron and M. Lohrenz (2004). POS Polyline Smoothing: Reduction of Polyline Vertices. Proceedings of the *Tenth International Conference on Industry, Engineering, and Management Systems*, Cocoa Beach, FL. March 15-17.

Lohrenz, M., S. Myrick, P. Wischow, and M. Trenchard (1992). An Overview of the U.S. Navy Compressed Aeronautical Chart Database. Proceedings of the *1992 International Conference of the Royal Institute of Navigation and the German Institute of Navigation*, London, England. November.

Lohrenz, M., M. Trenchard, S. Myrick, P. Wischow, and L. Riedlinger (1993). *The Navy Tessellated Spheroid Map Projection System: A Comprehensive Definition*. NRL/FR/7441-92-9408, August.

Lohrenz , M., P. Wischow, H. Rosche, M. Trenchard, and L. Riedlinger (1990). The Compressed Aeronautical Chart Database: Support of Naval Aircraft Digital Moving Map Systems. Proceedings of the *IEEE PLANS'90 Conference*, Las Vegas, NV. March.

Lu, Y (2001). Spatial Cluster Analysis for Point Data: Location Quotients verses Kernal Density. [http://www.ucgis.org/oregon/papers/lu.htm].

Miller, H. J. (2004). Necessary Space-time Conditions for Social Interaction. Proceedings of the *100 th annual meeting of the Association of American Geographers,* Philadelphia, Pennsylvania.

Miller, H., S. Coughlan, T. Fetterer, R. Guidry, M. Lohrenz, M. Trenchard, P. Wischow, K. Shaw, and S. Carter (1994). *DMAP Technical Review of Vector Quantization (VQ) Decompression for the National Imagery Transmission Format Standard*. NRL/MR/7441-94-7541.

Myrick, S., M. Lohrenz, M. Gendron, M. Trenchard and L. Riedlinger (2003). Moving Map Composer: A Tool For Consolidating Tasks Associated With The Design, Creation, and Management Of Digital Aeronautical Chart Products For Navy Aircraft. Presented at the *7th*

World Multiconference On Systemics, Cybernetics and Informatics, Orlando, FL. NRL/PP/7440-03-1016, July 27-30.

Myrick, S., M. Lohrenz, and P. Wischow (1992). Reproduction of Custom Colors for the Navy's Compressed Aeronautical Chart. *The 8th International Congress on Advances in Non-impact Printing Technologies*, Williamsburg, VA. October.

O'Rourke, J. (1998). *Computational Geometry in C (Second Edition)*. New York: Cambridge University Press.

Ozbutun, C. (1997). Bitmap Indexing in ORACLE 7.3 and 8.0. Proceedings of the *European Oracle Users Group (EOUG97) Conference*, 332.

Russ, J. C. (2002). *The Image Processing Handbook*, Fourth Edition, New York: CRC Press.

Scott, L., and W. J. Lloyd (1997). Spatial Analysis in a GIS Environment: Employment Patterns in Greater Los Angeles, 1980 - 1990, Proceedings of the *University Consortium for Geographic Information Science*, Orono, Maine (electronic publication: http://www.spatial.maine.edu/ucgis/testproc/scott_l/scott.html).

Snyder, J. (1987). *Map Projections: A Working Manual*. USGS Professional Paper 1395. Washington, DC: United States Government Printing Office.

Steward, D.E. and Z. Leyk (1994). Meschach C Libray: Version 1.2b. *School of Mathematical Sciences*, Australian National University, Canberra, Australia. http://www.netlib.org/c/

Tobler, W. (1979). Cellular Geography. *Philosophy in Geography*, Dordrecht: D. Reidel Publishing.

Wischow, P. and M. Lohrenz (1998). *Compressed Aeronautical Chart Access Software*. NRL/MR/7441-97-8073, July.

Yoon, J., V. Raghavan and V. Chakilam (2001). BitCube: Clustering and Statistical Analysis for XML Documents. *Thirteenth International Conference on Scientific and Statistical Database Management*, Fairfax, Virginia, July 18-20.

Ziv, J. and A. Lempel (1977). A Universal Algorithm for Spatial Data Compression. *IEEE Transactions on Information Theory*, 23:3.

Chapter 3 - The Geospatial Bitmap (GB) Clustering Algorithm

Marlin L. Gendron, Geary J. Layne, Maura C. Lohrenz
Naval Research Laboratory (NRL), Code 7440.1, Stennis Space Center, MS

Abstract

The author developed the Geospatial Bitmap (GB) clustering algorithm to be a fast, efficient, and repeatable method of clustering into bounded regions any set of unique elements, defined by any coordinate system, lying in n-dimensional space. The clustering is accomplished by mapping each element into one or more pre-defined geometric "expansion shapes" using GBs. Any mapping resulting in a shape with dimension less than or equal to the dimension of the element space can define a cluster. The clustering algorithm was tested on several data sets to determine which (if any) of the algorithm parameters have a significant impact on processing time. Parameters tested include the number (n), resolution (r), distribution (d) and ordering (o) of input data points, and the area (a) and shape (s) of the expansion region. Processing time was shown to be affected by n, a, and r, but not by s, d or o. The author suggests that the GB clustering algorithm is a good choice for applications that require automated clustering in real-time: the algorithm is single-pass (i.e., non-iterative); it does not require seed points; and the ordering of elements has no effect on the resultant clusters.

Introduction

Clustering is defined by Jain et al. (1999) as "the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters)". Clustering algorithms have been developed in support of various disciplines, including biology (e.g., clustering bacterial growth), physics (e.g., clustering high-energy particles), demographics (e.g., clustering populations), medicine (e.g., identifying clusters of tumors), and information technology (e.g., data mining/compression/sorting, image classification/segmentation/retrieval). The patterns or features to be clustered can be quantitative or qualitative (Jain et al., 1999): quantitative feature types include continuous values (e.g., weight), discrete values (e.g., a count), and interval values (e.g., length of time). Qualitative feature types include nominal (e.g., color) and ordinal (e.g., "big", "medium" and "small"). The examples described in this paper use continuous values {latitude, longitude} to cluster features in geographic space. However, the author's clustering algorithm works equally well on features represented by two or more dimensions of other quantitative and qualitative types, such as {latitude, longitude, color} or {color, size, shape}.

Clustering can be thought of as an unsupervised process because there are no predefined classes (Halkidi et al., 2002). Clustering methods can be classified as either hierarchical or non-hierarchical, as summarized in Barnard (1996) and Downs (2001). Hierarchical methods create clusters within clusters. A bottom-up hierarchy clusters the most "alike" points, and then clusters those clusters, and so on until there is only one cluster. It is helpful to think of the result of this method as a tree, in which the leaf nodes are the points, and the sub-trees at each level

define the clusters. A user can choose to terminate a hierarchical algorithm at any given level to result in a satisfactory number of clusters.

There are several methods of hierarchical clustering, often referred to as *linkage* algorithms. Commonly used single-link algorithms, including SLINK (Sibson, 1973) and Minimum Spanning Tree (Cormen et al., 2001), tend to produce long chains or clusters that are ellipsoidal in nature. The hierarchical agglomerative clustering method (HACM) attempts to cluster objects into hierarchically nested clusters (Day and Edelsbrunner, 1984; Voorhees, 1985a). Although HACM performed well in comparative studies by Voorhees (1985b), it can be quite slow when clustering large data sets. A hierarchical "fuzzy clustering" method (Höppner et al., 1999) forms clusters in which most vectors exhibit a high degree of membership to one cluster (Halkidi et al., 2002.)

Non-hierarchical methods cluster all points in a region of interest in a single iteration. If the number of resulting clusters is not ideal, then the method is altered (e.g., a different seed point is used) and reapplied. There are three major non-hierarchical methods: Single-Pass, Nearest Neighbor, and Relocation (Downs and Barnard, 2003). Single-pass algorithms, such as the cover coefficient algorithm of Can (Can and Ozkarahan, 1990) tend to be simple to implement and require only one pass, but cluster results will differ depending on the order of the input data. Relocation can accommodate larger data sets than most other methods, but converging to a solution can be time-consuming.

The Geospatial Bitmap (GB) clustering algorithm presented in this paper could be used repetitively to create a hierarchical cluster structure, but it is actually a single-pass, non-hierarchical method similar to Nearest Neighbor. Unlike the GB clustering algorithm, however, Nearest Neighbor must calculate and compare the distances between every pair of elements in the data set to determine which elements should be clustered together. By using GBs to cluster elements, we have developed a method that is an order of magnitude faster, less computationally intensive, and requires much less computer memory than Nearest Neighbor. Unlike Single-Pass methods, such as the Leader algorithm (Hartigan, 1975), the ordering of elements input to the GB algorithm has no effect on the resulting clusters. Unlike Relocation methods, such as K-means (Hartigan and Wong, 1979), the GB algorithm does not require seed elements to initiate the clustering process.

Geospatial Bitmaps

The clustering process relies on a GB technique (Gendron et al., 2001) patented by the author et. al. Simple bitmaps – with a depth of one bit per pixel – are binary structures, known as binary images in image processing (Russ, 2002), in which bits are turned on (set = 1) or off (cleared = 0). The index of each bit is unique and denotes its position relative to the other bits in the bitmap. The patent describes a 2D bitmap (e.g., Figure 3-1) in which a bit is indexed by column (x) and row (y). In 3D, a bit could be indexed by column (x), row (y) and depth (z). The author extended this concept to construct GBs in which every bit represents an object at some unique location in n-D space. A set bit indicates that some object of interest exists at that location, accurate to within the resolution of the bitmap. A cleared bit indicates the absence of any object at that location. Although a GB can be defined for an entire finite space, memory is only

allocated – dynamically – when groups of spatially close bits are set, resulting in a compact data structure that supports very fast Boolean and morphological operations.

Binary images have been used in combination with clustering techniques (e.g., Hobby and Ho, 1997; Ho and Nagy, 2000; Yoon et al., 2001), but GBs have never been used to drive the clustering directly. There are similarities between the author's GB clustering technique and cluster analysis by binary mathematical morphology, which is iteratively applied to detect cluster cores and eliminate irrelevant details in the cluster shapes, as is the case in a method developed by Postaire et al. (1993). Postaire's method is useful in reconstructing noisy or degraded images (e.g., scanned text), but it assumes there is a core of more significant points surrounded by and interspersed with less significant points (e.g., noise). Postaire assigns the less significant points to clusters using the Nearest Neighbor algorithm, which is order-dependent (i.e., resulting clusters depend on the order in which outlying points are input to the algorithm). The author's clustering algorithm assumes that all points are significant and the clustering is order-independent.

The GB Clustering Algorithm

The GB clustering algorithm provides a fast and efficient way to cluster any set of unique data elements in N-dimensional space into bounded polyhedron regions (in 2D, the regions are simply polygons). In this section, we present a derivation to show how data elements of a set D are represented in a GB, and then "expanded" by a specified expansion mapping, P'. The expansion produces "bounded" regions or clusters, which are stored in a new GB. When the expansion mapping is uniform, this process is equivalent to dilation of a structuring element, as defined in mathematical morphology (Gonzales and Woods, 2002). The author's process differs in that the expansion mapping does not have to be uniform. For example, given *a priori* knowledge of the data set, the diameter of the expansion shape could depend on the distance of the point of interest from some specified origin.

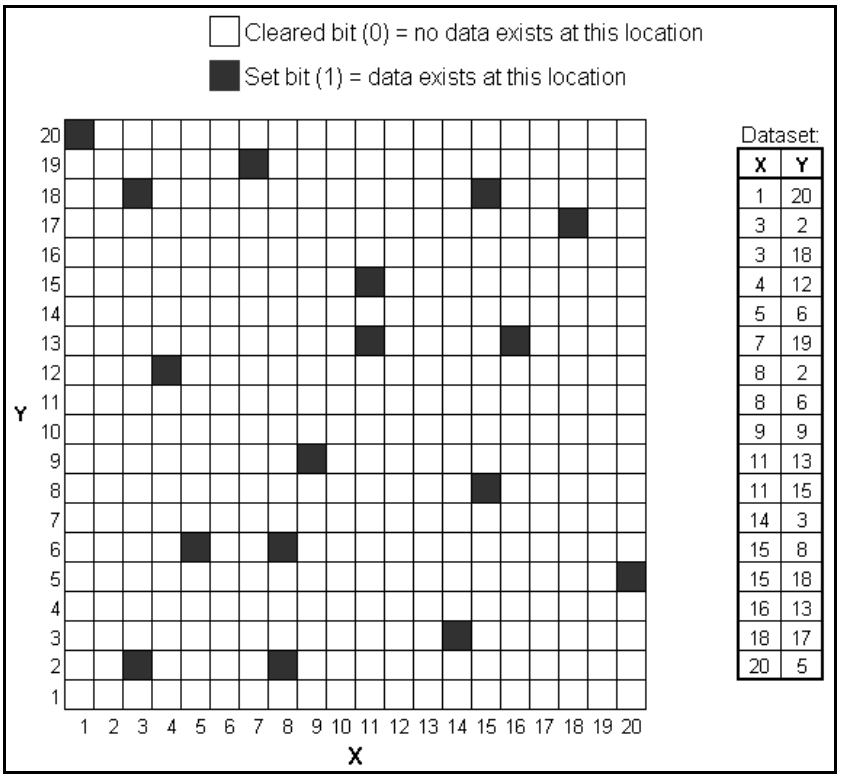


Figure 3-1. Example of a 2D bitmap.

The algorithm requires five initial arguments:

1. A set D of unique data elements.
2. A mapping P' used to expand the elements of D.
3. The mapping M' that maps the input space S into a continuous Cartesian space GS'.

4-5. rx and ry resolutions in specified units, in which the elements of D will be represented in a discrete Cartesian space GB.

P' is applied to each element d_i of D to produce a set T_i:

$$T_i = \{P'(d_i)\} \mid d_i \in D \text{ where } T_i \subset S \tag{3-1}$$

Let U_i be the set created by applying M' to T_i:

$$U_i = \{M'(t)\} \forall t \in T_i \tag{3-2}$$

Let GS be the Minimum Bounding Box (MBB) of the union of the expanded and mapped elements of D:

$$GS = MBB(U_i \cup ... \cup U_n) \tag{3-3}$$

An empty or "cleared" GB is then created as a discrete approximation of the continuous space GS at resolutions rx and ry.

2D Example in Cartesian Coordinate Space

A collection of unique elements is first mapped into an appropriate coordinate system. 2D coordinate systems include spherical (r, θ) and Cartesian (x, y), such as the geographic coordinate system (latitude, longitude). 3D coordinate systems include spherical (r, θ, φ) and Cartesian (x, y, z), of which color space (e.g., red, green, blue) is a nominal example. An example of a 4D coordinate system would be temporal-geospatial (x, y, z, t), in which t represents time. The following example illustrates how the GB clustering algorithm would operate on a set of points defined in 2D Cartesian space (Figure 3-2).

A GB is defined such that each bit contains at most one point (Figure 3-3). Each point is represented as a "set bit" (bit value=1) with an index that identifies its unique position (Figure 3-3). All other bits in the GB are cleared (bit value=0). To minimize memory requirements, the GB (Figure 3-5) is limited to the minimum-bounding rectangle (MBR) that will contain the union of sets U_i (Eq. 3-3).

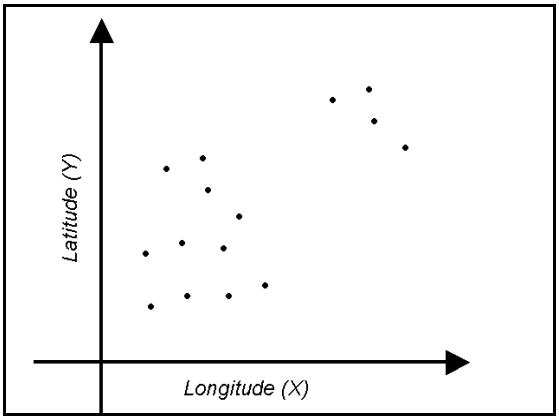


Figure 3-2. Collection of points in 2D geographic space.

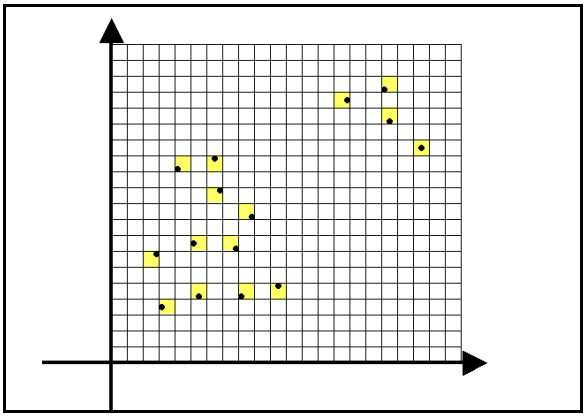


Figure 3-3. Points mapped to GB: each bit holding a point is set; other bits are cleared.

An expansion shape is defined to determine how the clusters will form around the points. Any shape with dimension less than or equal to the dimension of the element space can define a cluster. For example, a set of points in 2D space can be clustered using a line segment (1D expansion) or using rectangles, ellipses, or triangles (2D expansion). Points in 3D space can be clustered into any 1D, 2D, or 3D (e.g., boxes, spheroids, or cones) shape. Using a circle (2D) or sphere (3D) as an expansion shape provides the benefit of maintaining an equidistant expansion around each set bit. The expansion shape can be chosen to help preserve or accentuate a perceived pattern in the data. For example, a long, narrow ellipse might be an appropriate expansion shape for clustering a set of points that lie roughly along a line. The *size* of the expansion shape will have an effect on the resultant cluster resolution (i.e., how close two points must be to be clustered together). The example in this paper uses a 5-bit x 5-bit square for the expansion shape (Figure 3-4).

The GB must be enlarged just enough to accommodate the expansion shape in any direction. This new GB (called GB1) is determined by the MBR of the expansion of all points to be

clustered. The GB is enlarged by two bits along each edge, as shown in Figure 3-5. Each point is then "expanded" by setting the surrounding bits, according to the expansion shape.

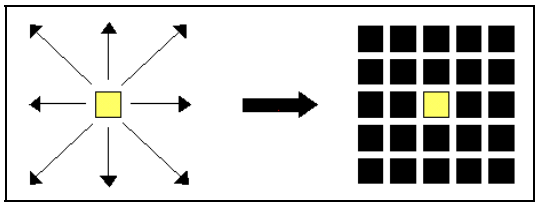


Figure 3-4. Example of expansion shape that is formed around each element in the data set.

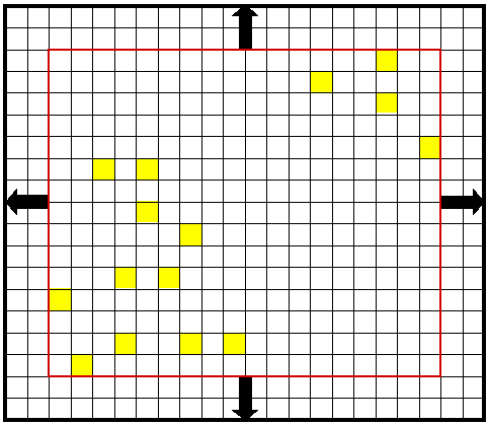


Figure 3-5. Size of GB1 is determined by the MBR of the expansion of all points to be clustered.

Points that are geographically close to one another will cluster together (Figure 3-6). Here, it is clear that the size of the expansion shape directly affects the size of the resultant clusters: a larger shape will result in larger clusters with a greater maximum point spacing, while smaller shapes will result in smaller, tighter clusters. In other words, the size of the expansion shape determines the resolution of the resulting clusters.

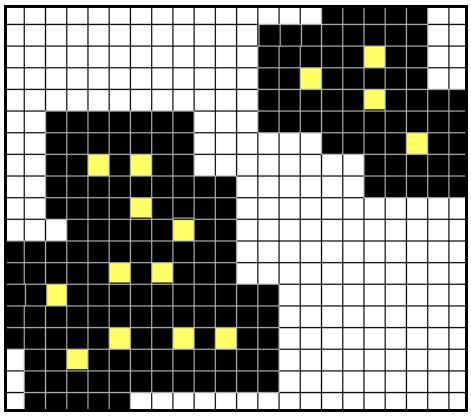


Figure 3-6. Points that are geographically close to each other will cluster together.

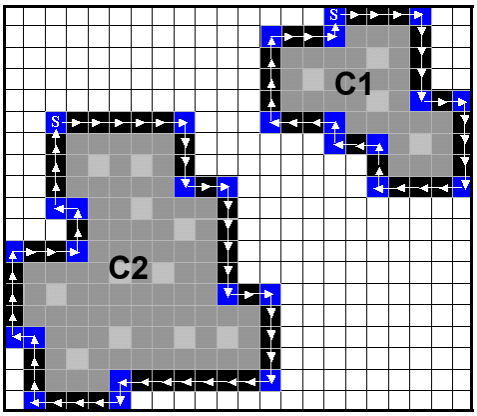


Figure 3-7. Starting at upper-left corner bit, traverse each cluster (in a consistent direction) to obtain vertices (in blue).

The next step in the GB algorithm is to reduce each cluster to a set (or graph) of 1-bit vertices that defines the cluster's bounding polygon. A copy of the bitmap is made, and all internal bits of each cluster in the copy are cleared (e.g., the gray bits in Figure 3-7). Traversal of the cluster starts at a convex vertex, for which the interior angle is less than 180 degrees, and for which there is knowledge of the vertex's relationship to the polygon. For example, if the traversal starts at an upper-left vertex of the cluster, the edge leaving that vertex in a clockwise direction is to the right (Figure 3-7). From this vertex, traversal continues clockwise around the cluster: at each bit along the boundary, the algorithm tests whether the next set bit is in one of three directions from the current bit, in order: 1) 90° counter-clockwise from the current bit, or 2) in the same direction as the previous iteration, or 3) 90° clockwise from the current bit. An allowable direction is based on whether the new bit is set, and whether that bit is not part of a different polygon. For example, in Figure 3-7, clockwise traversal of clusters C1 and C2 (one at a time) would progress as follows:

1. Start at the upper-left-most bit in the cluster, which is the first vertex in the traversal, and move one bit to the right (clockwise direction). We know this next bit to the right is set, since we started with the upper-left-most bit.
2. Test whether the bit immediately above (90° counterclockwise) is set: no.
3. Test whether the bit immediately to the right (continuing in the same direction as before) is set: yes. This becomes the current bit.
4. Repeat steps 2-3, moving one bit at a time along the top edge of the cluster, until the next vertex is reached. At that point, the tests in steps 2 and 3 will fail, and the algorithm will try the third directional test: whether the bit immediately below (90° clockwise) is set, which it is. When the direction of traversal changes, as in this case, the bit is tagged as a vertex (colored blue in Figure 3-7) and the process continues.
5. Traversal is complete when it returns to the original upper-left-most vertex in the cluster.

In some cases, a vertex can belong to more than one cluster. In Figure 3-8a, C1 and C2 are stored as two separate clusters, since the algorithm does not check for set bits that are diagonally adjacent from each other. In Figure 3-8b, clusters C1 and C2 share one vertex. In Figure 3-8c, the clusters overlap by more than one bit, and are considered one cluster. The algorithm determines whether there are two adjacent clusters or one continuous cluster by examining the 8-bit neighborhood around the bit in question. This test only occurs if more than one potential direction could be taken from the current bit (i.e., if the bits that are 90° counter-clockwise *and* straight ahead from the current bit are both set, as in Figures 3-8b and 3-8c). The algorithm examines the 8-bit neighborhood around the current bit: if there are two cleared bits diagonally opposite each other (and flanking the current bit), then the last possible direction is taken (e.g., 90° clockwise in Figure 3-8b). If there are *not* two cleared bits diagonally opposite each other (and flanking the current bit), then the first possible direction is taken (e.g., 90° counter-clockwise in Figure 3-8c). In both cases, traversal continues until it reaches the original upper-left-most vertex in the cluster, at which point the cluster's bitmap is reduced to a vector boundary, and a graph of all vertices defining the cluster's bounding polygon is created (Figure 3-9).

To determine the number and locations of all points inside each cluster, a logical AND is performed on each cluster GB with GB1 (the cluster that contained the original points). For

example, GB2 = GB AND GB1. The resulting GB2 is then traversed to obtain the number of set bits and their unique locations.

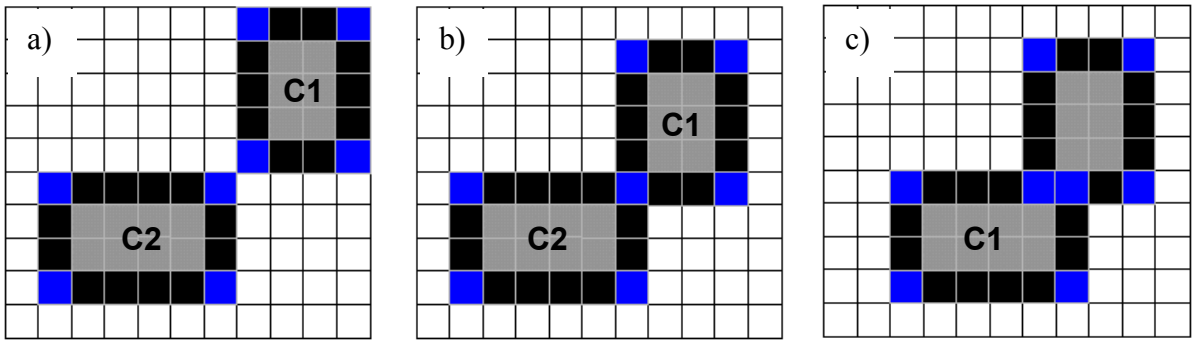


Figure 3-8. Testing whether there are two adjacent clusters or a single cluster: a) two clusters with 4 vertices each; b) two clusters with 4 vertices each (one vertex is shared between the two clusters); c) one cluster with 8 vertices.

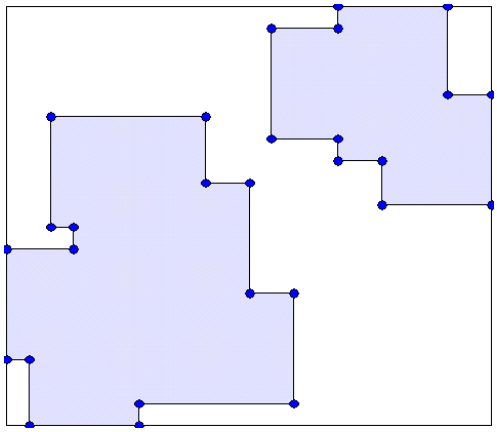


Figure 3-9. Convert bitmaps to vector boundaries.

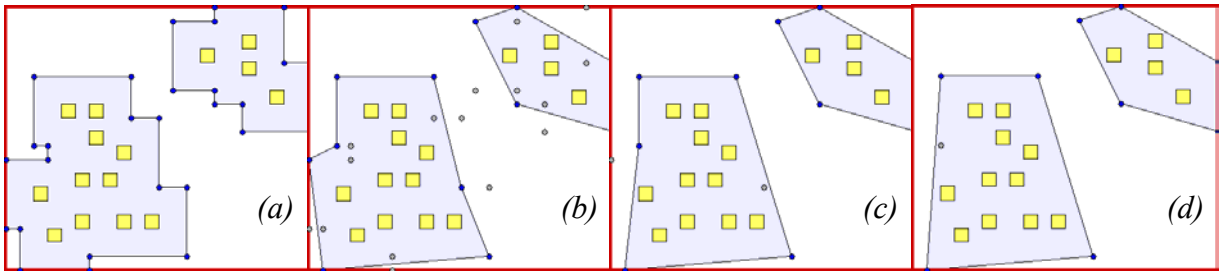


Figure 3-10. Smooth and simplify cluster boundaries by dropping vertices if the resulting polygon still contains all the original points and has an area equal to or less than the unsmoothed boundary. In this example, a recursive polyline-smoothing algorithm drops points and achieves maximum simplification of the clusters after four iterations (a through d).

If desired, the bounding region, now defined by its vertex graph, can be smoothed using the Points of Significance (POS) polyline-smoothing algorithm (Layne et al., 2004), as shown in Figure 3-10. In addition to reducing the number of points needed to define each cluster, these smoothed regions can be used to calculate various cluster metrics, such as density (i.e., number of clustered elements/area of smoothed cluster region). Thus, care must be taken to ensure smoothing does not adversely change the regions; e.g.: 1) the smoothed region should include all originally clustered elements, and 2) the area of the cluster should be maintained or minimized.

Experimental Runs

The clustering algorithm was tested on several data sets to determine which (if any) of the algorithm parameters (listed in Table 3-1) have a significant impact on processing time (**t**).

Table 3-1. GB clustering variables tested for their impact on processing time.

Variable name	Values tested	Description
n = Number of elements	10, 50, 100, 500, 1000, 5000, 10000	Total number of elements to be clustered.
s = Shape of expansion	Circle, Square	Shape of polygon used to determine how clusters would form.
a = Area of expansion	16, 36, 64, 100, 400, 10000 bit^2	Area of expansion shape.
r = Resolution of data set	1, 2, 5 bits/element	On average, how densely spaced elements are within data set.
d = Distribution of elements	Uniform, Distinct	*Uniform* – elements evenly distributed with no obvious clusters. *Distinct* – elements are grouped in obvious clusters.
o = Order of elements	Alternating, Sequential, Random	Order in which elements are input to the algorithm (only varied for Distinct distributions): *Alternating* – 1 element input from cluster 1, then 1 element from cluster 2, etc. (until 1 element is input from every cluster), then another element from cluster 1, another from cluster 2, etc., until all elements are input. *Sequential* – all elements input from cluster 1, then all from cluster 2, etc., until all elements input. *Random* – all elements randomly input.

Methods

All tests of the GB clustering algorithm were performed on a Pentium III 800 MHz PC with 256 MB of RAM running Linux. Six independent variables (table 3-1) were investigated for their effects on the algorithm processing time. All statistical analyses and plots were produced with JMPIN version 4.0.4 software (SAS Institute, 2003).

Results

One-way plots of time as a function of each variable suggest that processing time (**t**) is affected by the number of elements to be clustered (**n**, Figure 3-11), the area of the expansion (**a**, Figure

3-12), and the resolution of the data set (**r**, Figure 3-13). The plots also suggest that **t** is not significantly affected by the shape of the expansion (**s**, Figure 3-14), the distribution of elements (**d**, Figure 3-15), or the order of elements input to the clustering algorithm (**o**, Figure 3-16).

A standard least squares fit (Spiegel, 1975) was performed using all independent variables and their cross products (i.e., a full factorial model) to predict processing time. Since *order* was varied only for distinct distributions, the uniform distributions were said to have *order = uniform* (effectively combining the *order* and *distribution* variables into a single *order* variable, **o**). The final model resulted in a significant F Ratio (F=39.1, p<0.0001), indicating that at least one of the independent variables had an effect on processing time. Individual effect tests for this model indicate that #elements *(***n***)*, expansion area (**a**), and data set resolution (**r**) significantly impacted processing time, while expansion shape (**s**) and element distribution/order (**o**) did not. Interactions among the three significant elements also had an impact (i.e., **nxa**, **nxr**, **axr**, **nxaxr**). Interestingly, interactions among **axs** and **nxaxs** significantly impacted processing time, but the variable **s** on its own did not.

The final model – using all significant effects and interactions – is presented in Figure 3-17. This model has a reasonable R^2 value (0.79) and high F ratio (F=176.4, p<0.0001), and the lack of fit is not significant (p>0.1), suggesting that this model would make a reasonable prediction of processing time from the input variables:

$$\mathbf{t} \cong -0.117112 + 0.0010503 \times \mathbf{n} + 0.000805 \times \mathbf{a} - 0.989703 \times \mathbf{r} + 2.9941\times10^{-7} \times \Delta\mathbf{n} \times \Delta\mathbf{a} - 0.000334 \times \Delta\mathbf{n} \times \Delta\mathbf{r} - 0.000252 \times \Delta\mathbf{a} \times \Delta\mathbf{r} - 9.182 \times10^{-8} \times \Delta\mathbf{n} \times \Delta\mathbf{a} \times \Delta\mathbf{r} - 0.000112 \times \Delta\mathbf{a} \times \mathbf{s} - 4.84 \times10^{-8} \times \Delta\mathbf{n} \times \Delta\mathbf{a} \times \mathbf{s}$$

Where: $\Delta\mathbf{n} = \mathbf{n} - 2539.00$,
$\Delta\mathbf{a} = \mathbf{a} - 2712.34$,
$\Delta\mathbf{r} = \mathbf{r} - 1.50$

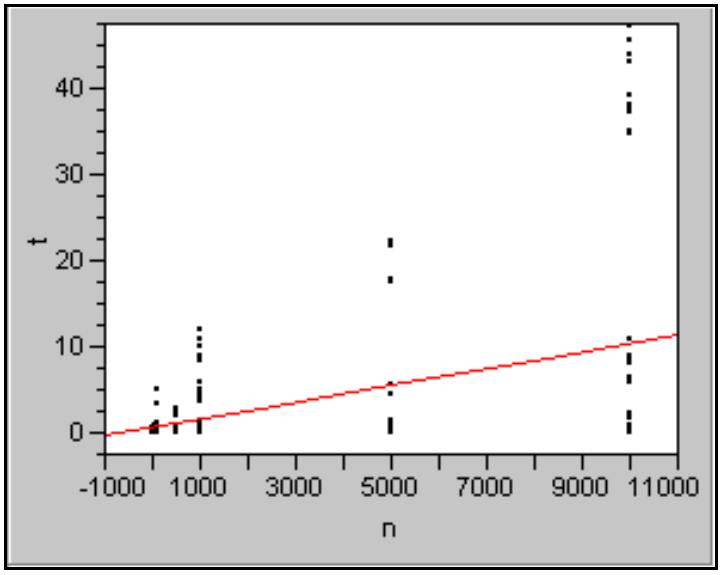


*Figure 3-11. Bivariate fit of time (**t**) by #elements to be clustered (**n**).*

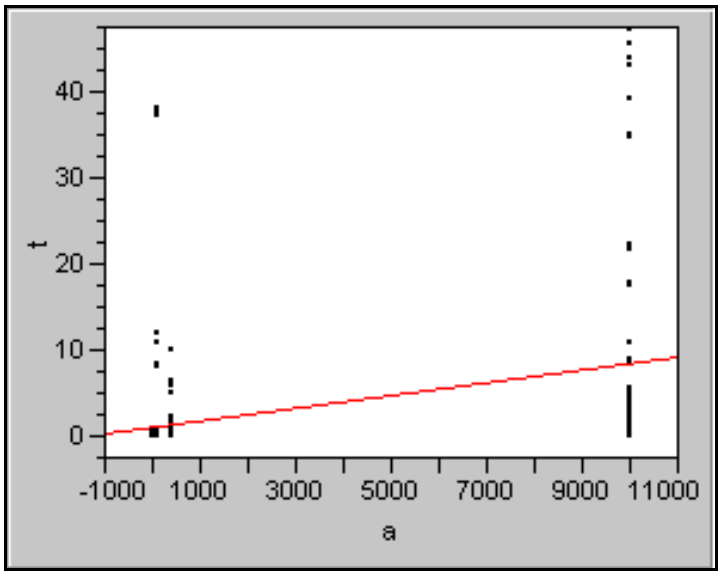


*Figure 3-12. Bivariate fit of time (**t**) by expansion area (**a**).*

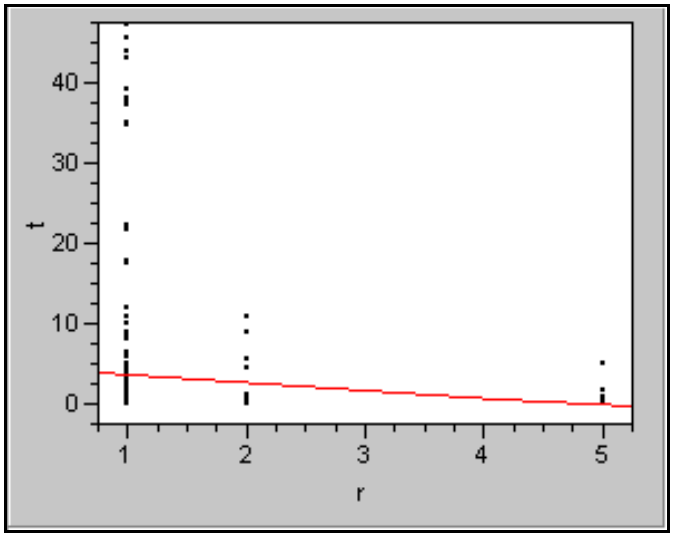


*Figure 3-13. Bivariate fit of time (**t**) by data set resolution (**r**) in bits/element.*

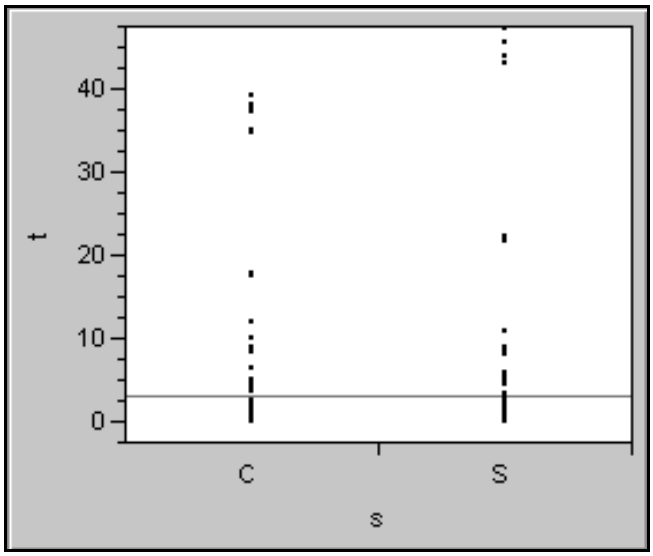


*Figure 3-14. One-way analysis of time (**t**) by expansion shape (**s**): C = circle, S = square.*

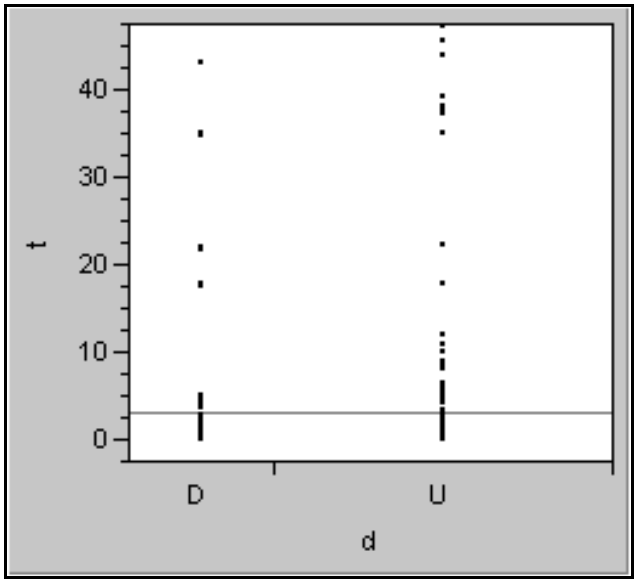


*Figure3- 15. One-way analysis of time (**t**) by element distribution (**d**): D=distinct, U=uniform.*

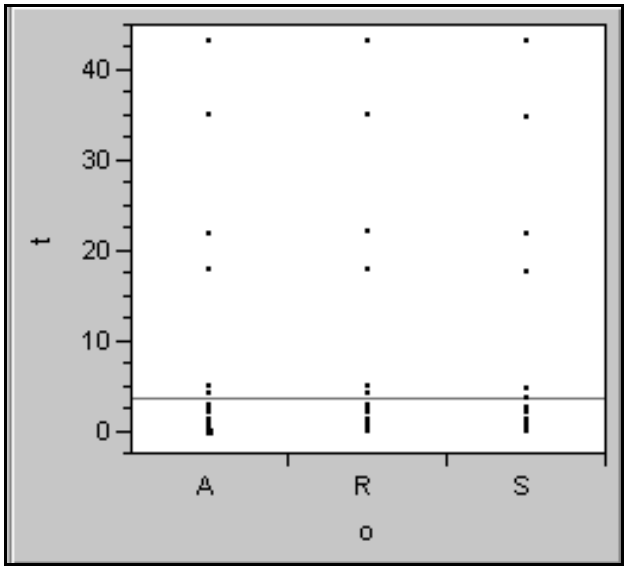
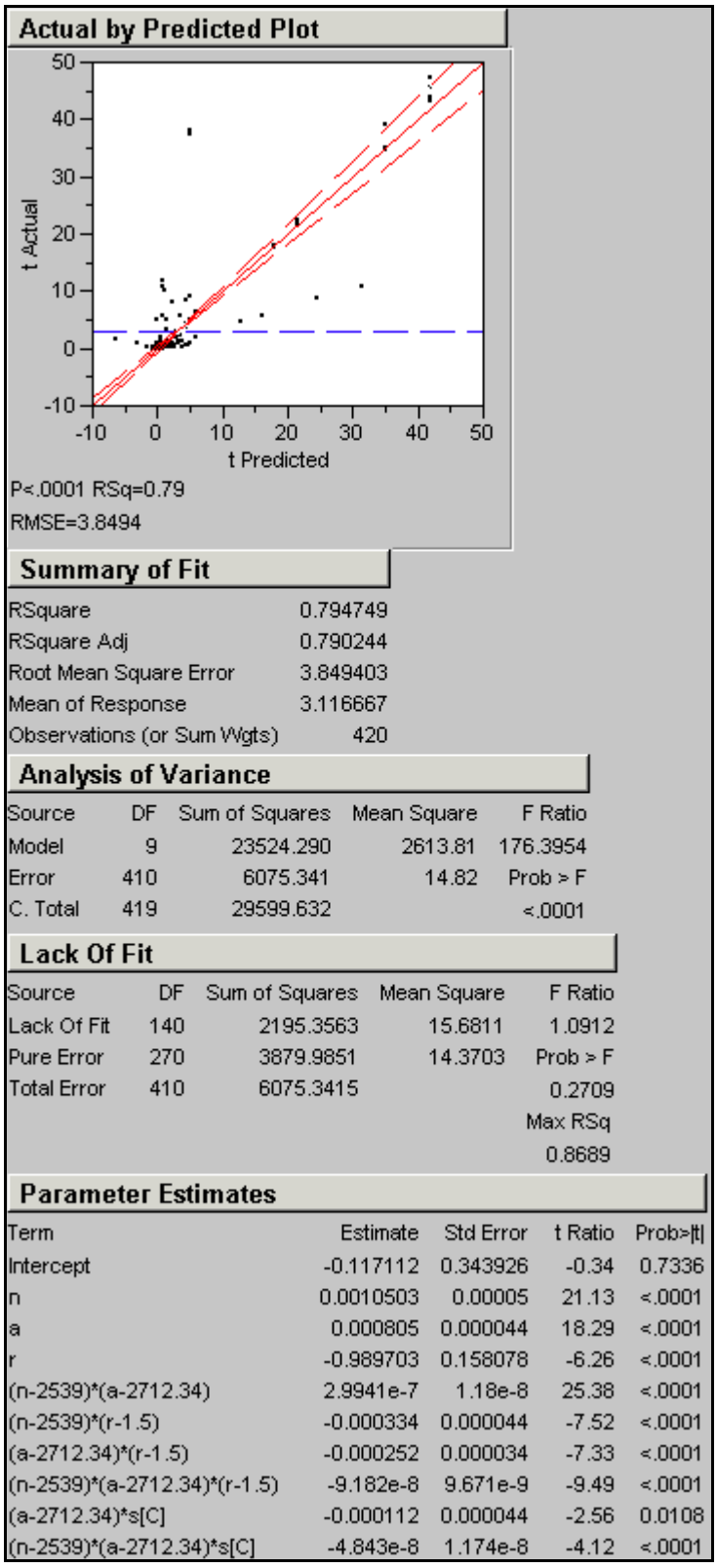


*Figure 3-16. One-way analysis of time (**t**) by ordering (**o**) of elements to algorithm: A=alternating, R=random, S=sequential.*



Summary of Fit

RSquare	0.794749
RSquare Adj	0.790244
Root Mean Square Error	3.849403
Mean of Response	3.116667
Observations (or Sum Wgts)	420

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	9	23524.290	2613.81	176.3954
Error	410	6075.341	14.82	Prob > F
C. Total	419	29599.632		<.0001

Lack Of Fit

Source	DF	Sum of Squares	Mean Square	F Ratio
Lack Of Fit	140	2195.3563	15.6811	1.0912
Pure Error	270	3879.9851	14.3703	Prob > F
Total Error	410	6075.3415		0.2709
				Max RSq
				0.8689

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	-0.117112	0.343926	-0.34	0.7336
n	0.0010503	0.00005	21.13	<.0001
a	0.000805	0.000044	18.29	<.0001
r	-0.989703	0.158078	-6.26	<.0001
(n-2539)*(a-2712.34)	2.9941e-7	1.18e-8	25.38	<.0001
(n-2539)*(r-1.5)	-0.000334	0.000044	-7.52	<.0001
(a-2712.34)*(r-1.5)	-0.000252	0.000034	-7.33	<.0001
(n-2539)*(a-2712.34)*(r-1.5)	-9.182e-8	9.671e-9	-9.49	<.0001
(a-2712.34)*s[C]	-0.000112	0.000044	-2.56	0.0108
(n-2539)*(a-2712.34)*s[C]	-4.843e-8	1.174e-8	-4.12	<.0001

Figure 3-17. Model of processing time (**t**) *as a function of #elements* (**n**)*, expansion area* (**a**)*,* data set *resolution* (**r**)*, expansion* shape (**s**)*, and significant interactions among these variables. Note that* shape (**s**) *on its own does not have a significant effect on* **t**.

Clustering Demonstration

The clustering results obtained with the author's GB algorithm are repeatable, order-independent, and produce bounded regions for which the density of the clustered objects can be computed. These characteristics make the GB clustering algorithm a good choice for automated applications such as clustering mine-like objects in SSI for MIW operations. Analysts at NAVOCEANO who previously clustered these objects manually to obtain clutter density now use the author's GB clustering algorithm exclusively in support of MIW operations. Figure 3-18 presents a sample MIW data set that was manually clustered; Figure 3-19 presents the same dataset clustered using the GB algorithm with a circular expansion shape and clustering radius of 250m. A comparison of these figures illustrates that the GB clustering algorithm closely approximated the manual clusters.

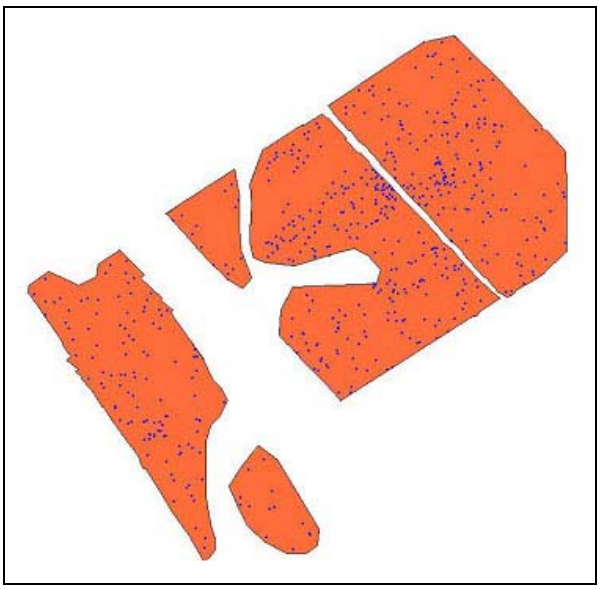


Figure 3-18. Manual Clustering

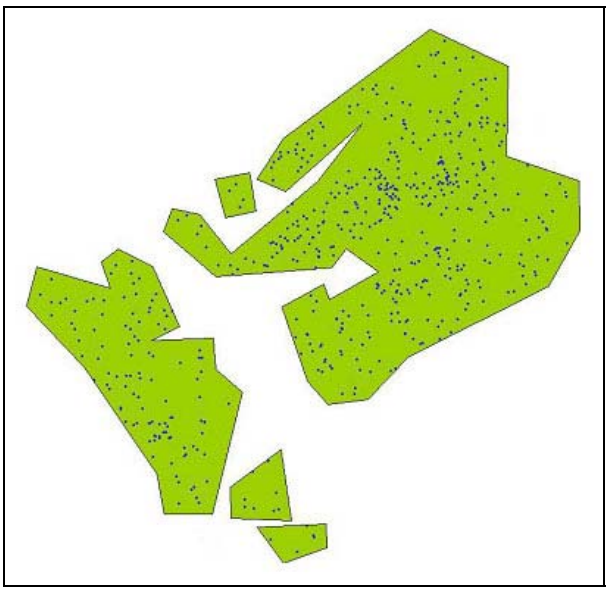


Figure 3-19. Automated Clustering

Conclusion

The author's GB clustering algorithm provides a repeatable method to cluster points into bounded regions where metrics such as density for the regions can be computed directly. Because the clustering is repeatable, MIW analysts at NAVOCEANO are able to use the algorithm to automatically cluster mine-like objects geographically and to determine the density of mines in areas of interest. The clustering algorithm was tested on several data sets to determine which (if any) of the algorithm parameters have a significant impact on processing time. Results from tests performed in this chapter show that the time to cluster a set of points is not affected by the distribution or the order of the points. The number of points to be clustered and the area and the resolution of the expansion shape affected the processing time of the algorithm. Future research by the author will involve comparing the clustering algorithm with several other clustering algorithms such as k-Means and Kohenon self-organizing maps.

Acknowledgements

This work was performed in support of the Naval Oceanographic Office (NAVOCEANO) and funded by the Space and Naval Warfare Systems Command (SPAWAR) through the Commander, Naval Meteorology and Oceanography Command (CNMOC) and ONR under Program Element 0603207N**.** The authors thank Captain Robert Clark (program manager, SPAWAR PMW 150), Dr. Edward Mozley (assistant program manager for data acquisition, SPAWAR PMW 150), Mr. Bruce Northridge (CNMOC) and Mr. Michael M. Harris (NRL) for their support of this project. We thank Mr. Jim Hammack (NAVOCEANO) for his recommendations concerning the application of this algorithm in support of NAVOCEANO requirements. Any mention of commercial products or the use of company names does not imply endorsement by the U.S. Navy. This paper is approved for public release; distribution is unlimited.

References

Barnard, J.M. (1996). Agglomerative Hierarchical Clustering Package. Barnard Chemical Information. Presented at *Daylight EUROMUG Meeting*, Basel, Switzerland, December 17.

Can, F. and E.A. Ozkarahan (Dec. 1990). Concepts and Effectiveness of the Cover-coefficient-based Clustering Methodology for Text Databases. *ACM Transactions on Database Systems* 15:4.

Cormen, T.H., C.E. Leiserson, and R.L. Rivest (2001). *Introduction to Algorithms, Second Edition.* Massachusetts: MIT Press and McGraw-Hill.

Day, W.H.E. and H. Edelsbrunner (1984). Efficient Algorithms for Agglomerative Hierarchical Clustering Methods. *Journal of Classification*, 1:1.

Downs, G. (2001). Clustering in Chemistry. Presented at *MathFIT workshop*, Belfast, April 27.

Downs, G.M. and J.M. Barnard (2003). Clustering Methods and Their Uses in Computational Chemistry. *Reviews in Computational Chemistry*, 18:1. New York: John Wiley and Sons.

Gendron, M.L., P.B. Wischow, M.E. Trenchard, M.C. Lohrenz, L.M. Riedlinger and M.J. Mehaffey (2001). *Moving Map Composer*. Naval Research Laboratory, Stennis Space Center, MS. U.S. Patent Number 6,218,965, April.

Gonzales, R.C. and R.E. Woods (2002). *Digital Image Processing, Second Edition.* New Jersey: Prentice Hall.

Halkidi, M., Y. Batistakis and M. Vazirgiannis (2002). Cluster Validity Methods: Part II. *Special Interest Group on Data Management Record*.

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

Hartigan, J.A. and M.A. Wong (1979). A K-means Clustering Algorithm. *Applied Statistics*, 28.

Ho, T.K. and G. Nagy (2000). OCR With No Shape Training. In Proceedings of the *15th International Conference on Pattern Recognition,* Barcelona, Spain, September. 3-8.

Hobby, J. and T.K. Ho (1997). Enhancing Degraded Document Images Via Bitmap Clustering and Averaging. In Proceedings of the *4th International Conference on Document Analysis and Recognition,* Ulm, Germany, August 18-20.

Höppner, F., F. Klawonn, R. Kruse and T. Runkler (1999). *Fuzzy Cluster Analysis*. Chicester: John Wiley and Sons.

Jain, A.K., M.N. Murty, and P.J. Flynn (1999). Data Clustering: A Review. *ACM Computing Surveys,* 31:3.

Layne, G., M. Gendron and M. Lohrenz (2004). POS Polyline Smoothing: Reduction of Polyline Vertices. In Proceedings of the *Tenth International Conference on Industry, Engineering and Management Systems*, Cocoa Beach, Florida, March.

Postaire, J.G., R.D. Zhang, C. Lecocq-Botte (1993). Cluster Analysis by Binary Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:2.

Russ, J. C. (2002). *The Image Processing Handbook, Fourth Ed.* New York: CRC Press.

SAS Institute, Inc. (2003). *JMPIN V.4.0.4 statistical analysis software package*. Cary, North Carolina.

Sibson, R. (1973). SLINK: An Optimally Efficient Algorithm for the Single-Link Cluster Method. *Comput. J*. 16:1.

Spiegel, M.R. (1975). *Schaum's Outline of Theory and Problems of Probability and Statistics*. Schaum's outline series. New York: McGraw-Hill.

Voorhees, E.M. (1985a). *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* Ph.D. Thesis, Cornell University, New York.

Voorhees, E.M. (1985b). The Cluster Hypothesis Revisited. In Proceedings of the *8th Annual International ACM Special Intertest Goup on Information Retrevial Conference on Research and Development in Information Retrieval*.

Yoon, J.P., V. Raghavan, and V. Chakilam (2001). BitCube: A Three-dimensional Bitmap Indexing for XML Documents. *Journal of Intelligent Information Systems*, 17.

Chapter 4 - A Real-time Computer-Aided Detection (CAD) Algorithm

Marlin L. Gendron, Geary J. Layne, Maura C. Lohrenz
Naval Research Laboratory (NRL), Code 7440.1, Stennis Space Center, MS

Juliette Ioup
University of New Orleans (UNO), New Orleans, LA

Abstract

The author presents his real-time computer-aided detection (CAD) algorithm to automatically detect mine-like objects on the seafloor, which show up as bright spots and adjacent shadows in sidescan sonar imagery (SSI). The CAD algorithm uses Geospatial Bitmaps (GB) to detect shadows and bright spots in real-time. The algorithm calculates "shadow" and "bright" intensity thresholds one scan-line at a time (i.e., as the imagery is being collected). The author ran his GB-based CAD algorithm on real SSI data, adjusting input parameters until the algorithm successfully determined the geographical location of at least 90% of 48 known inert mines. The results of these tests and comparisons with other CAD algorithms indicate that the author's real-time CAD algorithm performs comparably to or better than other non-real-time CAD algorithms. A prototype version of the real-time CAD algorithm has been transitioned to the Mine Warfare community to assist sidescan sonar analysts performing change detection. The algorithm automatically detects mine-like echoes (MILECs) in SSI that otherwise might be missed, identifies clutter that might be obscuring MILECs, and eliminates large amounts of benign imagery (e.g., with minimum clutter or other features) that would be of no interest to the analyst.

Acronyms

AUV	autonomous underwater vehicle
CAD	computer-aided detection
CNMOC	Commander, Naval Meteorology and Oceanography Command
GB	geospatial bitmap
GPS	Global Positioning System
LAT/LON	latitude and longitude
LCM	Layne Completion Method
MILEC	mine-like echo
MIW	Mine Warfare
MRF	Markov Random Field
MU	mini-UNISIPS
NAVOCEANO	Naval Oceanographic Office
NRL	Naval Research Laboratory
NSWC-PC	Naval Surface Warfare Center – Panama City

P_c	probability of correct detection
P_{fa}	probability of false alarms
SPAWAR	Space and Naval Warfare Systems Command
SSI	sidescan sonar imagery
SSS	sidescan sonar system
TVG	time-varying gain
UNISIPS	Unified Sonar Image Processing System

Introduction

Modern remote sensing devices are capable of collecting tremendous amounts of high-resolution digital imagery in remote locations. This imagery must be evaluated by an analyst to obtain useful information (e.g., detect and identify specific features of interest). If the imagery is time-critical, analysis must be performed as quickly as possible. However, the sheer volume of data available to analysts often precludes timely analysis (Reed, 2001; Cronin et al., 2003).

The design and development of computer-aided detection (CAD) algorithms is advancing rapidly in many fields, such as medicine. In 1967, Winsberg, et al. recognized the difficulty of viewing and analyzing large amounts of screening mammograms and suggested computer algorithms could help. There were also early attempts to identify lesions such as malignant tumors by using CAD algorithms (Ackerman and Gose, 1972). Giger (2001) suggested CAD algorithms should be considered for incorporation into all new medical digital imaging systems.

During military Mine Warfare (MIW) operations, analysts currently perform change detection to aid in the clearing of assault lanes. Change detection is done by visually comparing historical high resolution imagery collected by a sidescan sonar system (SSS) with recently collected sidescan sonar imagery (SSI) in an attempt to identify mines placed at sea since the last time the area was surveyed. Manual change detection using SSI can be a significant time saving tool, but problems such as large data volumes (Reed, 2001; Cronin et al., 2003), navigational errors, and sonar tow-body instabilities can hinder these time savings (Lingsch and Lingsch, 2001).

Computer algorithms can be designed to aid the SSI analyst perform change detection by 1) automatically detecting mine-like echoes (MILECs) in the imagery that otherwise might be missed, 2) identifying clutter that might be obscuring MILECs, and 3) eliminating large amounts of benign imagery (e.g., with minimum clutter or other features) that would be of no interest to the analyst. This paper presents a new CAD algorithm developed by the author to operate on imagery as it is being collected (i.e., in real-time) by SSS towed by a ship or mounted on an autonomous underwater vehicle (AUV).

SSS, deployed during underwater survey of an area of interest, use an acoustic transducer to send acoustical waves – or beams – through the water and receive them back to image the seafloor. The beams are sent in a wide angular pattern down to the bottom in swaths 100-500 meters wide, and the echoes are received back creating a narrow strip below and to the sides of the transducer track (Blondel and Murton, 1997). When the raw SSS data is processed into SSI, image analysts can extract details about the local morphology of the seafloor.

The beams spread or fan out perpendicular to the direction of travel, called across-track. Across-track resolution is determined by the bandwidth. The bandwidth for the Klein 5000 is 20kHz (Klein Associates, Incorporated, 2004). The amount of across-track spreading is measured by the beam angle (Huff et al., 1991). The top of Figure 1-1 shows a typical beam pattern for a SSS. The main lobes point to the side making the sonar effectively "blind" directly below the sonar, called nadir.

The SSS can be hull-mounted, towed from a platform such as a ship or helicopter, or carried on-board an AUV. The SSS is usually equipped with pressure or altimeter sensors that allow it to follow the bottom, maintaining a constant height above the sea floor, or "fly" at a constant depth below the surface (Figure 4-1).

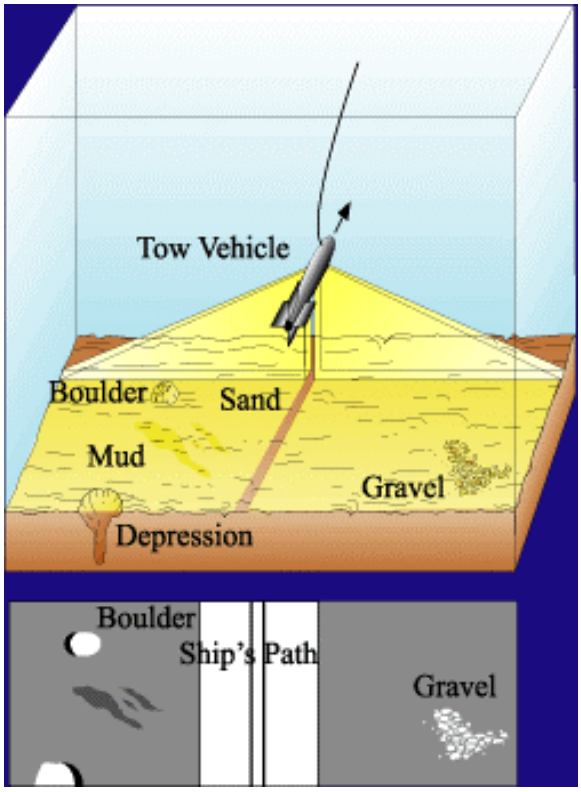


Figure 4-1. Illustration of a towed sidescan sonars system with resulting sidescan sonar imagery .

There are many different methods that aid in the extraction of MILECs from SSI, including wavelet-denoising (Ioup et al., 2001) and neural networks (LeBlanc and Manolakos, 1991), but the software implementation of these methods often is not practical for real-time CAD. One extraction method commonly used is a technique that utilizes the Markov Random Field (MRF) model to segment the entire gray-scale SSI into bright regions ("brights"), dark regions ("shadows"), and reverberation (Murino, 2001; Reed et al., 2001; Reed et al., 2003). A bright corresponds to reflected acoustical energy, a shadow represents a lack of reflected energy, and what remains is so-called bottom reverberation (Mignotte et al., 2000b). The CAD algorithm developed by the author and presented in this paper uses a similar technique and produces similar results, but it operates on one scan line at a time, rather than on the entire image at once.

This allows the algorithm to detect in real-time, but introduces other challenges addressed later in the paper.

Researchers have shown that it is not reasonable for a single algorithm to address every difficulty associated with mine-like detection. It is better to employ several reliable CAD algorithms in combination to perform the detection (Dobeck and Cobb, 2004; Neretti, et al., *submitted)*. This concept of combining several different CAD algorithms (each with its own strengths and weaknesses) into one detection bank was proposed Dobeck (2004). In this configuration, each CAD algorithm performs detection individually on the same imagery and computes a confidence measure for each detected MILEC. Dobeck's method, called Algorithm Fusion (Aridgides et al., 2004; Ciany et al., 2004; Dobeck and Cobb, 2004), combines the confidence terms from each algorithm for a given MILEC into a single weighted average. This average, along with a snippet of the suspected MILEC, is presented to the analyst for further classification and false detection rejection. The author will compare this Algorithm Fusion method with the author's real-time CAD in the "Testing the CAD Algorithm" section of this paper.

Sidescan Sonar Imagery (SSI)

To produce SSI, each scan line is sampled between 0 and 255 to produce intensity (grayscale) values. The imagery is built up one scan line of data at a time (Figure 4-2).

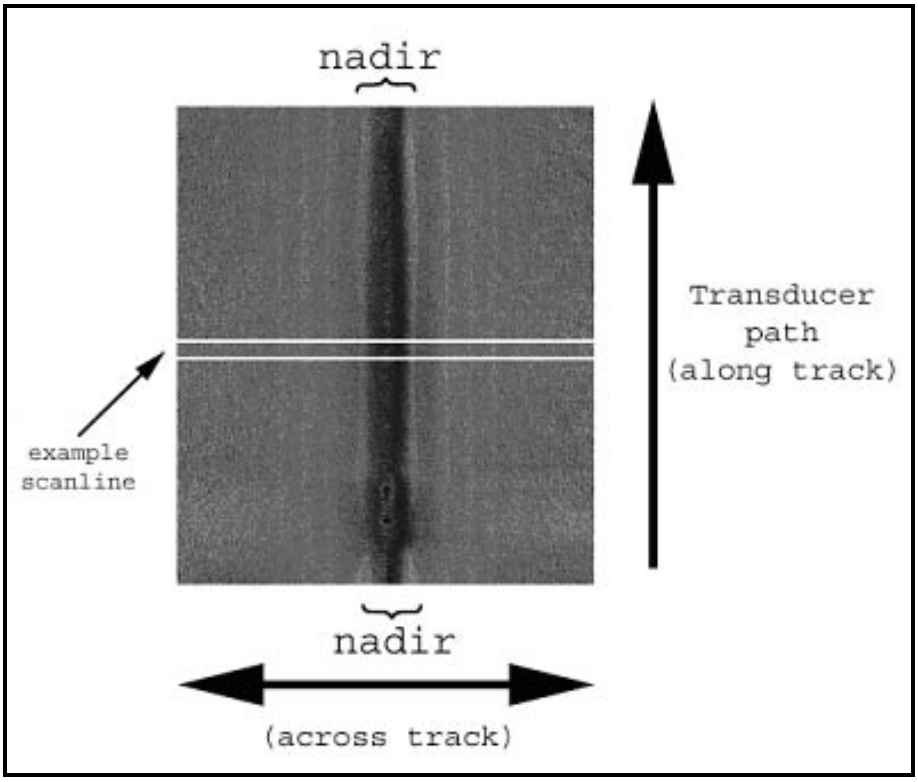


Figure 4-2. Sidescan sonar imagery

The Klein 5000 SSS, develop by Klein Associates Incorporated, collected the sonar data used in this paper. The center frequency is 455 kHz with a pulse length of 50 to 200 μ sec that gives a

20 cm along-track resolution. The across track resolution is 36 cm at a maximum of 150 m for an array 120cm long (Klein Associates, Incorporated, 2004).

The Klein 5000 is usually towed at 10 knots with a 300 m swath at an altitude of 25 m. The sonar tow fish is equipped with standard nominal heading, pressure, pitch, altimeter and roll sensors. The towfish is compact and relatively light and easy to use. The body length is 194 cm and the diameter 15.2 cm. It only weighs 70 kg in air (Klein Associates, Incorporated, 2004).

The data were processed into imagery by the Naval Oceanographic Office (NAVOCEANO) and stored in the Unified Sonar Image Processing System (UNISIPS) format. Each scan line is stored in the UNISIPS file as a separate record. For any given line, the latitude and longitude coordinates at nadir are known, as well as the heading of the sonar and its depth above the seafloor.

Features such as pockmarks, sand ripples, and MILECs close to or on the seafloor are visible in the SSI. MILECs, in particular, are discernable as brights with adjacent shadows perpendicular to nadir. The dimensions of a MILEC can be estimated as a function of the shadows' dimensions, but only the reflective structures of that portion of the MILEC exposed above the seafloor can be estimated. These estimates are calculated along the MILEC's two axes that are parallel and perpendicular to nadir. For example, the height of the MILEC (above seafloor) can be estimated from the perpendicular dimension of the shadow (relative to nadir), which varies as a function of beam angle (Fish and Carr, 1990). Likewise, the length of the surface of the MILEC facing the SSS can be estimated from the parallel dimension of the shadow. Figure 4-3 shows an example of a MILEC.

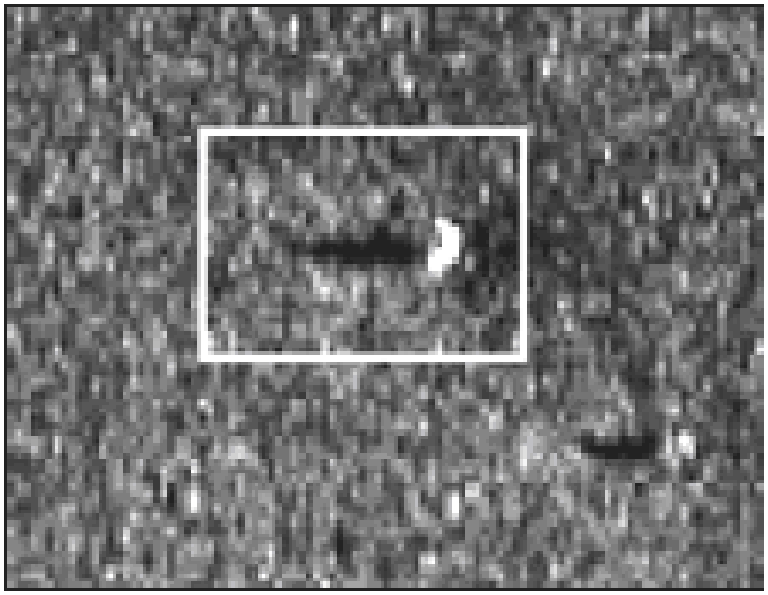


Figure 4-3. Mine-like Echo detected in sidescan imagery

SSI, by its nature, is non-linear (Reed et al. 2002). Correlation among pixels in SSI is affected by "speckle" noise, which reduces resolution (Blondel and Murton, 1997). It is desirable to

remove as much noise as possible without degrading or altogether obliterating the MILECs (Sams and Agerkvist, 2002). SSI may also contain phantom features created by surface return when the sonar is in shallow water or when the towfish is pitched at a few degrees or more. Global Positioning System (GPS) signal dropout can cause the image processing software to lose or duplicate scan lines (Fish and Car, 1990). All of these factors combine to hamper the ability of algorithms to detect features in the imagery. Any automated feature detection method applied to the imagery must be robust and able to cope with these difficulties. The task takes on a new dimension of difficulty when developing a real-time CAD that only processes a single scan line at a time, because each scan line is unlikely to adequately represent (i.e., in a statistically significant manner) the intensity range of the entire image.

Real-Time CAD Algorithm

The real-time CAD algorithm developed by the author ingests one UNISIPS scan line at a time. Across-track bright and shadow positions, lengths, and intensity information are immediately gathered from the scan line and stored in two one-dimensional geospatial bitmaps (GBs): one for shadows and one for brights. A circular lookup table is created to "window" the imagery, several scan lines at a time. This lookup table is the same width as the GBs and is populated with the positions and run-lengths of shadows and brights stored in the GBs. The window is used to make the final detection decision and is described in the "Detecting Objects" section of this paper.

Due to real-time processing considerations, the author's CAD algorithm relies on a GB technique (Gendron et al., 2001) patented by the author et al. Simple bitmaps – with a depth of one bit per pixel – are binary structures in which bits are turned on (set = 1) or off (cleared = 0). The index of each bit is unique and denotes its position relative to the other bits in the bitmap. The author extended this concept to construct GBs in which every bit represents an object at some unique geospatial location. A set bit indicates that some object of interest exists (in this case a bright or shadow pixel) at that location, accurate to within the resolution of the bitmap. A cleared bit indicates the absence of any object at that location. Although a GB can be defined for an entire finite space, memory is only allocated – dynamically – when groups of spatially close bits are set, resulting in a compact data structure that supports very fast Boolean and morphological operations.

Across-Track Processing

First, we start with a modification to the common image processing technique called thresholding to extract shadows and brights from the scan line. Traditionally, thresholding is performed on an entire image, a process often called posterization, where select pixels, representing the foreground, are set to black and all other pixels (the background) are set to white to produce a binary image (Russ, 2002). The threshold is typically chosen in one of two ways: 1) ad hoc based on a histogram, or 2) interactively by a human after the entire image has been equalized (Russ, 2002). Because the author's algorithm is automatic and operates in real-time, neither of these options is practical. Furthermore, we desire two distinct thresholds to capture both brights and shadows; all remaining pixels can be ignored.

We locate shadows and brights in a scan line by first determining a lower intensity bound, $i_{\min}$, in which all samples of intensity lower than $i_{\min}$ are considered shadows. Likewise, an upper intensity bound, $i_{\max}$, is found in which all samples of intensity that fall above $i_{\max}$ are considered brights. These intensity bounds are determined for each scan line, minus the center 10% of pixels, which are located close to nadir where the SSS is blind.

The minimum and maximum intensity values, $m_{\min}$ and $m_{\max}$, are also determined for all pixels in the scan line, minus the center 10%. If $m_{\max} < 128$, the scan line is considered too dark to contain any suitable brights, and thus the entire scan line is ignored. This can greatly reduce processing time. If the scan line is not ignored, the mid-range value, m_{half}, is computed by:

$$m_{half} = \frac{m_{\max} - m_{\min} + 1}{2} \tag{4-1}$$

Ideally, the intensity values for each line (and thus, the entire UNISIPS file) would be normally distributed with a range of 0 to 255. The thresholds, $i_{\min}$ and $i_{\max}$, could then be chosen as the first and third quartiles of the distribution. I.e., $i_{\min} = \frac{256}{4} = 64$ and $i_{\max} = \frac{3(256)}{4} = 192$ would be true for the entire file, but this is not generally the case. The intensity actually varies with range (more rapidly closer to nadir) partly because the time-varying gain (TVG) function is imperfectly matched to the environment and the system (Anstee, 2001). The TVG function is applied during transducer amplification in an attempt to keep the average output power approximately constant. The intensity is also affected by the sonar frequency, substrate slope, micro-scale roughness, and surface composition and density (Kvitek et al., 1999).

The distribution also tends to be skewed toward lower intensities. Traditionally in image processing, a gamma shift is used to stretch skewed distributions, maximizing the ability of the human eye to detect individual differences in the intensities (Foley and van Dam, 1994). In practice, a lookup table is created experimentally, based on actual measured intensity values. The use of this lookup table to approximate a gamma shift (which improves processing speed) is known as a gamma correction (Foley and van Dam, 1994).

An appropriate gamma shift will convert the intensities of the image to fit a normal distribution, such that $i_{\min}$ and $i_{\max}$ could be set to the quartiles of the shifted (normal) distribution. Since the distribution of intensity values is different for each scan line, the gamma correction would also be different for each scan line. However, performing a gamma correction on every scan line to calculate intensity thresholds would be too computationally intensive to support a real time application. Therefore, the author simply map the ideal $i_{\min}$ and $i_{\max}$ values from the standard normal distribution to the actual distribution via an approximation of the inverse gamma function, based on statistical measurements of each scan line's intensities. In other words, only two values ($i_{\min}$ and $i_{\max}$) are mapped for each scan line, rather than 90% of the scan line, as follows:

$$i_{\min} = Am_{half} + \sigma^2 r \tag{4-2}$$

$$i_{\max} = 256 - Bm_{half} + \sigma^2 r^{-1} \tag{4-3}$$

where the value 256 in equation 4-3 is the largest intensity value possible. The constants A and B and, σ^2, and r, are defined below.

The mean, μ, and variance, σ^2, are computed from all pixels in the scan line minus the center 10%. A single pass variance calculation, used by some hand-held calculators, is used to improve processing speed (Iman, 1999):

$$\sigma^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} X_i^2 - \frac{(\sum_{i=1}^{n} X_i)^2}{n} \right), \tag{4-4}$$

r can be thought of as a percentile shift from the mean:

$$r = \frac{m_{half}}{\mu} \tag{4-5}$$

A and B are experimentally determined values through direct testing (discussed in the *Testing the CAD Algorithm* section of this chapter) that allow the CAD algorithm to achieve a probability of correct detection (P_c) of at least 90% on ground truth data sets. As A and B increase, the P_c and the probability of false alarms (P_{fa}) both increase.

$$A = 0.292 \tag{4-6}$$

$$B = 1.480 \tag{4-7}$$

Scan line Shadow/Bright Thresholding

After $i_{\min}$ and $i_{\max}$ have been determined for scan lines with $m_{\max} \geq 128$, the port and starboard halves of the scan lines are processed separately. (Note that the center 10% of the scan line is retained for this process.) Each half of the scan line can be represented by a vector, X, of length N. The following method is used to process shadows and brights for the starboard side; the port side is processed similarly.

Two GBs of size $1 \times N$ are created, one for shadows and one for brights. A different gamma adjustment, γ, based on an error approximation of SSS parameters, is computed for position x within X:

$$\gamma = e^{-\beta x / N} \tag{4-8}$$

where β is based on the sonar parameters, such as TGV. As β approaches infinity, the gamma correction approaches 0 over a greater range of X (Figure 4-4), and therefore has less affect on the intensity thresholds.

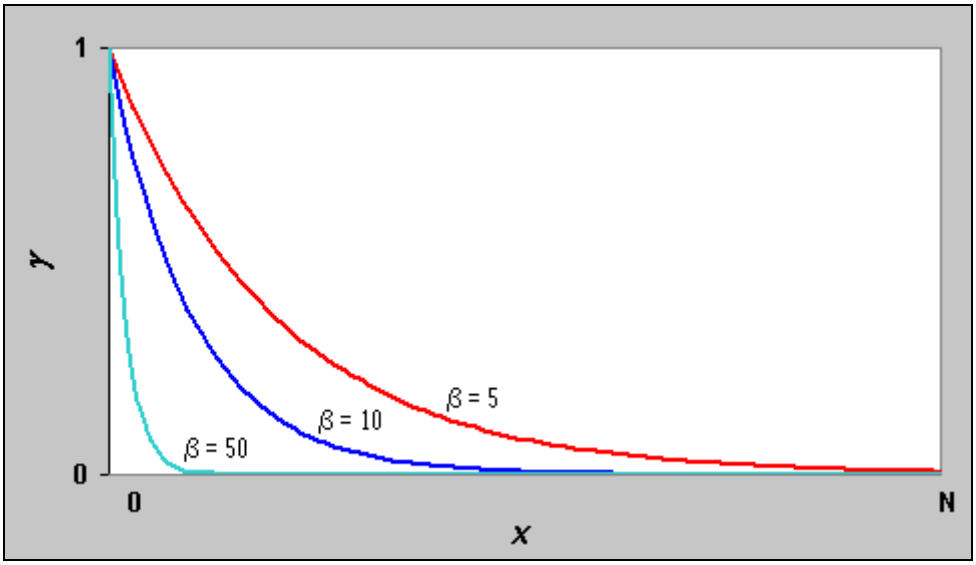


Figure 4-4. Gamma as a function of β.

The bright and shadow thresholds $I_{max}(x)$ and $I_{min}(x)$ are defined as:

$$I_{max}(x) = i_{max}(1+\gamma) \tag{4-9}$$

$$I_{min}(x) = i_{min}(1-\gamma) \tag{4-10}$$

All pixels with intensity values above $I_{max}(x)$ are considered brights, while all with intensities below $I_{min}(x)$ are considered shadows, and the corresponding bits in the bright and shadow GBs are set (Figure 4-5).

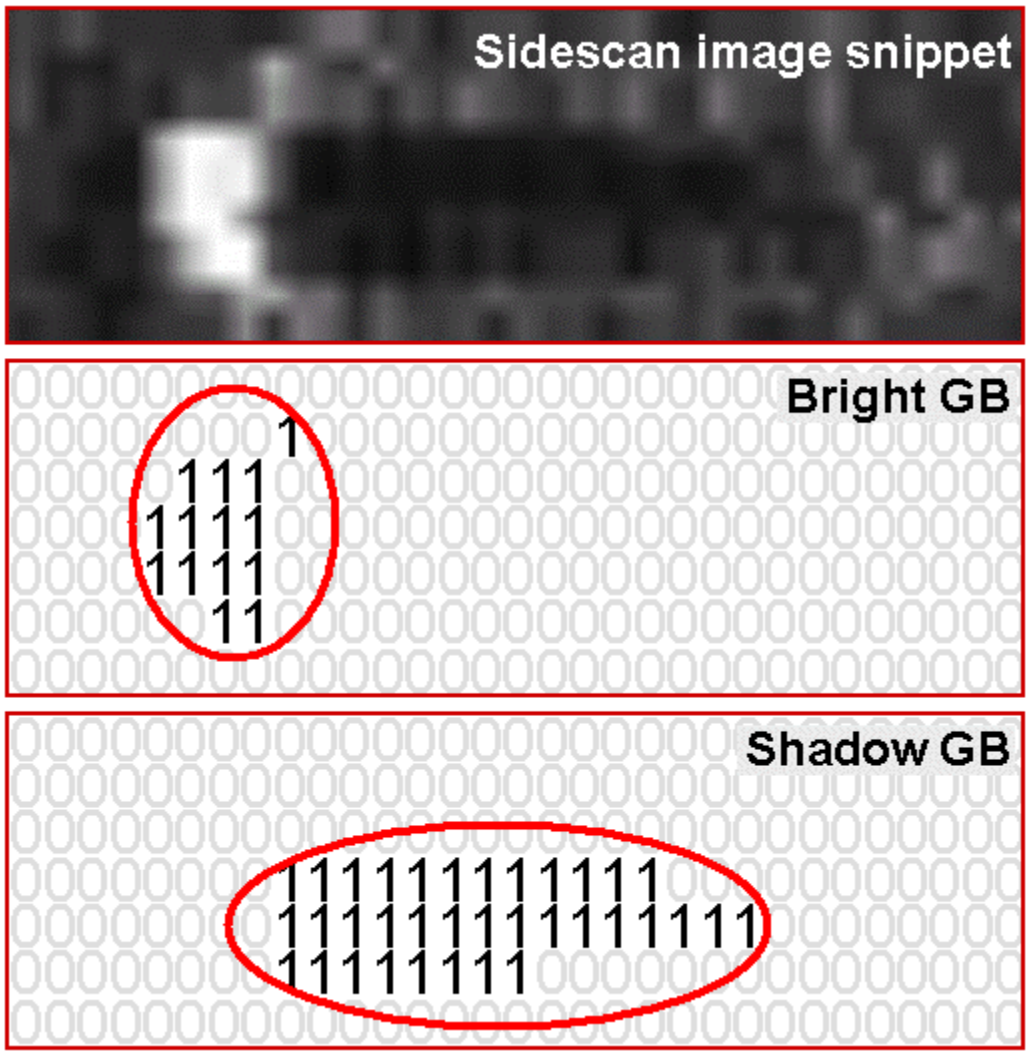


Figure 4-5. GBs are used to facilitate computer-aided detection of objects in SSI. Each row of bits in both GBs corresponds to a single scan line in the image. All pixels in the image with an intensity value greater than an upper threshold are considered "brights" and the appropriate bit in the bright GB is set. Likewise, all pixels in the image with an intensity value less than a lower threshold are considered "shadows" and the appropriate bit in the shadow GB is set.

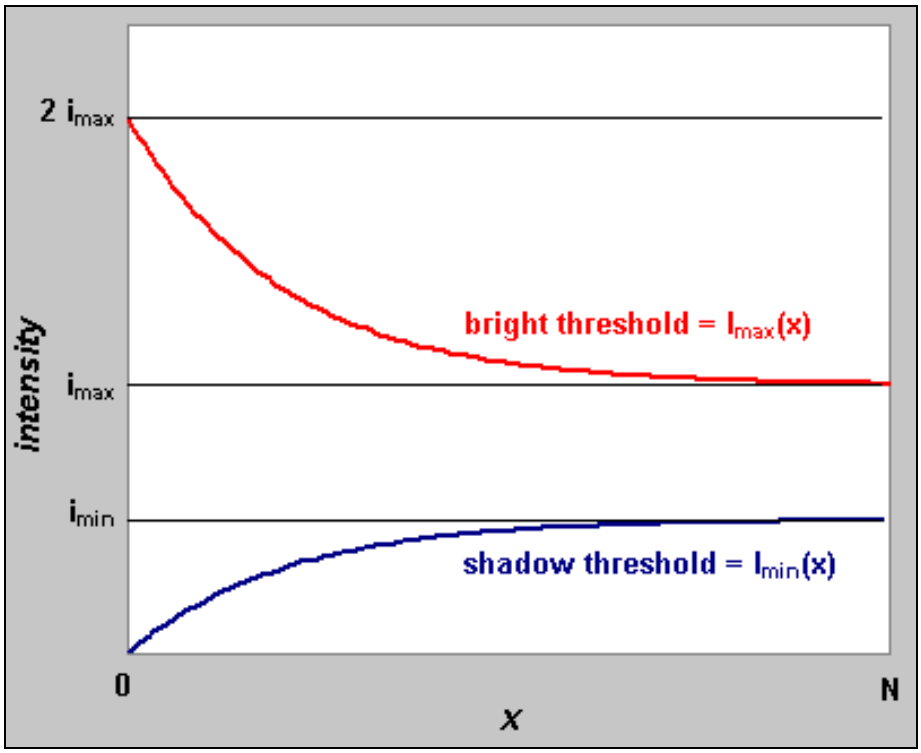


Figure 4-6. Intensity thresholds for brights and shadows.

Figure 4-6 illustrates how the intensity thresholds vary over x for a given γ. For example, the closer a pixel is to nadir (x = 0), the greater its intensity must be to be detected as a bright (Blondel and Murton, 1997), and the lower its intensity must be to be detected as a shadow. It is interesting to note that the two threshold curves do not diverge from their respective asymptotes (i_{min} and i_{max}) at the same rate as they approach nadir. This is by design, because shadows are more detectable than brights in SSI (Mignotte et al., 2000a, , Mignotte et al., 2000b; Reed et al., 2002; Reed et al., 2003). In other words, a single shadow threshold value (i_{min}) suffices for more values of x than a single bright threshold value (i_{max}).

Detecting Objects

The bright and shadow GBs are then examined from the outer edges of scan lines toward nadir to detect runs of shadows followed by runs of brights. (A valid object will have a shadow that falls away from nadir, with an adjacent bright closer to nadir). A summary of the steps to be performed to detect an object is as follows:

1) Search the shadow GB (1 scan line at a time, from the edge toward nadir) for a set bit. This is the shadow's start bit.
2) Continue examining both GBs to determine the end of the shadow. The concept of momentum is used here to allow some cleared bits to be considered part of the shadow (described below).
3) Determine whether the length of the shadow is consistent with the size of an object of interest. Specifically, shadows with lengths that fall within desired MILEC height ranges

– considering the distance from nadir, SSS height above seafloor, across-track resolution, and across-track error – are maintained in the shadow GB.

4) Search for a set bit in the bright GB (starting at the bit index corresponding to the end of the shadow). This is the start bit of a bright.
5) Continue examining both GBs to determine the end of the bright (again using momentum).
6) Determine whether the combination of the bright and shadow runs is consistent with an object's allowable size, given the parameters stated in step 3.
7) Translate the detected object's start bit and length into a 2D sliding window that will contain a summary of possible single-scan line object detections, which are examined to determine whether they are consistent with along-track MILEC dimensions.
8) Repeat steps 1-7 for all shadows and brights detected in the scan line.
9) Examine the sliding window to determine whether the previous series of scan lines defines an object.
10) Repeat steps 1-9 for each scan line.

The authors use the term "momentum" to describe a set of rules that were developed to govern how the configuration of set and cleared bits within the two GBs defines a run of brights or shadows. In the case of a shadow run, the first set bit in the shadow GB defines the start of a new run, at which point momentum is set to 1. Each subsequent set bit in the shadow GB increases momentum by 1, while a cleared bit decreases momentum by 1. When the momentum returns to 0, the last set bit encountered in the run defines the end of the shadow. For example, in Figure 4-7, momentum reached 0 four bits after the last set bit in the shadow run. The four clear bits are not included as part of the shadow.

One exception to these rules is that a cleared bit in the shadow GB cannot be included in the current run if the bright GB contains a set bit at that location. In other words, even if momentum has not yet reached 0, the run is terminated and the shadow is considered complete (minus any ending clear bits, as described above).

The process is repeated until the end of the scan line is reached. Momentum is used in the same manner to define bright spots in each scan line.

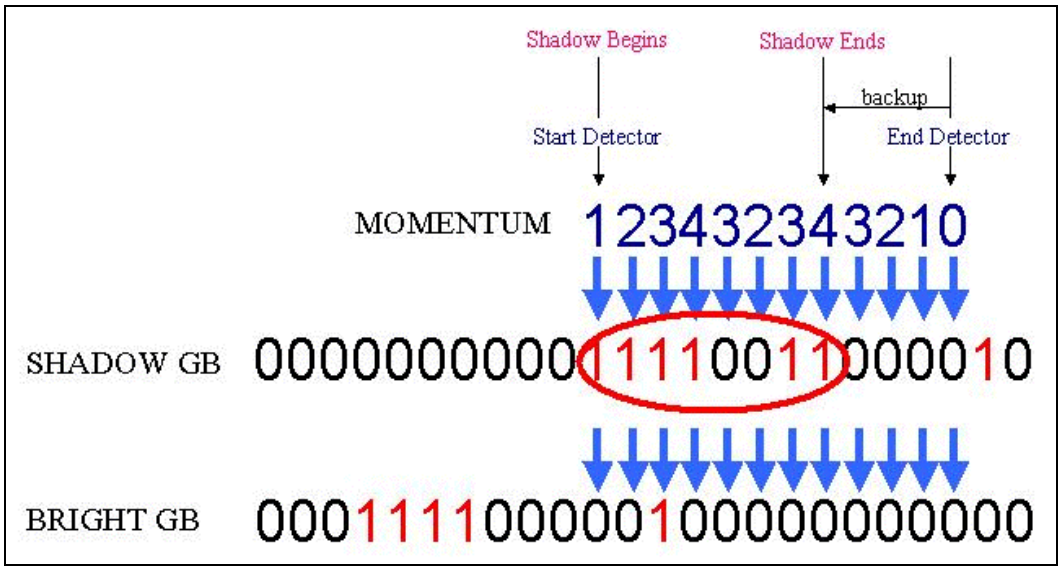


Figure 4-7. Momentum is used to determine how many cleared bits to include in a shadow or bright GB.

Generating Snippet in the Mini-UNISIPS (MU) Format

When the algorithm detects a MILEC, the scan line and the center of the object (defined here as the center of the bright spot) is recorded. The center point is expanded out by some preset value (e.g., 10 meters) on all sides and a snippet is created. The snippets are saved in the same UNISIPS format, called mini-UNISIPS, or MUs.

The latitude and longitude (LAT/LON) position of the MILEC's center can be computed with a great circle calculation, given the LAT/LON position of nadir for the scan line, the distance (in meters) from nadir to the center of the object, along-track and across-track resolutions, and the heading of the SSS at the time the line was collected.

Testing the CAD Algorithm

Dobeck (2004) compares the P_c verses P_{fa} for four different CAD algorithms developed by NSWC-PC, Lockheed Martin, Alphatech, and Raytheon (Figure 4-8). Dobeck first ran the four CAD algorithms separately on the input sidescan raw data. Each algorithm on its own detected at least 90% of the objects in a ground truth data set, and produced a confidence measure for each detection, but the false alarm rate was high. Dobeck's "Algorithm Fusion" method, which uses a weighted average of these measures, was shown to significantly reduce the false alarm rate while maintaining a correct detection rate of at least 90%.

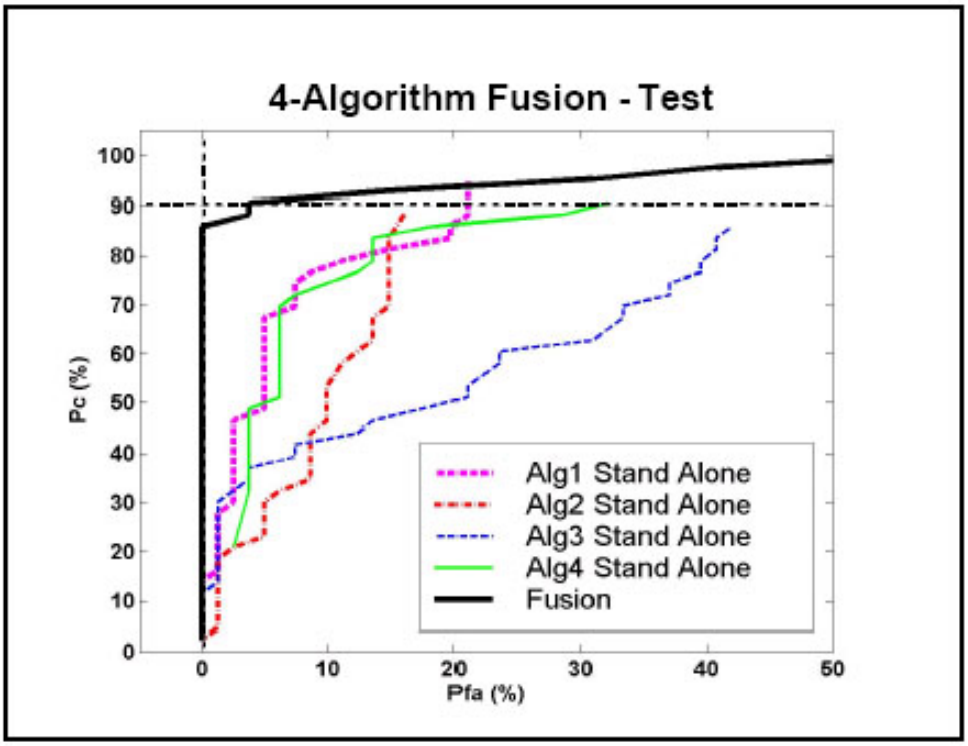


Figure 4-8. Dobeck's Fusion method increases correct detections and reduces false alarms, compared with each individual algorithm (figure 5 from Dobeck, 2004).

NAVOCEANO provided the author of this dissertation chapter with a copy of the UNISIPS data for the same geographical area used by Dobeck in these CAD tests, along with locations of 48 inert mines. Table 4-1 shows results from 25 runs of the CAD algorithm using these UNISIPS files while adjusting parameters A and B (which control how the lower and upper shadow/bright thresholds are computed).

Next, the author used the Layne Completion Method (LCM), discussed in chapter 6 of this dissertation, to reduce the false alarm rate (Figure 4-9 and Table 4-2). LCM adaptively filters each snippet produced by the detection stage into an image with only three intensity values: black (for shadows), white (for brights), and gray for everything else. The algorithm determines whether the snippet actually contains a shadow, and if so, the shadow is used to "complete" the bright spot. If no shadow exists, the snippet is flagged as a false alarm.

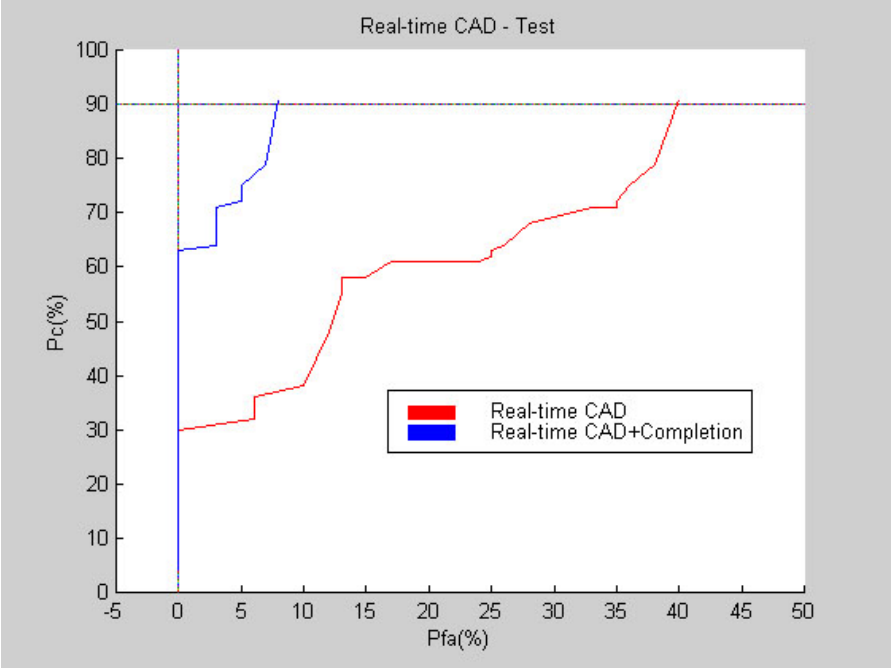


Figure 4-9. The author's real-time CAD algorithm produced detection and false alarm rates similar to the individual algorithms tested in Dobeck (2004). Combining the real-time CAD with the LCM resulted in greater correct detection rates and fewer false alarms, with results similar to Dobeck's Fusion method.

Table 4-1. Results from 25 runs of the CAD algorithm using the same UNISIPS files while adjusting parameters A and B (which control how the lower and upper shadow/bright thresholds are computed).

Run #	Total # Detections	# Correct Detections	Probability of Correct Detections (P_c)	# False Alarms	Probability of False Alarms (P_{fa})	A	B
1	2	2	4	0	0	0.100	1.000
2	7	7	15	0	0	0.108	1.020
3	9	9	19	0	0	0.116	1.040
4	14	14	29	0	0	0.124	1.060
5	16	15	31	1	6	0.132	1.080
6	17	16	33	1	6	0.140	1.100
7	18	17	35	1	6	0.148	1.120
8	20	18	38	2	10	0.156	1.140
9	26	23	48	3	12	0.164	1.160
10	30	26	54	4	13	0.172	1.180
11	32	28	58	4	13	0.180	1.200
12	33	28	58	5	15	0.188	1.220
13	35	29	60	6	17	0.196	1.240
14	38	29	60	9	24	0.204	1.260
15	40	30	63	10	25	0.212	1.280
16	40	30	63	10	25	0.220	1.300
17	42	31	65	11	26	0.228	1.320
18	46	33	69	13	28	0.236	1.340
19	51	34	71	17	33	0.244	1.360
20	52	34	71	18	35	0.252	1.380
21	54	35	73	19	35	0.260	1.400
22	56	36	75	20	36	0.268	1.420
23	61	38	79	23	38	0.276	1.440
24	61	38	79	23	38	0.284	1.460
25	73	44	92	29	40	0.292	1.480

Table 4-2. Results after running LCM to reduce the false alarm rate.

Run #	Total # Detections	# Correct Detections	P_c	# False Alarms	P_{fa}
1	2	2	4	0	0
2	7	7	15	0	0
3	9	9	19	0	0
4	14	14	29	0	0
5	15	15	31	0	0
6	16	16	33	0	0
7	17	17	35	0	0
8	18	18	38	0	0
9	23	23	48	0	0
10	26	26	54	0	0
11	28	28	58	0	0
12	28	28	58	0	0
13	29	29	60	0	0
14	29	29	60	0	0
15	30	30	63	0	0
16	30	30	63	0	0
17	32	31	65	1	3
18	34	33	69	1	3
19	35	34	71	1	3
20	35	34	71	1	3
21	37	35	73	2	5
22	38	36	75	2	5
23	41	38	79	3	7
24	41	38	79	3	7
25	48	44	92	4	8

Examples of Detected MILECs

Since 2001, NAVOCEANO has been evaluating and validating this CAD algorithm coupled with an automated clustering algorithm (Gendron et al., *submitted*) for use as a means to estimate clutter density of a geographic area for MIW. The algorithm has been tested on many data sets and demonstrated at several fleet exercises. The algorithm is constantly being improved, and a human factors study is planned to determine how analysts visually detect MLOs in SSI. The purpose of the study is threefold: 1) determine better ways to train analysts, 2) improve automated CAD, and 3) derive a metric to measure the performance of CAD algorithms.

Figures 4-10, 4-11, 4-12 and 4-13 show snippets of typical MILECs detected by the CAD algorithm.

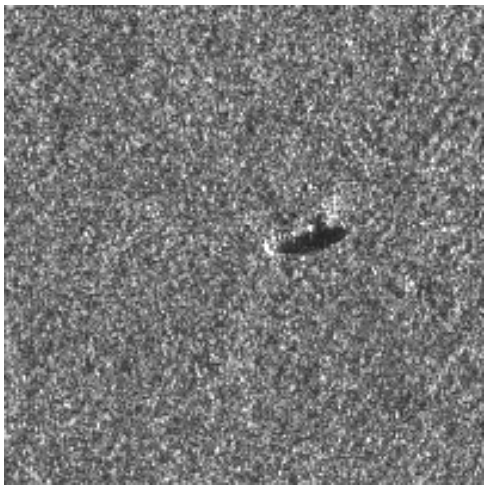


Figure 4-10. Snippet of cylindrical MILEC.

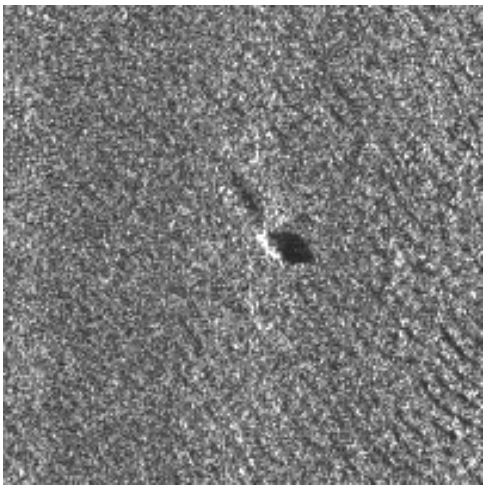


Figure 4-11. MILEC partially buried in sand.

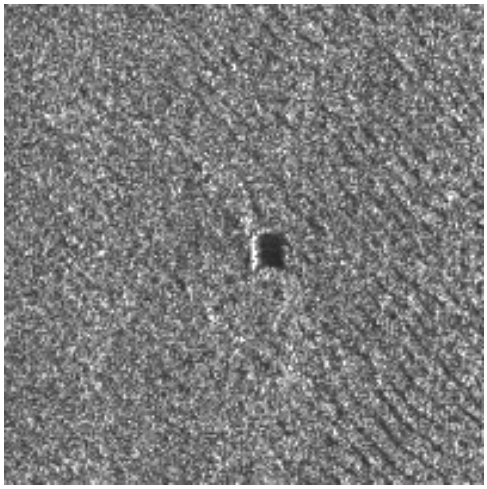


Figure 4-12. MILEC with rectangular shadow.

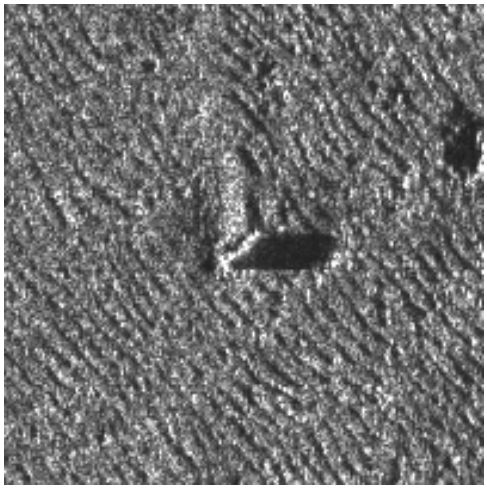


Figure 4-13. MILEC with quadrilateral shadow.

A persistent problem with any CAD algorithm is false detection (e.g., Figures 4-14 through 4-17). The features in Figure 4-14 are likely schools of fish close to the seafloor, which can mimic brights and shadows. Figure 4-15 presents objects that are too small to be MILECs (according to MIW doctrine), but were probably detected by the algorithm due to high speckle noise. Figure 4-16 is a rock. Another area of active CAD research is the reduction of false detections in SSI due to surface return (Figure 4-17).

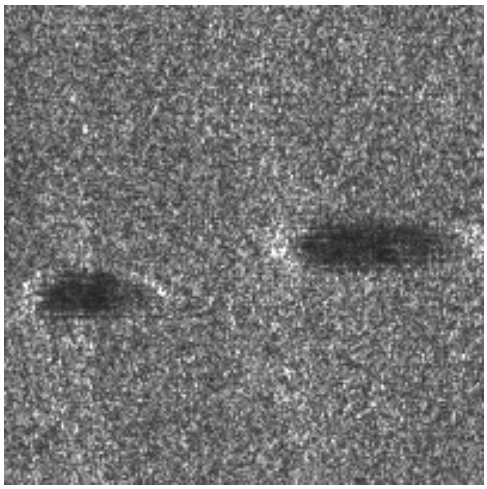


Figure 4-14. False detection: schools of fish.

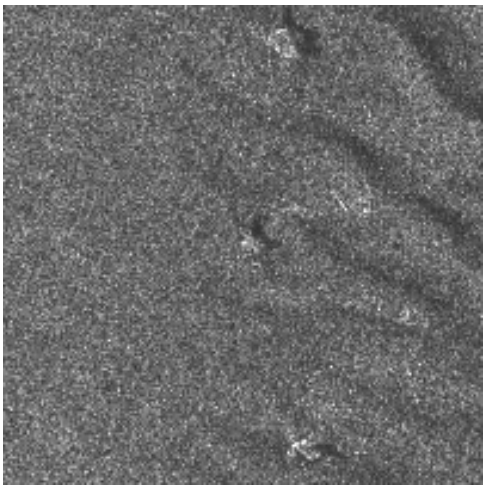


Figure 4-15. False detection: objects too small.

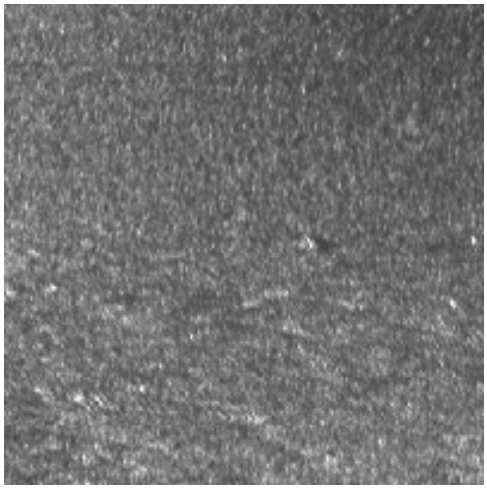


Figure 4-16. False detection: rock

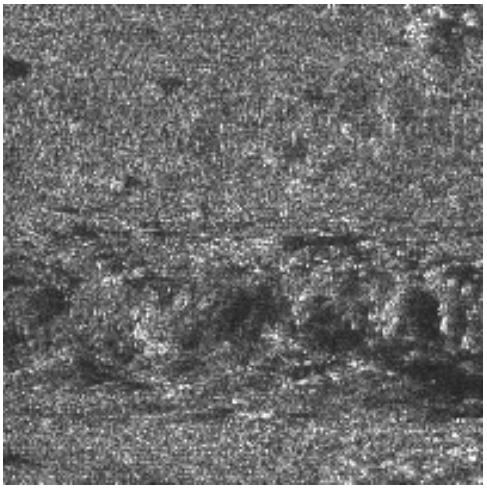


Figure 4-17. False detection: surface return.

Conclusion

Many different CAD algorithms have been developed, most of which are difficult to implement in real-time. This paper described a new CAD algorithm that is capable of operating in real-time as imagery is being processed (post-survey or during a survey, e.g., onboard an AUV).

The real-time CAD algorithm presented in this paper was shown to perform comparably to four other non-real-time CAD algorithms documented in previous research. The real-time CAD algorithm correctly detected 90% of the 48 inert mines in the test data set with a false alarm rate of 40%. The previously documented CAD algorithms exhibited false alarm rates between approximately 15% and 45% with a 90% correct detection rate (Dobeck, 2004).

When the real-time CAD algorithm was coupled with the LCM, the false alarm rate dropped to 8% (with 90% correct detections). This compares well with the Dobeck Fusion method, in which 90% of the mines were correctly detected with a false alarm rate of approximately 5%

(Dobeck, 2004). However, none of the comparison CAD algorithms, nor the Fusion method were implemented as in real-time.

The key limitation of the real-time CAD algorithm with LCM enhancement is that the LCM is not a real-time algorithm. The author plans to investigate ways to implement the LCM in real-time to better support MIW operations.

Acknowledgements

This work was performed in support of the Naval Oceanographic Office (NAVOCEANO) and funded by the Space and Naval Warfare Systems Command (SPAWAR) through the Commander, Naval Meteorology and Oceanography Command (CNMOC) and ONR under Program Element 0603207N**.** The authors thank Dr. Edward Mozley (SPAWAR), Dr. Bruce Northridge (CNMOC) and Mr. Michael M. Harris (NRL) for their support of this project. We thank Mr. Jim Hammack (NAVOCEANO) for his recommendations concerning the application of this algorithm in support of NAVOCEANO requirements. Any mention of commercial products or the use of company names does not imply endorsement by the U.S. Navy. This paper is approved for public release; distribution is unlimited.

References

Ackerman, L.V. and E.E. Gose (1972). Breast Lesion Classification by Computer and Xeroradiograph. *Cancer*, October, 3:4.

Anstee, S. (2001). *Removal of Range-dependent Artifacts from Sidescan Sonar Imagery*. DSTO Aeronautical and Maritime Research Laboratory, Fishermans Bend, Australia, Report DSTO-TN-0354. dsto.defence.gov.au/corporate/reports/DSTO-TN-0354.pdf.

Aridgides, T., M.T. Fernandez, and G.J. Dobeck (2004). Fusion Approach Investigation For Sea Mine Classification in Very Shallow Water. SPIE Proceedings of *Detection and Remediation Technologies for Mines and Minelike Targets VII*, Orlando, Florida, 4742:46.

Blondel, P. and B Murton (1997). *Handbook of Seafloor Sonar Imagery*. New York: John Wiley & Sons.

Ciany, C.M., Zurawski, W., and G.J. Dobeck (2004). Application of Fusion Algorithims for Computer-aided Detection and Classification of Bottom Mines to Shallow-water Test Data. SPIE Proceedings of *Detection and Remediation Technologies for Mines and Minelike Targets VII, Orlando*, Florida, 4742:45.

Cronin, D., M. Broadus, B. Reed, S. Byrne, W. Simmons, and L. Gee (2003). Hydrographic Work Flow – From Planning to Products. Proceedings of the *U.S. Hydro 2003 Conference*, March 24-27, Biloxi, Mississippi.

Dobeck, G.J. (2004). Algorithm Fusion: An Overview - Combining Multiple Detection and Classification Algorithms. Proceedings of the *Sixth International Symposium on Technology and the Mine Problem*, Naval Postgraduate School, Monterey, CA. May 9-13.

Dobeck, G.J., J.T. Cobb (2004). Fusion of Multiple Quadratic Penalty Function Support Vector Machines (QPFSVM) for Automated Sea Mine Detection and Classification. SPIE Proceedings of *Detection and Remediation Technologies for Mines and Minelike Targets VII*, Orlando, Florida, 4742:44.

Fish, J. P. and Carr, H. A. Carr (1990). *Sound Underwater Images: A Guide to the Generation and Interpretation of Sonar*. American Underwater Search and Survey, Massachusetts: Lower Cape Publishing.

Foley, J.D. and A. van Dam (1994). *Fundamentals of Interactive Computer Graphics*. Adison/Wesley Publishers: Massachusetts.

Gendron, M., G. Layne and M. Lohrenz, (*submitted*). The Geospatial Bitmap (GB) Clustering Algorithm. Naval Research Laboratory, Stennis Space Center, Mississippi, submitted to the *Journal of Discrete Algorithms*.

Gendron, M.L., P. B. Wischow, M.E. Trenchard, M.C. Lohrenz, L.M. Riedlinger, M. Mehaffey (2001). *Moving Map Composer*. Naval Research Laboratory, US Patent No. 6,218,965.

Giger, M.L. (2001). *CE: Mammography 4: Update on Computer-Aided Diagnosis in Mammography Handout*. AAPM meeting, Salt Lake City, Utah.

Huff, L., N. Langhorne, R. Quiqley, J. Weintroub, R. Kuwahara, J. Preston, and A. Hart (1991). *Multibeam Foucused Sonar (MBFS) Experiments. Plymouth, UK, June-July 91*. Klein Associates, Inc.

Iman, R. L. (1999). *A Data-Based Approach to Statistics (Statistics)*. Belmont: Buxbury Press.

Ioup, Gendron, Bourgeois and Ioup (2001). Wavelet Denoising of Sidescan Sonar Images. Abstract of *the J. Acoust. Soc. Am*., 110.

Klein Associates, Incorporated (2004). Klein Associates, Incorporate. Website: http://www.kleinsonar.com.

Kvitek, R., P. Iampietro, E. Sandoval, M. Castleton, C. Bretz, T. Manouki and A. Green (1999). *Final Report: Early Implementation of Nearshore Ecosystem Database Project*. Monterey Bay: SIVA Resource Center, California State University.

LeBlanc, M.J., E.S. Manolakos (1991). Neural Networks for Sidescan Sonar Automatic Target Detection. *Neural Networks for Signal Processing*, 1, Princeton, September.

Lingsch, W.C. and S.C. Lingsch (2001). Sonar Image Change Detection as a Mine Counter Measures Tactical Tool. Abstract for the *Shallow Survey 2001 Conference,* Sidney, Australia.

Mignotte, M., C. Collet, P. Perez and P. Bouthemy (2000a). Markov Random Field and Fuzzy Logic Modeling in Sonar Imagery: Application to the Classification of Underwater Floor. *Computer Vision and image Understanding*, 79.

Mignotte, M., C. Collet, P. Perez and P. Bouthemy (2000b). Sonar Image Segmentation Using an Unsupervised Hierarchical MRF Model. *IEEE Transactions on Image Processing*, 9:7, July.

Murino, V. (2001). Reconstruction and Segmentation of Underwater Acoustic Images Combining Confidence Information in MRF Models. *Pattern Recognition*, 34:5.

Neretti, N., N. Intrator and Q. Huynh (*Submitted*). Target Detection in Side-scan Sonar Images: Expert Fusion Reduces False Alarms. http://www.physics.brown.edu/users/faculty/intrator/research.html.

Reed, B. (2001). Data, Data Everywhere and Nary a Bit to Drop. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

Reed, S., E. Dura, D.M.Lane, Y. Petillot, J. (2002). Extraction and Classification of Objects from Sidescan Sonar Bell. *IEEE Workshop on Nonlinear and Non-Gaussian Signal Processing*, 8-9 July.

Reed, S., Y. Petillot, J. Bell (2003). An Automatic Approach to the Detection and Extraction of Mine Features in Sidescan Sonar. *IEEE Journal of Oceanic Engineering*, 28:1, January.

Reed, S., Y. Petillot, and J. Bell (2001). Unsupervised Mine Detection and Analysis in Sidescan Sonar: A Comparison of Markov Random Fields and Statistical Snakes. Processings of *CAD/CAC 2001*, Halifax.

Russ, J. C. (2002). *The Image Processing Handbook, Fourth Ed. New York:* CRC Press.

Sams T. and F. T. Agerkvist (2004). Coherence Enhancing Diffusion Filtering of Sidescan Sonar Images. Submitted to the *IEEE Transactions on Image Processing*, http://www.nbi.dk/~sams/.

Winsberg F., Elkin M., Macy J. et al. (1967). Detection of Radiographic Abnormalities in Mammograms by Means of Optical Scanning and Computer Analysis. *Radiology 1967*, 89.

Chapter 5 - A Computer-Aided Search (CAS) Algorithm

Marlin L. Gendron, Maura C. Lohrenz and Geary J. Layne
Naval Research Laboratory (NRL), Code 7440.1, Stennis Space Center, MS

Juliette Ioup
University of New Orleans (UNO), New Orleans, LA

Abstract

The author presents his computer-aided search (CAS) algorithm to help Mine Warfare analysts locate mine-like features that are geospatially close to previously detected features (stored in historical databases) and perform change detection. The CAS algorithm uses a modified quadtree (MTREE) and geospatial bitmap (GB) data structure, both developed by the author, to successfully search for features based on feature locations that may have been corrupted by position error inherent in sidescan sonar data. The processing time required for the CAS algorithm to perform location queries and return all spatially close data points within a region of error was recorded for four randomly generated data sets. The resulting times were compared against the processing times obtained from a Great Circle Distance Method (GCDM) search algorithm operating on the same data points. CAS algorithm performed better for all test cases than the GCDM. The radius of the error region had no significant effect on time for either algorithm. As the number of points increased, the difference in processing time between the two algorithms became more pronounced (e.g., GCDM ran only 1.05x slower than CAS with 600 points, but GCDM ran 1.75x slower than CAS for 10,000 points). This implies that the CAS algorithm is a better choice than GCDM for geospatial search applications, particularly on very large data sets.

Acronyms

1D	one-dimensional
2D	two-dimensional
AOI	area of interest
AM	area matching
ACDC	Automated Change Detection and Classification
AUV	autonomous underwater vehicle
CAC	computer-aided classification
CAD	computer aided detection
CAI	computer-aided identification
CAS	computer-aided search
CNMOC	Commander, Naval Meteorology and Oceanography Command
DVS	Doppler velocity sonar
FM	feature matching
GB	geospatial bitmap

GBS	Global Positioning System
GCDM	Great Circle Distance Method
INS	inertial navigation systems
LAT/LON	latitude/longitude
MBR	minimum-bounding rectangle
MCDC	Master Contact Database
MILCO	mine-like contact
MILEC	mine-like echoes
MIW	Mine Warfare
MTREE	modified quadtree
NAVOCEANO	Naval Oceanographic Office
NRL	Naval Research Laboratory
SSI	sidescan sonar imagery
SSS	sidescan sonar system
SPAWAR	Space and Naval Warfare Systems Command
UNISIPS	Unified Sonar Image Processing System
UUV	unmanned underwater vehicle

Introduction

The process of tracking the change of detected features between historical remotely-sensed imagery and recently collected imagery is called "change detection". Change detection is usually performed manually by analysts. The term "digital change detection" applies when computer algorithms are used to aid analysts perform change detection tasks on data collected by a remote sensing device (Singh, 1989). Digital change detection is a major application performed on remote-sensed data (Rosin, 1994).

An area that can greatly benefit from change detection on remotely sensed data is Mine Warfare (MIW). During MIW operations, assault lanes must be cleared of mines to allow naval ships to pass safely. Analysts perform change detection by visually comparing historical high resolution imagery collected by a sidescan sonar system (SSS) with recently collected sidescan sonar imagery (SSI) in an attempt to identify objects (which might be explosive mines) placed at sea since the last time the area was surveyed.

The objective of MIW change detection is to match new features detected in SSI with historical features that analysts have archived in a database and divers, or autonomous machines, have visually investigated and confirmed. Eliminating features that have already been investigated saves clearance time, but problems such as large data volume, navigational errors (Kenny et al. 2001), and sonar towfish instabilities can impede these time savings (Lingsch and Lingsch, 2001), with position error being the biggest hindrance (Schwab et al., 1991).

This paper presents a new computer-aided search (CAS) algorithm, developed by the author, that is part of a larger digital change detection system, called the Automated Change Detection and Classification (ACDC) system. CAS aids MIW analysts to find those features archived in a historical database, which are geospatially close to newly detected features in the SSI, based on latitude/longitude (LAT/LON) locations of the features.

ACDC consists of five primary stages, each of which is performed by a separate algorithm. The first component is a computer aided detection (CAD) algorithm that automatically detects mine-like echoes (MILECs) by looking for bright spots and shadows in newly collected SSI (Figure 5-1). The second component is a computer-aided classification (CAC) algorithm that attempts to automatically classify the features (i.e. mines, rocks, sand waves) by a unique "completion" method. The third component is the CAS algorithm. The last two components are computer-aided identification (CAI) algorithms. These algorithms are currently being developed by the author and will automatically perform feature- and area-matching (FM and AM) operations to verify whether the same object was observed in the past (change detection).

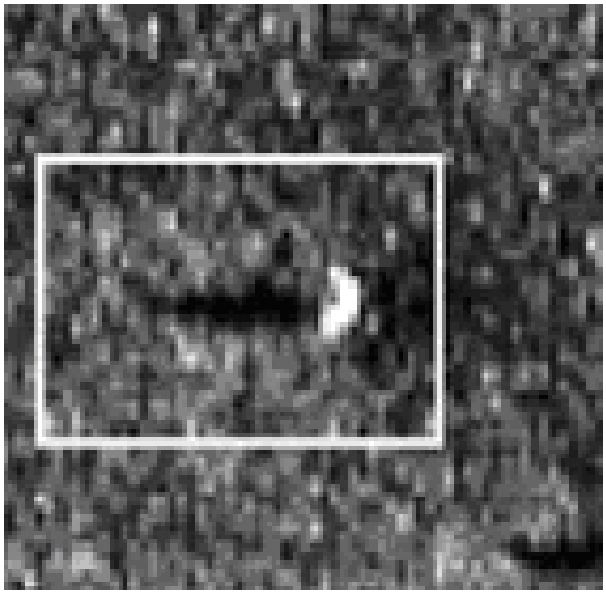


Figure 5-1. Snippet of Mine-like Echo

One factor that can greatly affect the accuracy of change detection is the geometric co-registration between two images being compared (Howarth and Wickware, 1981; Mas, 1999; Singh, 1989). Correctly matching corresponding features is crucial to accurately registering the images (Kamgar-Parsi et al., 1991). The accurate co-registration of two or more SSI, in particular, is inherently difficult, due to potentially large SSS position errors (Joannides and Le Gland, 1993; Kamgar-Parsi and Kamgar-Parsi, 1997), which can distort the apparent locations of SSI features.

The CAS algorithm presented here is capable of quickly and efficiently searching for spatially close features in SSI with large position errors, using a modified quadtree (MTREE) and the geospatial bitmap (GB) data structures (Gendron et al., 2001), both developed by the author.

The section below discusses position error in SSI, and emphasizes the need for a CAS algorithm that can quickly geospatially search large historical databases when position error is present. The next section stresses the importance of selecting the correct data structure for the CAS algorithm and gives some background on data structure in general that could be used in the CAS. The author's MTREE and Geospatial Bitmaps (GB) data structure, which are ultimately used in the CAS, are also presented in their own sections of this chapter.

Position Error in Sidescan Sonar Imagery (SSI)

The imagery used for this research comes from the Klein 5000 SSS (Figure 5-2), developed by Klein Associates, Inc (Klein Associates, Incorporated, 2004). The center frequency of the sonar is 455 kHz with a pulse length of 50 to 200 μ sec that gives 10 or 20 cm along-track resolution. The across track resolution is 36 cm at a maximum of 150 m for an array 120 cm long.

The Klein 5000 (Figure 1-6) is usually towed at 10 knots with a 300 m swath at an altitude of 25 m. The sonar tow fish is equipped with standard nominal heading, pressure, pitch, altimeter and roll sensors. The sonar is compact and relatively light and easy to use. The body length is 194 cm and the diameter 15.2 cm. It only weighs 70 kg in air (Klein Associates, Incorporated, 2004).

Figure 5-2. Klein 5000 towfish

During a typical survey, the Klein 5000 raw data is collected as 12-bit intensity values, which are downsampled to 8 bits using software developed by the Naval Oceanographic Office (NAVOCEANO) and stored together as a scan line in the Unified Sonar Image Processing System (UNISIPS) format to produce SSI. Each scan line is stored as a separate record in the UNISIPS file. In addition to the intensity values, each UNISIPS record also contains the sonar heading, speed, depth above the seafloor, and LAT/LON location for the center of the scan line, called nadir (Figure 5-3).

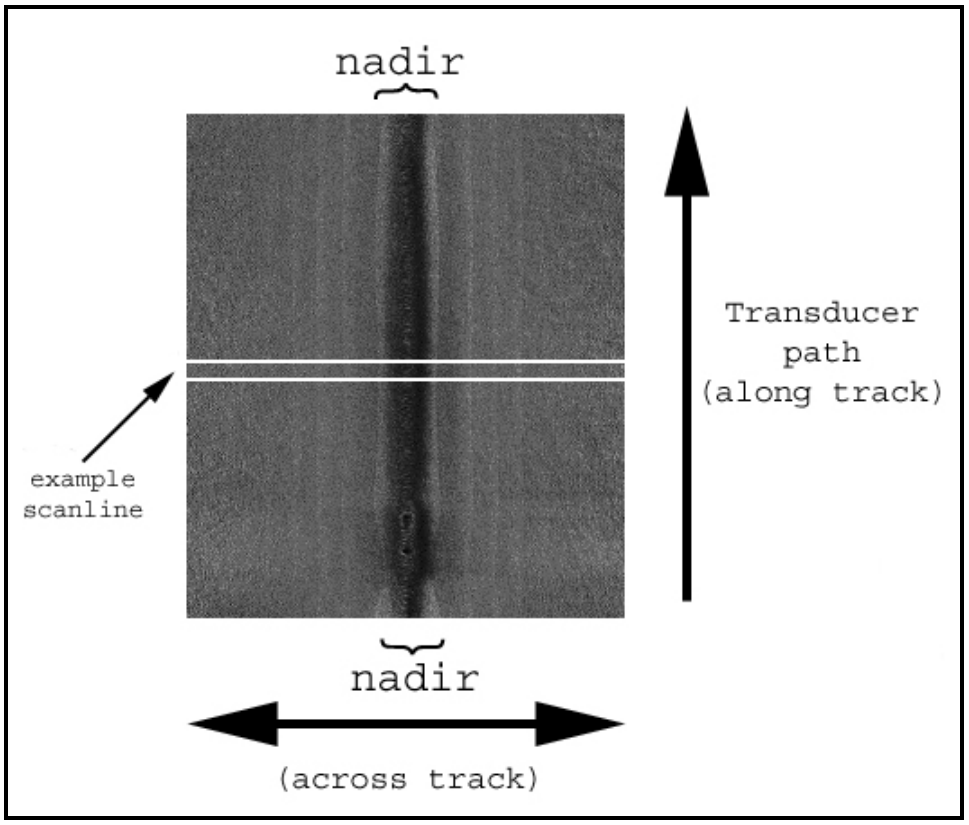


Figure 5-3. Sidescan Sonar Imagery (SSI)

In the early 1990's, NAVOCEANO began collecting and producing SSI, and analysts started cataloging features of interest worldwide. NAVOCEANO created a Master Contact Database (MCDB) to store these features as snap-shots called snippets (Figure 5-1). The MCDB also contains feature attributes, including the estimated LAT/LON feature position, dimensions and a detailed description.

The LAT/LON position of a feature on any given scan line can be estimated using a great circle calculation, given the following information: 1) the LAT/LON position of nadir (i.e., the position of the towfish), 2) the heading at nadir (i.e., the direction the towfish is traveling), and 3) the distance from nadir to the feature (Figure 5-4). The accuracy of the towfish location (1, above) has the greatest impact on the accuracy of the feature position (Schwab et al., 1991; Neill, 1999).

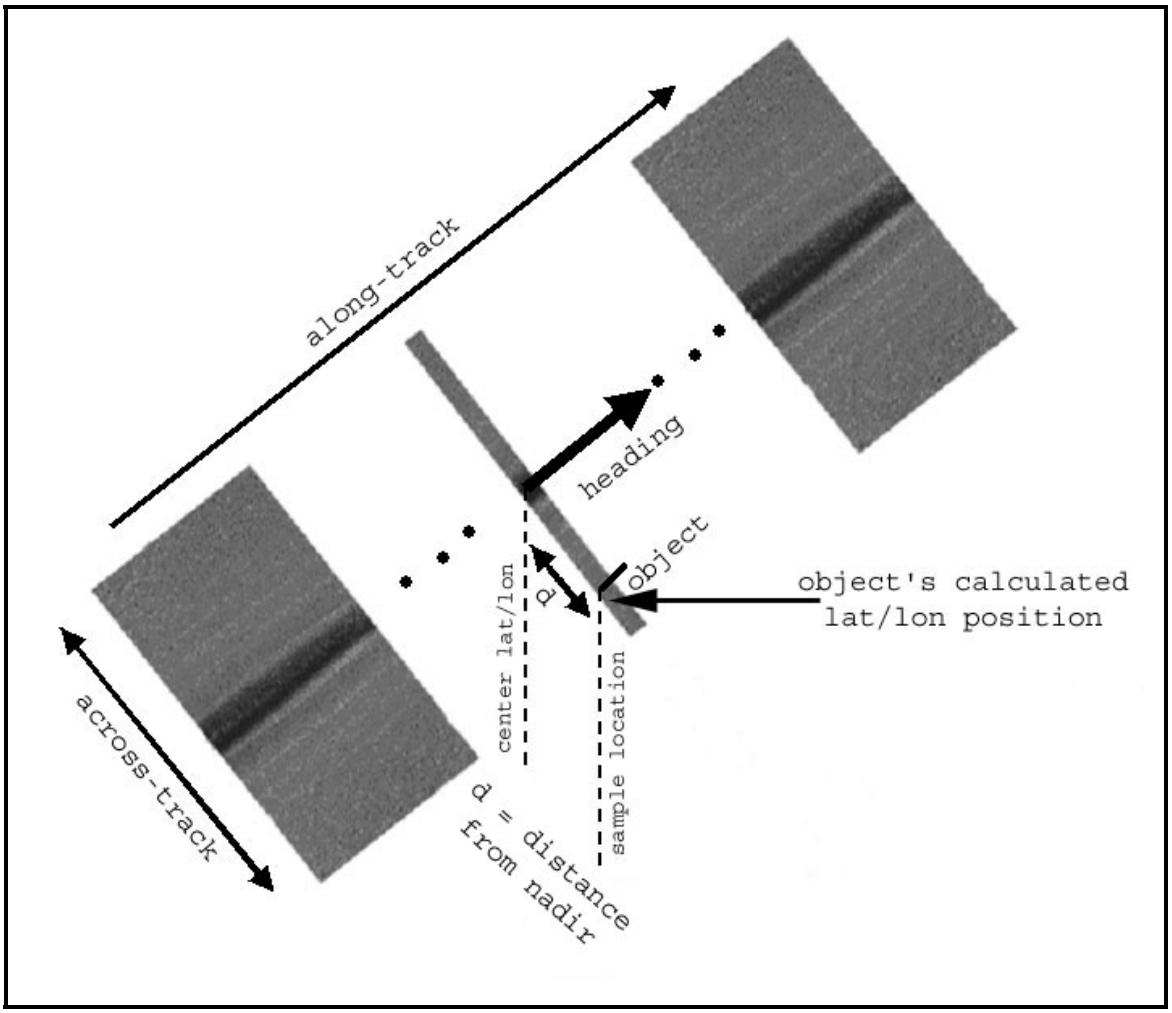


Figure 5-4. Estimated lat/lon position of an object on a scan line.

The Global Positioning System (GPS) can be used to indirectly estimate towfish location. GPS is made up of a constellation of 24 satellites placed in orbit around Earth by the United States Department of Defense. A minimum of 3 satellites is required to estimate a 2D position. The LAT/LON coordinate of a given location can be estimated by a GPS receiver, which measures the time it takes a signal to travel from each of the satellites to the location (King and Egbert, 2003).

GPS signals are rapidly attenuated in water; therefore, positions of objects below the surface (such as a SSS towfish) cannot be estimated directly using GPS (Joannides and Le Gland, 1993; Kamgar-Parsi and Kamgar-Parsi, 1997). Autonomous underwater vehicles (AUVs) and unmanned underwater vehicles (UUVs) that rely on GPS for position estimation must surface frequently to obtain a GPS "fix" and interpolate between the fixes to estimate their position when underwater. AUVs also use inertial navigation systems (INS) and Doppler velocity sonar (DVS) sensors, but position error with these systems greatly increases as a function of travel distance (Feder et al., 1998).

The LAT/LON position of towed SSS systems is determined by GPS measurements onboard the towing vessel and an estimation of the length of the tow cable. If the GPS antenna is not mounted at the point where the tow cable attaches to the platform, the "antenna offset" must be included in the calculation (Figure 5-5).

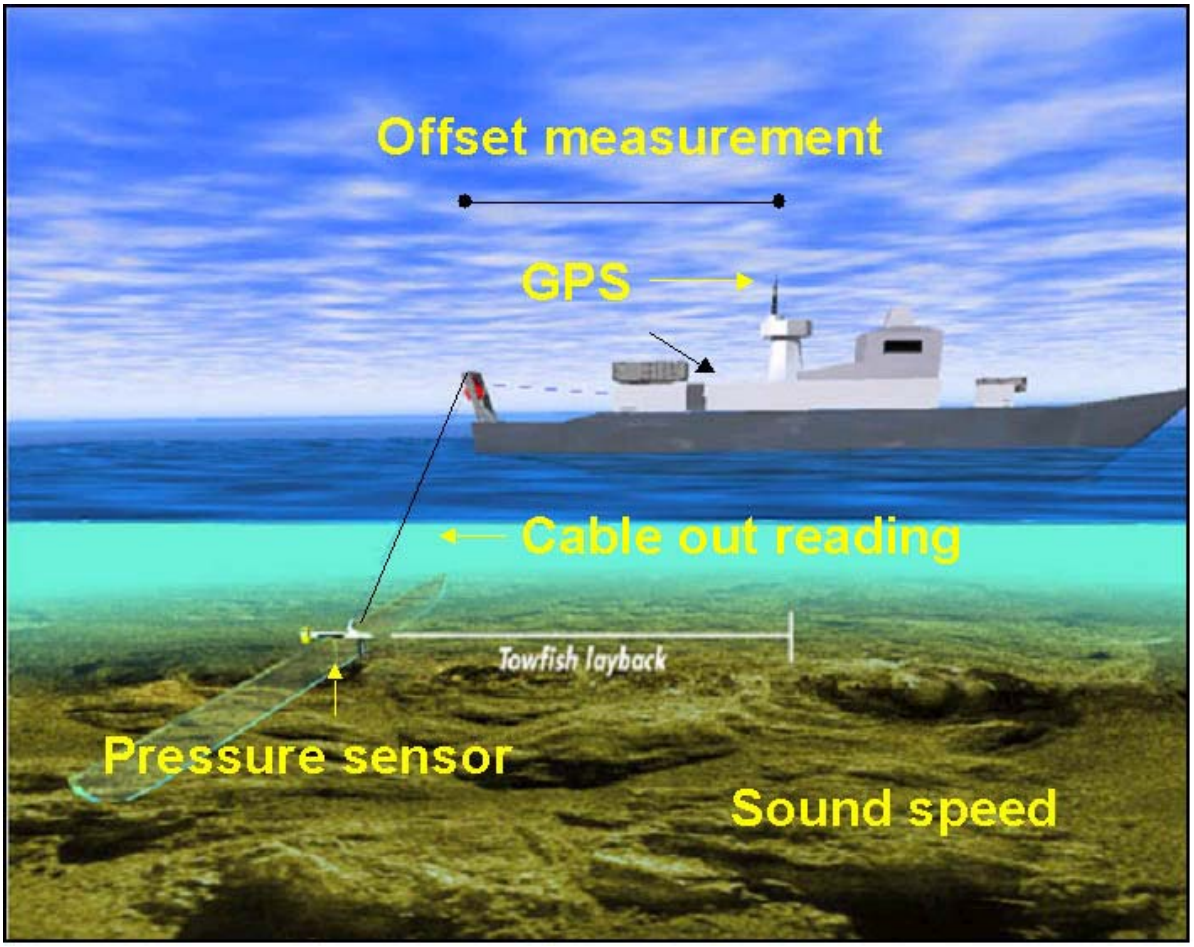


Figure 5-5. Location of the towfish (Klein Associates, Incorporated, 2004).

Assuming the cable is always taut (i.e. in a straight line), the length of the cable can be computed via the Pythagorean theorem. This is usually not the case, and a more complicated equation, called a cable layback model, must be used to estimate the distance from the tow-platform to the towfish. Errors in the layback model will increase if the length of the cable is allowed to change frequently during a survey (Neill, 1999). These errors – between the estimated LAT/LON location of an object observed during one pass with a SSS and the estimated location on subsequent passes over the same object – have been observed to be 15 m or greater during actual surveys (Schwab et al., 1991).

Data Structures

A central concept in computer science is the data structure, or method by which algorithms organize and access data. Often, the efficiency of an algorithm depends directly on the choice of data structure. Picking an appropriate data structure – based on the data (e.g., 2-dimensional

imagery, text lists, etc.) – ensures that the algorithms that operate on the data will be more efficient, using fewer resources (e.g., memory) and less processing time (Jackson, 1978; Chon et al., 2002).

Various data structures capable of storing LAT/LON values of historical SSI features were considered for the CAS algorithm. Speed of retrieval was the primary concern. Candidates included arrays, link-lists, hash tables, and trees. Arrays can be static (where the length is preset, and memory is allocated only at the start of the program) or dynamic (where the length can change, and memory is allocated during run-time). Data stored in arrays can be retrieved directly only if the exact index is known; otherwise, each array element must be examined until the desired data element is located.

In some cases, a hash table is more efficient. A hashing function computes an index into the hash table, called a hash key, based on some aspect of the data element to be stored. Collisions occur when the same hash key is computed from two different elements. When an element is hashed to a empty location, it is stored. Subsequent elements hashed to the same location are stored nearby (often by adding 1 to the index until an empty memory location is found) and retrieved later in the same fashion.

Link-lists are dynamically allocated structures that contain data nodes linked together by memory pointers. Link-lists can grow and shrink in size during run-time and the nodes can contain many data fields to store attributes of the data. Because link-lists are dynamic, they cannot be indexed directly to locate a data element: each node must be searched sequentially until the data element is located.

Trees are naturally efficient at representing hierarchical data, particularly well suited to the implementation of databases, and allow for the quick retrieval of data items based on spatial location (Guttman, 1984). Rooted trees are comprised of a set of nodes (vertices) and a set of arcs (edges) that link a parent node to one of its child nodes. Each root node has one parent. Any node can be reached by following a unique path of arcs from the root. When vertices are interconnected, the tree is considered bi-directional.

Binary trees are the simplest kind of tree, in which each parent node has at most two children. Binary trees and hash tables are most useful for one-dimensional (1D) data (Deitel and Deitel, 1994). Two-dimensional (2D) data, such as maps and feature data, is better stored in quadtrees (Tobler and Chen, 1986; Kardos et al., 2003). A quadtree is similar to a binary tree, but parent nodes have at most four vertices.

Because the CAS algorithm needs to store and retrieve 2D geographical datg, the quadtree data structure was chosen as the starting point in the design of the CAS algorithm. To improve efficiency and speed of retrieval, the author combined GBs with the quadtree to create a new data structure called a MTREE, which allows for storage and quick retrieval of the locations and related attributes of historical SSI features. Other modified trees, such as the R-tree (Gutt, 1984), have been implemented successfully in algorithms that locate spatially close points based on a query (Roussopoulous et al., 1995). The next two sections describe the GB, quadtree, and MTREE data structures in more detail.

Geospatial Bitmaps (GBs)

Bitmaps are 2D binary structures, in which bits are turned on (set) or off (cleared) and the row and column of each bit gives it a unique position. This concept is extended to construct GBs, in which each bit represents a unique location on the Earth at a given map scale. A set bit denotes that data exists at that location, while a cleared bit indicates the absence of any data at that location. Although a GB is defined for the entire world at a given map scale, memory is only allocated dynamically as groups of spatially close bits are set, resulting in a compact data structure that supports very fast Boolean and morphological operations. The author patented the GB and supporting algorithms (Gendron et al., 2001).

The GB data structure minimizes computer memory by breaking its bitmap into submaps (Figure 5-6). The submaps are 2D tiles of equal size, which are actually pointers to byte arrays. The individual bits represent the cells of the bitmap using bit masking. To determine the number and size of the submaps, a "best fit" algorithm is used. The algorithm attempts to optimize the number, size and shape of the submaps, to minimize wasted space and processing time.

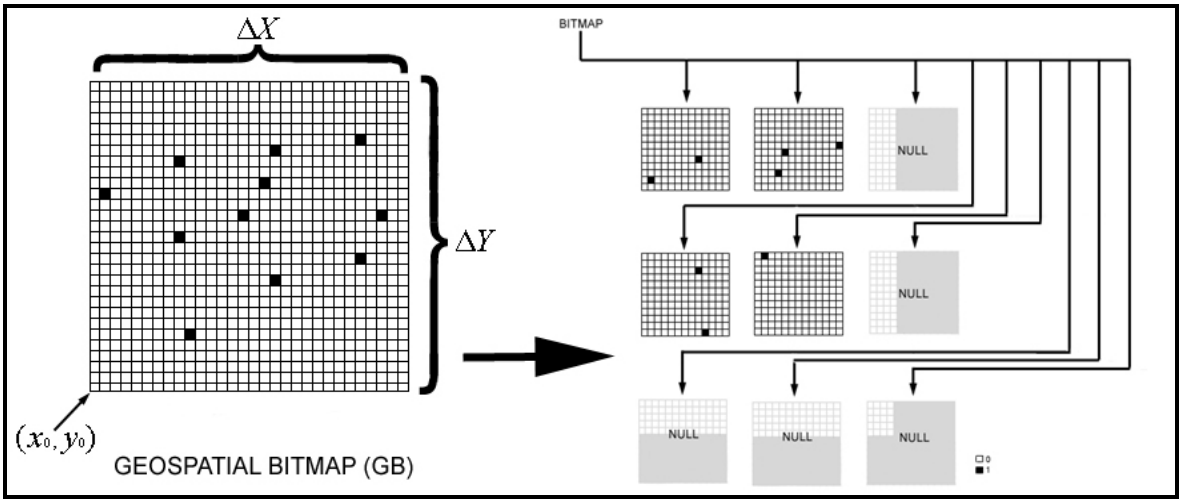


Figure 5-6. Geospatial Bitmap (GB)

Quadtrees

The fundamental structure of a quadtree is the node. When a quadtree is used to store 2D data, in which the LAT/LON location of the data is one of the primary keys, a node represents one geographical location and can contain attribute fields that describe certain aspects of that location. The precision of the location is driven by the number of levels in the quadtree and the geographic area represented by the root node. A node also contains memory pointers to four other nodes that may or may not exist. At level 0 of the quadtree there is a single node, called the parent node, and the four nodes below in level 1 are called children. Each child node represents a quarter of the total geographic area represented by its parent node. These children can also be parent nodes, each of which can point to four additional children at the next level, etc. Nodes with no children are called leaf nodes (Deitel and Deitel, 1994).

Figure 5-7 illustrates how a 6-level quadtree spatially divides a 1-degree LAT/LON cell, and Figure 5-8 depicts four levels of a quadtree. Note that if a node has no children or does not contain any data, it does not exist; i.e., memory has not been allocated, and the memory pointer within its parent node is equal to 0, or null.

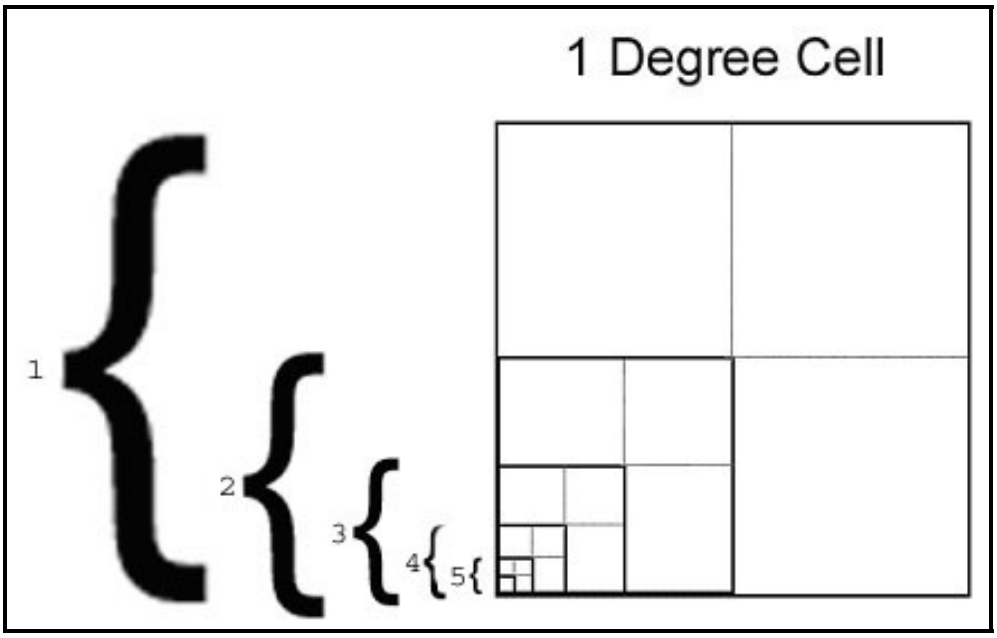


Figure 5-7. This is a 6-Level quadtree where each level contains four children nodes except for level 0. Level 0 is the root node which contains only a memory pointer to the four children nodes of level 1.

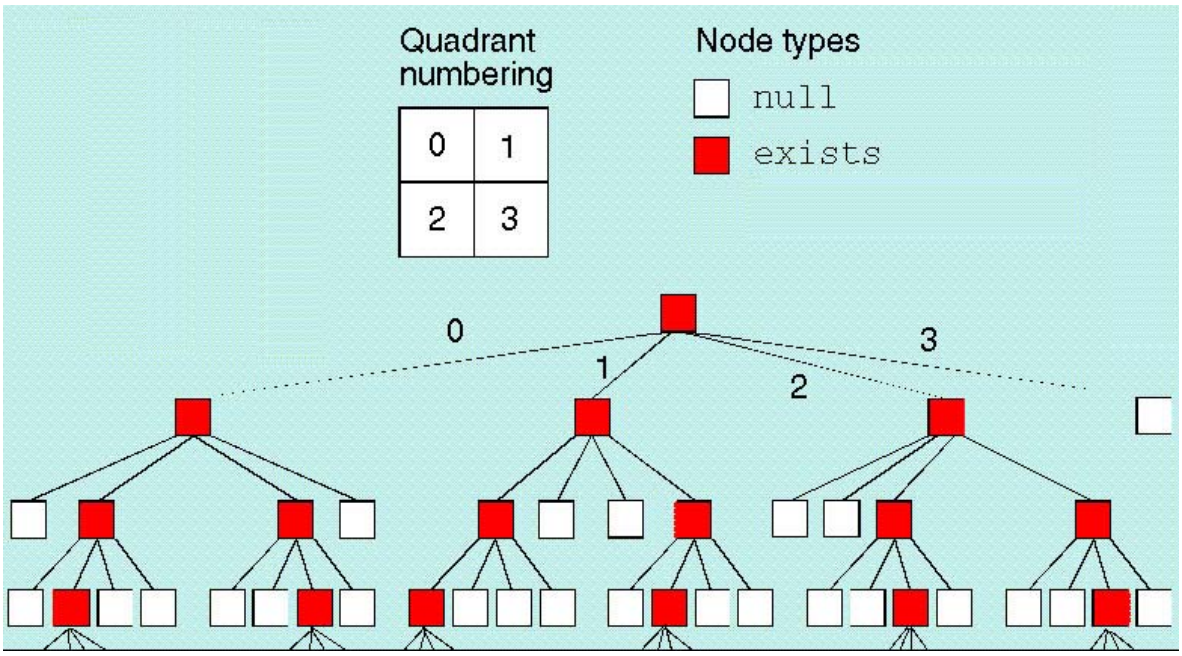


Figure 5-8. Four levels of a sample quadtree.

From any level, to determine which of the four nodes represents the geographical area for a given LAT/LON point, denoted (Y,X), the algorithm compares (Y,X) with the center LAT/LON point (Y_C,X_C), of each node at that level, as follows:

$$\begin{aligned}
&\text{Node } 0 : X < Xc \text{ AND } Y >= Yc \\
&\text{Node } 1 : X >= Xc \text{ AND } Y >= Yc \\
&\text{Node } 2 : X < Xc \text{ AND } Y < Yc \\
&\text{Node } 3 : X >= Xc \text{ AND } Y < Yc
\end{aligned} \tag{5-1}$$

Modified Quadtrees (MTREEs)

A MTREE is a modified quadtree data structure, in which certain nodes contain GBs that indicate whether additional nodes exist at lower levels. These GBs reduce the time required to search for and retrieve data stored within the MTREE. This is similar to the binary buddy scheme implemented by Zhang and Ghose (2003) where "buddies" are small fixed structures at a high level that provide information about nodes in a binary tree at lower levels. The small GBs in the MTREE are called "look-ahead" GBs. Because the underlying structure of the MTREE is a quadtree, the number of rows, n_r, and columns, n_c, of the GB will always be equal:

$$n_r = n_c = 2^x \mid x = \text{number of levels to look down}$$

For example, if a look-ahead GB in the parent node at level 0 indicates the existence of nodes up to and including level 2, then $x = 2$, the size of the GB is 4x4, and the GB is known as a 2-level look-ahead GB (Figure 5-9). A set bit within the GB indicates that the corresponding node exists, i.e. is not null, at level 2. For example, in Figure 5-9, node 0 in level 2 exists (i.e., its corresponding bit is set), indicating that data exists at that location.

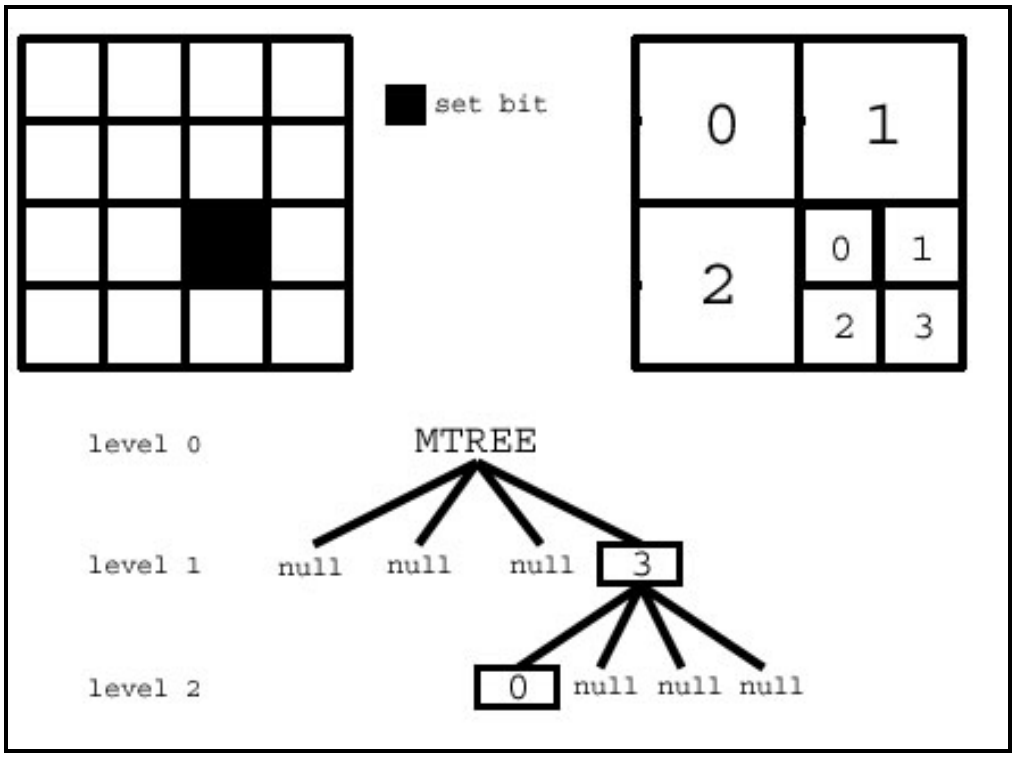


Figure 5-9. Look-Ahead GB

A 7-level MTREE might have 2-level look-ahead GBs on levels 0 and 2 and 4. In this example, each look-ahead GB forecasts the next two levels in the tree. Each time a node is created in the MTREE, the corresponding bit in each look-ahead GB is set. To further increase efficiency during data loading, lookup tables have been created that quickly give the bit row, b_r, and column, b_c, indices into the two-level look-ahead GB, indicating which bit to set, based on the node numbers:

$$b_r = \begin{bmatrix} 0011 \\ 0011 \\ 2233 \\ 2233 \end{bmatrix} \tag{5-2}$$

$$b_c = \begin{bmatrix} 0101 \\ 2323 \\ 0101 \\ 2323 \end{bmatrix} \tag{5-3}$$

To quickly determine which GB bit to set in the Figure 5-9 example, first the unique path in the MTREE to the data of interest is identified as node 3 (level 1) to node 0 (level 2). Next, these node numbers (3, 0) are used as the row and column indices into both lookup tables, from which

b_r and b_c are determined to be (2, 2). This corresponds to the bit index (2, 2) in the GB, which must be set to indicate that these nodes (3 in level 1 and 0 in level 2) exist.

Unlike a quadtree, in which a leaf node represents a single geographic location, the leaf node of an MTREE points to a list of all geographic locations that fall within the geographic area represented by the node. The number of degrees of LAT/LON that a node at a given level of the MTREE will represent is determined by:

$$\Delta Lat = \Delta Lon = \frac{1}{2^{level}},$$

In the previous example, a node at level 0 represents an area that is 1 degree of LAT x 1 degree of LON; a node at level 1 represents 0.5 degrees LAT x 0.5 degrees LON, etc. A lookup table replaces the above calculation, for computational efficiency:

$$\Delta Lat = \Delta Lon = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.25 \\ 0.125 \\ 0.0625 \end{bmatrix} \qquad (5\text{-}4)$$

To satisfy the precision requirements of the CAS algorithm (about 11 cm / point for MIW applications), a simple quadtree would require more than 20 levels, which results in a memory prohibitive data structure. The MTREE uses the concepts of a quadtree to decrease search time, and requires fewer levels to retain the desired precision of the data set. Since each leaf node of the MTREE points to a list of one or more points contained in the geographic area represented by the node, an increase in the number of levels in the tree results in a decreased average list size. Note that if each list contains at most a single point, the tree is a simple quadtree. Therefore, there is a trade-off between the number of levels chosen for the MTREE and the time required to search for a location of interest (i.e., the algorithm searches through more levels with shorter lists, or vice versa).

Data Loading

The MCDB contains features of interest that were detected on or close to the seafloor in SSI during surveys worldwide. NAVOCEANO extracts a subset of this historical data for roughly the same geographical area as a recent survey (i.e., the area of interest, AOI) when performing change detection. The CAS algorithm loads the center LAT/LON position of each historical feature and related attributes within the AOI by first creating a 2D memory pointer array of 180 rows by 360 columns. Each pointer stores the address of a MTREE containing feature data for a 1-degree LAT/LON cell. A MTREE representing a given 1-degree cell is not actually created until a feature located at a LAT/LON position within the cell is loaded. The memory pointer in the 2D array, which addresses the MTREE, will have a value of null until the MTREE is actually created.

The central LAT/LON location of each historical feature within an AOI is examined to determine in which 1-degree cell it resides. The LAT values are shifted from a range of {-90, …, 90} to {0, …, 180}by truncating the LAT, reducing it by 1 if the result is less than 0, and adding 90. The LON values are shifted from a range of {-180, …, 180} to {0, …, 360} in a similar manner. The resulting LAT/LON or (Y_S, X_W) value represents the lower-left (southwest) corner of the 1-degree cell and is used as a direct index into the 2D pointer array.

A MTREE is created if the corresponding pointer is null. The null value is replaced with the address of the root node of the MTREE. Next, all nodes in the path between the root node and each leaf node are created. At any given level, the four conditional checks (5-1) are made to determine which node must be created. The resolution of any given node (Δlat and Δlon) is found quickly by using a lookup table (5-4). The last level (i.e., the level of the leaf nodes) is where the features and the feature attributes are stored.

Certain nodes in the path will contain look-ahead GBs, as discussed in a previous section. If the look-ahead GBs do not already exist, they are created at this point in the process. Bits that represent the nodes created in the path are set. The bit row and column corresponding to each node is found by using the lookup tables, 5-2 and 5-3, respectively.

Searching for Spatially Close Features

The search algorithm within CAS, based on the central LAT/LON location of a mine-like contact (MILCO) from a recent survey, must find all spatially close MILCOs in the historical MCDB, while accounting for position error within and between the two data sets. The results from the search are then sent to the next stage of the ACDC process, which performs the FM to determine if any of the historical features are the same as the recent MILCO. If a match is not found, the recent feature is marked as a new MILCO.

The search algorithm manages this efficiently by using position error estimates, logical GB operations, and a two-step process. The first step (Figure 5-10) attempts to find all historical MILCOs within a region of uncertainty generated around each new MILCO. If this fails to produce a match during the FM stage, the search algorithm then performs the second step (Figure 5-11), which returns all historical MILCOs with an uncertainty region that overlaps the uncertainty region of the new MILCO. The regions of uncertainty are based on errors estimated during the survey, such as navigation and sensor errors.

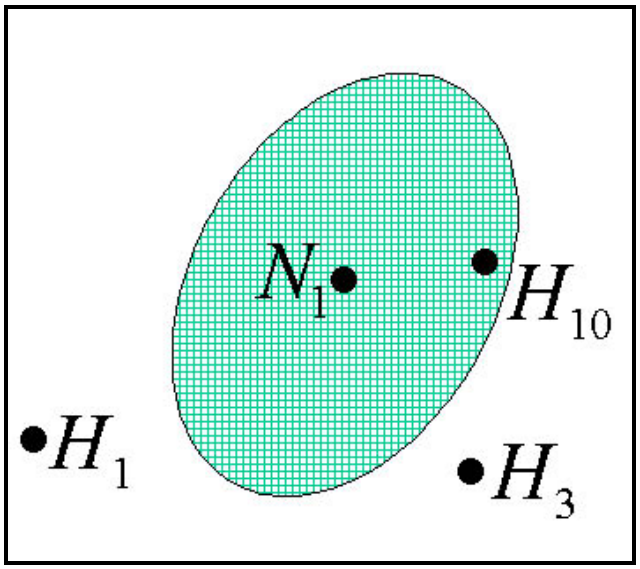


Figure 5-10. Step 1 searching returns the location of one historical MILCO, H_{10}, because it falls within the uncertainty error region of the recently detected MILCO, N_1.

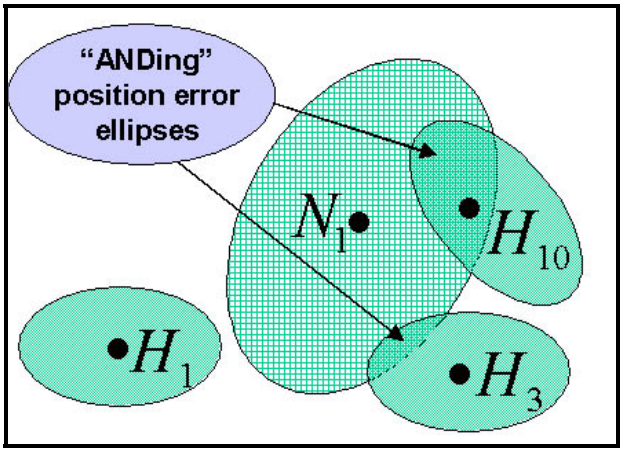


Figure 5-11. If the FM process did not determine that H_{10} was the same as N_1, then step 2 searching is performed, determining that both H_3 and H_{10} are spatially close to N_1 because their error regions overlap the error region of N_1.

The shape of the error region can be complex or as simple as a circle or ellipse. For step 1, all the leaf nodes that fall within the error region are determined (Figure 5-12) using a GB, called the error GB. Starting with the root node, the minimum-bounding rectangle (MBR) of the error region is determined. The size of the error GB is the minimum of the side of the MBR and the size of the look-ahead GB. The resolution of the error GB is equal to that of the look-ahead GB. Next, bits in the error GB that fall inside the error region are set (e.g., in Figure 5-12, this GB would have 4 rows and 5 columns with every bit set). This error GB is then logically ANDed with the current look-ahead GB. The resulting GB is traversed one bit at a time; each set bit represents a node that exists. If the node is a leaf, the list of points within the node will be sent to

the FM algorithm. If the node is not a leaf, it will contain another look-ahead GB, and the algorithm will repeat, using the current look-ahead GB in place of the previous.

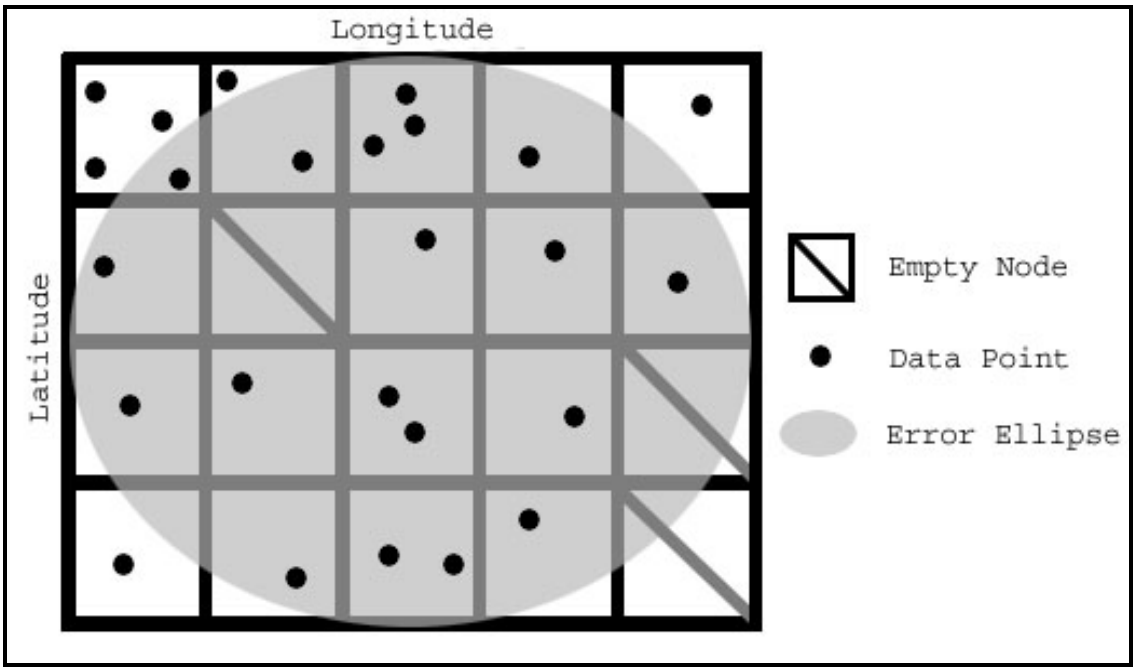


Figure 5-12. All MTREE leaf nodes falling within error ellipse for the feature of interest.

For step 2, the algorithm repeats step 1, but with an enlarged error region. The increased size should be sufficient to find all points within the historical database that need to be checked to determine whether their error regions overlap with the new MILCO's error region. Rather than sending all of the resulting leaf nodes' lists of points to the FM algorithm, as in step 1, step 2 checks whether the historical error region of each point overlaps the pre-expanded error region of the new MILCO, via GB logical operations. If so, the point is sent to the FM algorithm. If not, it is ignored.

Experimental Runs

Algorithms can be evaluated according to many different criteria, but a good measure of performance is the rate of growth in time required to solve larger and larger instances of the same problem. The time needed by an algorithm expressed as a function of the size of a problem is called the time complexity of the algorithm (Aho et al., 1974).

Methods

The processing time required for the CAS algorithm using 7 level MTREEs with two level look-ahead GBs on levels 0, 2, and 4 to perform LAT/LON queries and return all spatially close data points within a region of error was recorded for four randomly generated data sets of 600, 1500, 3000, and 10000 points. The resulting times were compared to the measured processing time from another search algorithm operating on the same data points. This search algorithm used a 2D array to store the data points and a method to calculate the great-circle distance between two

LAT/LON points (Xue-Lian, 1985). For clarity, this paper will refer to this method as the Great Circle Distance Method (GCDM).

In MIW applications, as described previously, the CAS algorithm is used to find points in a historical database (e.g., the MCDB) that are spatially close to points in a newly surveyed area (e.g., new MILCOs). For the purposes of these tests, the set of historical points (set "H") and the set of new MILCOs (set "N") are the same. For convenience, circular error regions were centered about these points, with radii of 5 m, 15 m and 25 m. Both algorithms loaded all points into their respective data structures (MTREEs for CAS and a 2D array for GCDM). For each point in set "N", the algorithms returned all the LAT/LON points in set "H" that fell within the region of error.

Both algorithms were run on a Pentium III 800 MHz PC with 256 MB of RAM running Linux. Each test was repeated five times, and the processing times were recorded and averaged.

Results

For the three smaller data sets, the time to load the points into the two data structures was nearly equivalent for the CAS and the GCDM. The CAS algorithm was slightly slower than the GCDM in loading the 10,000-point data set. Tables 1 – 3 list times to search for geospatially close features, given 5 m, 15 m, and 25 m error regions, respectively.

Table 5-1. Results from 5 meter error region

# of points	Avg. time CAS	Avg. time GCDM
600	8.55s	9.14s
1500	13.21s	38.87s
3000	55.63s	152.34s
10000	705.73s	1261.40s

Table 5-2. Results from 15 meter error region

# of points	Avg. time CAS	Avg. time GCDM
600	8.63s	9.12s
1500	13.87s	39.01s
3000	56.02s	153.41s
10000	711.49s	1268.29s

Table 5-3. Results from 25 meter error region

# of points	Avg. time CAS	Avg. time GCDM
600	8.71s	9.14s
1500	13.93s	39.01s
3000	56.21s	151.98s
10000	718.00s	1257.83s

As shown in Figure 5-13, the CAS algorithm performed better for all test cases than the GCDM. The radius of the error region had no significant effect on time for either algorithm. As the number of points increased, the difference in processing time between the two algorithms became more pronounced (e.g., GCDM ran only 1.05x slower than CAS with 600 points, but GCDM ran 1.75x slower than CAS for 10,000 points). This implies that the CAS algorithm is a better choice than GCDM for geospatial search applications, particularly on very large data sets.

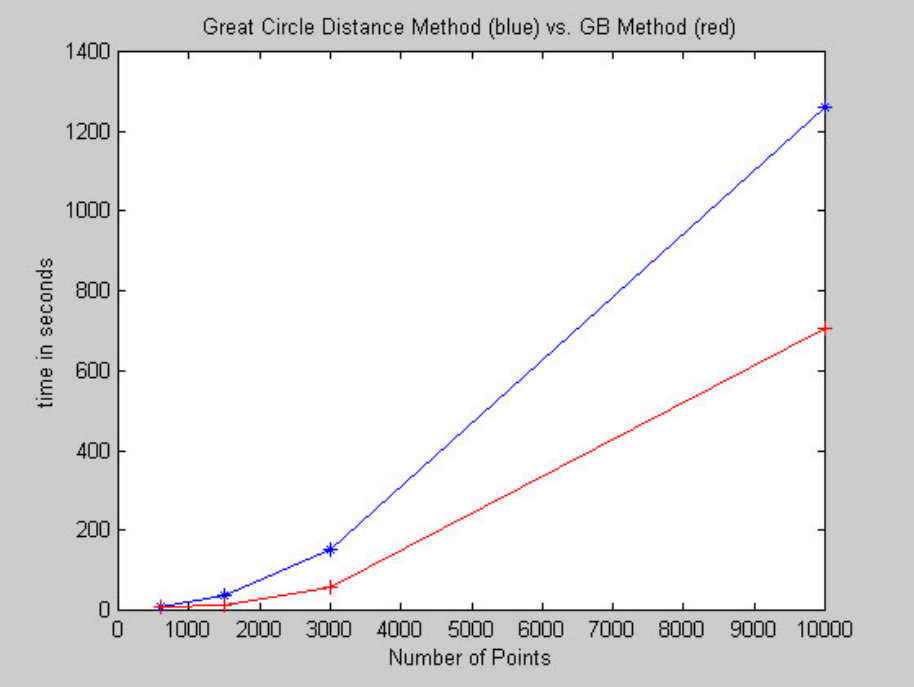


Figure 5-13. Average time (in seconds) that each method took to spatially search the data sets. Results were averaged for error radii of 5m, 15m and 25m.

Conclusions

SSI analysts need algorithms to help them perform change detection in a timely manner. Change detection allows MIW commanders to determine where assault lanes should be geographically placed and to estimate the time and resources needed to clear the lanes of explosive mines. The data collected by towed SSS during surveys cannot be accurately geo-registered due to position errors. The main contribution to this error is from the inability of the cable layback model to estimate the geographical position of the towfish accurately. The time it takes analysts to perform change detection increases because analysts must load and review large amounts of historical imagery geo-registered at and near an estimated position of a feature of interest.

This paper shows that CAS performs well on large data sets because of its use of the author's MTREEs and look-ahead GBs. Because GBs are also used to represent search regions based on an estimate of position error, the regions can be any 2D shape. Tests in this chapter show that CAS was able to search using a curricular error region with a radius of 25 m and a large data set of 10,000 random points almost twice as fast as an algorithm that uses a 2D array and a great circle calculations for the error search radius. MIW analysts at NAVOCEANO are currently

using and evaluating the author's CAS algorithm during the Gulf of Mexico (GOMEX) exercises (December 2004). Initial results are positive, and the analysts are successfully able to use the algorithm to accurately perform geospatial searching of the historical database and perform change detection more rapidly.

Acknowledgements

This work was performed in support of the Naval Oceanographic Office (NAVOCEANO) and funded by the Space and Naval Warfare Systems Command (SPAWAR) through the Commander, Naval Meteorology and Oceanography Command (CNMOC) and ONR under Program Element 0603207N. The authors thank Dr. Edward Mozley (SPAWAR), Dr. Bruce Northridge (CNMOC) and Mr. Michael M. Harris (NRL) for their support of this project. We thank Mr. Jim Hammack (NAVOOCEANO) for his recommendations concerning the application of this algorithm in support of NAVOCEANO requirements. Any mention of commercial products or the use of company names does not imply endorsement by the U.S. Navy. This paper is approved for public release; distribution is unlimited.

References

Aho, A.V., J.E. Hopcroft, and J.D. Ullman (1974). *The Design and Analysis of Computer Algorithms (Addison-Wesley Series in Computer Science and Information Processing.* Boston: Addison Wesley.

Chon, H.D., Agrawal, D. and Abbadi, A.E. (2002). Data Management for Moving Objects. *IEEE Data Engineering Bulletin*, 25:2.

Deitel, H.M. and P.J. Deitel (1994). *How to Program C*. New Jersey: Prentice-Hall.

Feder, H.J.S., C.M. Smith, and J.J. Leonard (1998). Incorporating Environmental Measurements in Navigation. *IEEE AUV*, Cambridge, MA, August.

Gendron, M.L., P. B. Wischow, M.E. Trenchard, M.C. Lohrenz, L.M. Riedlinger, M. Mehaffey (2001). *Moving Map Composer*. Naval Research Laboratory, US Patent No. 6,218,965.

Guttman, A. (1984). R-trees: A Dynamic Index Structure for Spatial Searching. Proceedings of *ACM Special Interest Group on Data Management*, June.

Howarth P.J. and G. M. Wickware (1981). Procedures for Change Detection Using Landsat Digital Data. *Int. J. of Remote Sensing,* 2.

Jackson, M. (1978). The Jackson Design Methodology. *Infotech State of the Art Report*, Structured Programming.

Joannides, M. and F. Le Gland (1993). Position Estimation of a Towed Underwater Body. Proceedings of the *32nd Conference on Decision and Control*, San Antonio, Texas, December.

Kamgar-Parsi, B. and B. Kamgar-Parsi (1997). Registration Algorithms for Geophysical Maps. Proceedings of *IEEE Oceans' 97 Conference*, October.

Kamgar-Parsi, B., J.L. Jones and A. Rosenfeld (1991). Registration of Multiple Overlapping Range Images: Scenes without Distinctive Features. *IEEE Transactions on Pattern Analysis and Machine intelligence,* 13:9, September.

Kardos, J., A. Moore, and G. Benwell (2003). Visualizing Uncertainty in Geographic Data Using Hierarchical Spatial Data Structures. Processings of the *7th International Conference on GeoComutation*, University of Southampton, United Kingdom, 8-10 September.

Kenny, A.J., B.J. Todd, and R. Cooke (2001). *Procedual Guidelin No. 1-4 – The Application of Sidescan Sonar for Seabed Habitat Mapping, Marine Monitoring Handbook*. Peterborough: The Joint Nature Conservation Committee, Peterborough.

King, J.E. and R.I. Egbert (2003). *The GPS Handbook*. New York: Burford Books.

Klein Associates, Incorporated (2004). Klein Associates, Incorporated. Website, http://www.kleinsonar.com.

Lingsch, W.C. and S.C. Lingsch (2001). Sonar Image Change Detection as a Mine Counter Measures Tactical Tool. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

Mas, J.F. (1999). Monitoring Land-cover Changes: A Comparison of Change Detection Techniques. *Int. J. Remote Sensing*, 20.

Neill, R. (1999). Non-Acoustic Tracking of Towfish Location. Proceedings of the *Shallow Survey 1999 Conference*, Defence Science and Technology Organisation (DSTO), Australia.

Rosin, P.L., (1994). Parcel-Based Change Detection. *European Symposium on Satellite Remote Sensing*, Institute for Remote Sensing Applications Joint Research Centre, Ispra, Italy.

Roussopoulos, N., S. Kelley and F. Vincent (1995). Nearest Neighbor Queries. Proceedings of the *ACM Special Interest Group on Data Management International Conference on the Management of Data*, San Jose, California.

Schwab, W.C., Webb, R.M.T., Danforth, W.W., O'Brien, T.F., and Irwin, B.J. (1991). High-Resolution Sidescan-Sonar Imagery of the Manchas Interiores - Manchas Exteriores Coral Reef Complex, Mayagüez, Puerto Rico. *U.S. Geological Survey Open-File Report*.

Singh A. (1989). Digital Change Detection Techniques Using Remotely-Sensed Data. *Int. J. of Remote Sensing*, 10.

Tobler, W. and Z. Chen (1986). A Quadtree for Global Information Storage. *Geographical Analysis*, 18:4.

Xue-Lian, Z. (1985). The Nested Coefficient Method For Accurate Solutions of Direct and Inverse Geodetic Problems With Any Length. Proceedings of the *7th International Symposium on Geodetic Computations*, Cracow, June 18-21.

Zhang Z. and K. Ghose (2003). yFS: A Journaling File System Design for Handling Large Data Sets with Reduced Seeking. *2nd USENIX Conference on File and Storage Technologies*.

Chapter 6 – Conclusion

Marlin L. Gendron, Maura C. Lohrenz, and Geary J. Layne
Naval Research Laboratory, (NRL) Code 7440.1, Stennis Space Center, MS

Juliette Ioup
University of New Orleans (UNO), New Orleans, LA

Acronyms

ACDC	automated change detection and classification
AM	area matching
AOI	area of interest
CAC	computer-aided classification
CAD	computer-aided detection
CAI	computer-aided identification
CAS	computer-aided search
FM	feature matching
GB	geospatial bitmap
GIS	geographic information system
GPS	Global Positioning System
GUI	graphical user interface
LAT/LON	latitude/longitude
LCM	Layne Completion Method
MCDB	Master Contact Database
MILCO	mine-like contact
MILEC	mine-like echoes
MIW	Mine Warfare
MRF	Markov Random Field
MU	mini-UNISIPS
NAVOCEANO	Naval Oceanographic Office
NON-MILCO	non-mine-like contact
NRL	Naval Research Laboratory
QC	quality control
SSI	sidescan sonar imagery
SSS	sidescan sonar system
TDA	Tactical Decision Aid
UNISIPS	Unified Sonar Image Processing System
UNO	University of New Orleans

Introduction

In the book, *Software Engineering with Ada*, Grady Booch (1983), one of the leading pioneers in software engineering, states that developing and completing an efficient, reliable and understandable software system can be a Herculean task, especially when the system is large and operates in real-time. Booch also says that the creators of such systems must act as part scientists, part artists, and must develop smaller individual software algorithms separately to realize the larger system.

B.J. MacLennan (1987), from the Naval Postgraduate School, agrees with Booch when he states that much of the difficulty of programming large systems stems from the complexity of dealing with many different details at one time. A sound engineering approach is to break the whole system down into smaller manageable parts and develop algorithms and data structures to solve the parts separately.

The goal of this dissertation is to divide the complex task of automating sidescan sonar imagery (SSI) change detection into smaller, more easily solvable problems and to present selected algorithms and data structures that the author developed to solve these problems. The algorithms and data structures presented do not represent the total solution to SSI digital change detection, but are key to the development and implementation of an Automated Change Detection and Classification (ACDC) system. The research and development needed to fully implement ACDC will continue over the next few years. When completed, the ACDC system will be capable of aiding SSI analysts to detect seafloor features in imagery collected by a sidescan sonar system (SSS), classify, and catalog said features; then compare them with ones "seen" in the past (stored in historical databases) to determine whether the features have moved or are new.

As stated above, the main purpose of ACDC is to aid the analysts, so therefore ACDC must be "user-friendly" and robust. The components of the ACDC system must work together seamlessly inside a graphical user interface (GUI) that functions as a geographical information system (GIS) to allow analysts to perform change detection on SSI faster, more accurately and efficiently. ACDC will also enable analysts to display results during all phases of the change detection process, combine the results with other useful geospatial data, and manipulate the results with GIS tools. The final fused graphics and supporting information will then be passed to another computer system, called a Tactical Decision Aid (TDA), that will in turn aid naval commanders to make critical decisions rapidly during mine clearing operations.

The next section of this chapter goes through flow diagrams that describe the processing flow for ACDC one stage at a time and reiterate components that: 1) have been completed and discussed in the main chapters of this dissertation, 2) have been completed but not discussed, and 3) are yet to be completed, i.e. future work. An illustration of the entire processing flow is shown in Figure 1-9 of the introductory chapter of this dissertation.

The ACDC System

During military Mine Warfare (MIW) operations, assault lanes are cleared of mines to enable naval ships to pass safely. In support of MIW, analysts perform change detection by visually comparing historical high resolution imagery collected by a SSS with recently collected SSI in an attempt to identify objects (which might be explosive mines) placed at sea since the last time the area was surveyed. The goal of change detection is to match newly detected objects with previously detected objects, which naval divers or autonomous machines have visually investigated and verified. Change detection using SSI is a potentially significant time-saving tool, but problems such as large data volume (Reed, 2001; Cronin et al., 2003), navigational errors, sonar tow-body instabilities, differences in look-angle, and differences in environmental conditions can hinder these time savings (Lingsch and Lingsch, 2001).

The key components of ACDC are detection, classification, search, identification, and display. A sixth component is clustering, which can be performed after the detection, classification, and/or identification stages. The following paragraphs detail each of these stages.

Detection Stage

Detection is the first stage of ACDC, during which SSI, stored in the Unified Sonar Image Processing System (UNISIPS) format, is analyzed in an attempt to detect all so-called mine-like echoes (MILECs). MILECs appear in the imagery as bright spots with accompanying shadows adjacent to the bright spot and away from nadir. Typical errors during the detection stage are false detections (i.e., false echoes) and misses (i.e., missed MILEC detections). Because it is so important that no mines are missed during change detection, the detection "parameters" that control the "sensitivity" of the detector are relaxed. This causes the detector to produce more false echoes (which can be filtered in the next stage), but increases the probability that echoes will not be missed.

The detected echoes, both MILECs and false echoes, are "snipped" out of the SSI into small image files, called snippets, and stored in the MU (mini-UNISIPS) file format. The latitude/longitude (LAT/LON) location of the center of the echo is calculated and stored in the MU along with all information present in the source UNISIPS file (e.g., the distance of the object from nadir, the speed of the sonar, etc.). Figure 6-1 shows a block diagram of the detection stage. Note that analysts can choose to use a computer-aided detection (CAD) algorithm developed by the author and presented in Chapter 4, or can perform detection manually (e.g., if the SSI is particularly noisy).

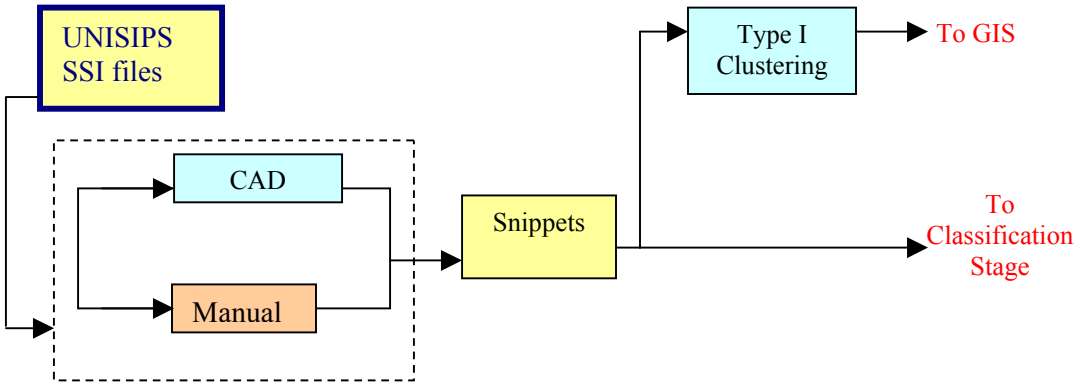


Figure 6-1. Detection Stage

An analyst, with or without the aid of a computer-aided classification (CAC) algorithm, sorts and classifies the snippets in the next stage of ACDC. Areas with an abundance of echoes, called areas of high "clutter", pose a problem because they can take considerably more time to analyze. If identified before the classification stage, areas of high clutter can be avoided altogether, and naval planners can choose geographical areas of low clutter, which take less time to investigate and clear of mines. Therefore, SSI analysts need a software algorithm, presented in Chapter 3, which will allow them to quickly cluster echoes geographically into polygons and estimate a clutter density for the resulting polygons. The output of the clustering, called Type I Clustering, is sent to the ACDC GIS display, where the polygons and densities are presented to the analyst.

Classification Stage

Two types of snippets result from the detection phase: false echoes and MILECs. The purpose of the classification stage is to eliminate the false echoes and classify the MILECs as either mine-like contacts (MILCOs) or non-mine-like contacts (NON-MILCOs). This can be a time-consuming task for analysts to perform. The co-author of this chapter, Layne, developed an algorithm called the Layne Completion Method (LCM) that adaptively filters each snippet produced by the detection stage into an image with only three intensity values: black (for shadows), white (for brights), and grey for everything else. This is similar to the image processing technique that utilizes the Markov Random Field (MRF) model to segment the entire gray-scale SSI into bright regions ("brights"), dark regions ("shadows"), and reverberation (Collet et al., 1996; Mignotte et al., 2000a; Mignotte et al., 2000b; Murino, 2001; Reed et al., 2001; Reed et al., 2002; Reed et al., 2003), but LCM is unique in its approach and relies on geospatial bitmaps (GBs) for efficiency of operation.

A threshold is adaptively determined to identify all bright pixels from the echo snippet and replace their intensity values with pure white (i.e., an intensity value of 255). Another threshold is found, during the same processes, to identify all shadow pixels and replace their intensity values with pure black (i.e., an intensity value of 0). All remaining pixels are set to a neutral intensity (i.e., an intensity value of 128). The resulting image, called a filtered echo image, is a three-intensity representation of the echo. Figure 6-2 shows four sample echo snippets and their corresponding filter echo images.

Research suggests that shadows are easier to extract from SSI than bright spots, and shadows better represent an object (Mignotte et al., 2000a, , Mignotte et al. 2000b; Reed et al., 2002; Reed et al., 2003). The LCM extracts the shadow from the filtered echo image and applies a process to "clean" the shadow. The cleaned shadow is then used to "complete" the bright spot based on the dimensions of the shadow, the look-angle of the SSS, and the distance the echo is from nadir. The completed bright spot and cleaned shadow are then automatically measured to estimate the size of the echo in all three dimensions. Echoes with estimated dimensions consistent with known mines are flagged as MILCOs and all others are thought to be non-MILCOs.

Figure 6-3 shows a flow diagram of the classification stage. An analyst can manually classify the snippets from the detection stage, or allow the LCM algorithm to suggest a classification. The LCM is not fully automated and calculates a measure of confidence for every completion. The analyst makes "the final call" when this measurement falls below a set threshold, e.g. 50% confidence. The center LAT/LON locations of the MILCOs can also be clustered, called Type II Clustering. At the end of the detection stage, clustering of clutter into polygons aids in the determination of which areas should be avoided altogether. The clustering of MILCOs into polygons with densities helps to estimate the amount of time it will actually take to clear a lane.

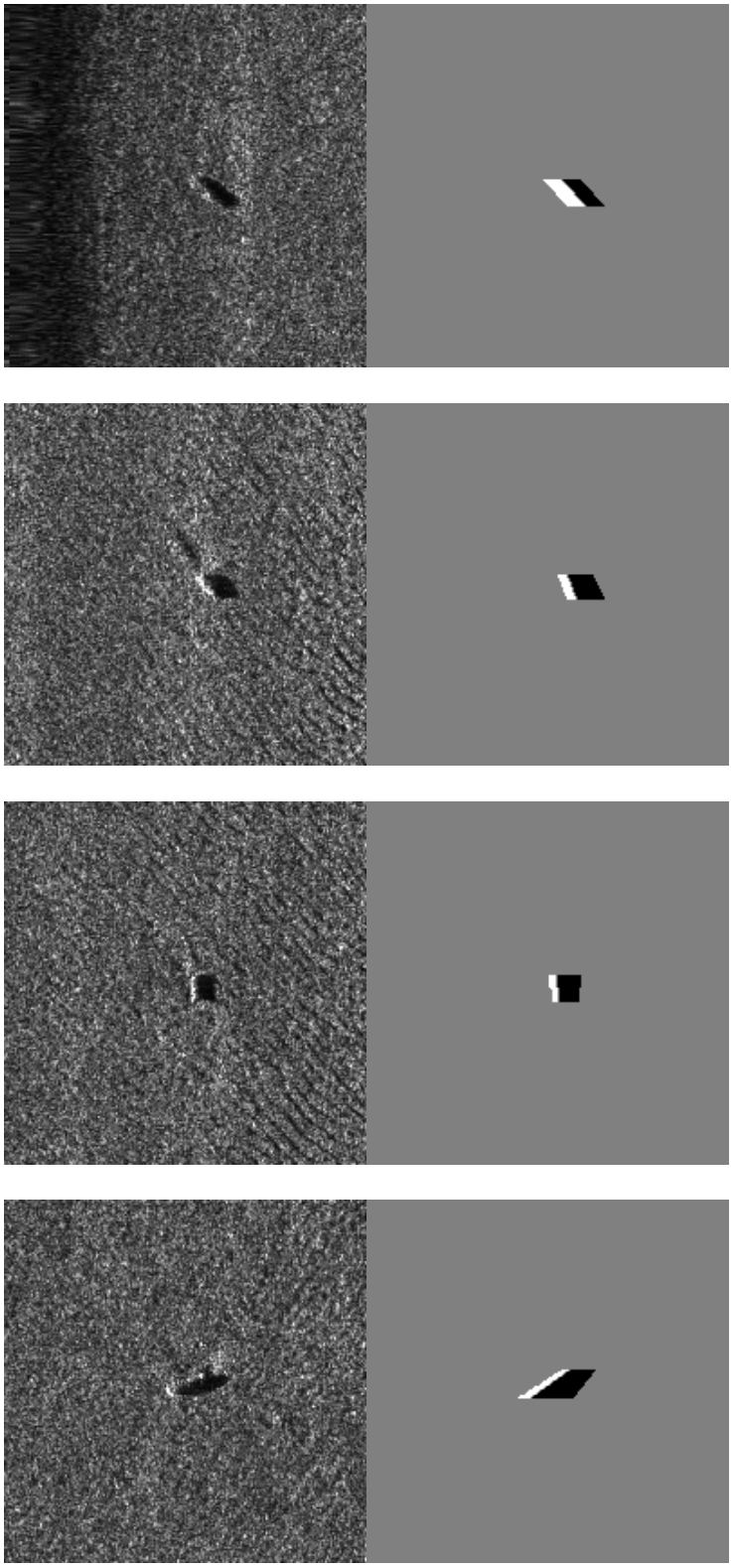


Figure 6-2. Sample echo snippets (left) along with their filtered echo images (right).

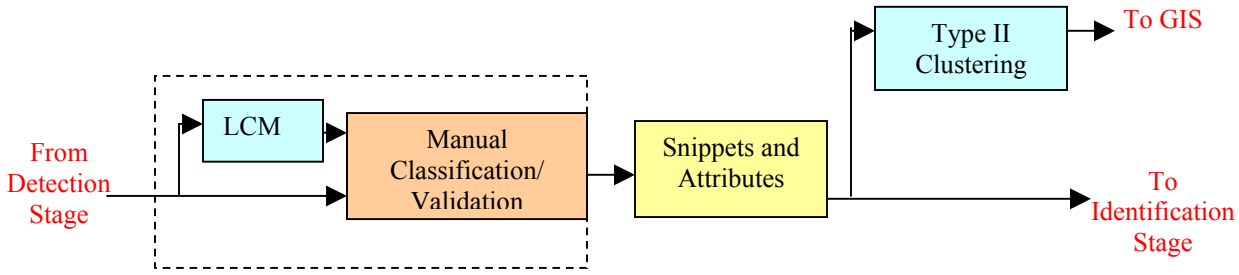


Figure 6-3. Classification Stage

Search Stage

In the early 1990's, the Naval Oceanographic Office (NAVOCEANO) began collecting and producing SSI, and analysts started cataloging features of interest worldwide. NAVOCEANO created a Master Contact Database (MCDB) to store these features as snippets. The MCDB also contains feature attributes, including the estimated LAT/LON of feature positions, dimensions, and detailed descriptions.

The LAT/LON position of a feature on any given scan line can be estimated using a great circle calculation, given the following information: 1) the LAT/LON position of nadir (i.e., the position of the towfish), 2) the heading at nadir (i.e., the direction the towfish is traveling), and 3) the distance from nadir to the feature. The accuracy of the towfish location (1, above) has the greatest impact on the accuracy of the feature position (Schwab et al., 1991; Neill, 1999).

The position of the towfish is often determined by measurements made by a Global Positioning System (GPS) receiver located aboard the towing vessel, along with an estimation of the length of the tow cable using a cable layback model. GPS and cable layback errors affect the accuracy of the estimated location of both new and historical features.

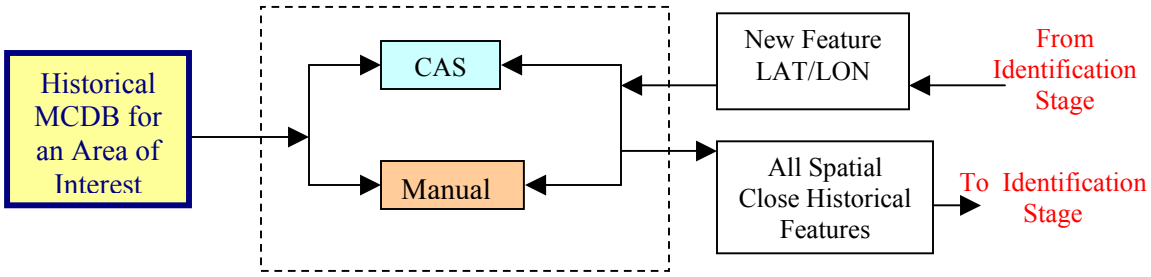


Figure 6-4. Search stage of ACDC

Chapter 5 of this dissertation presents a computer-aided search (CAS) algorithm developed by the author that is capable of spatially searching a historical database where large position errors exist. Figure 6-4 shows the search stage of ACDC. In this case, NAVOCEANO provides the historical imagery and feature snippets (with attributes) for the geographic area of interest (AOI) where change detection is to be performed. The identification stage, described below, requires

all the spatially close historical features, within an error region, to attempt to match a new feature with one observed in the past. An analyst can manually search for these historical features, or can choose to have the automated CAS algorithm perform the search to save time.

Identification Stage

The actual change detection is performed in the next stage of ACDC, called identification. Analysts attempt to "match" new features with ones observed in the past (stored in the MCDB). The matching is done manually, or with the aid of two algorithms that together perform computer-aided identification (CAI). The first algorithm performs feature matching (FM) by attempting to match a newly classified MILCO (from the classification stage) with one of the historical MILCOs found during the search stage. The second algorithm performs area-matching (AM) operations that attempt to match clusters of new features in close spatial proximity with clusters of the same features in the historical database.

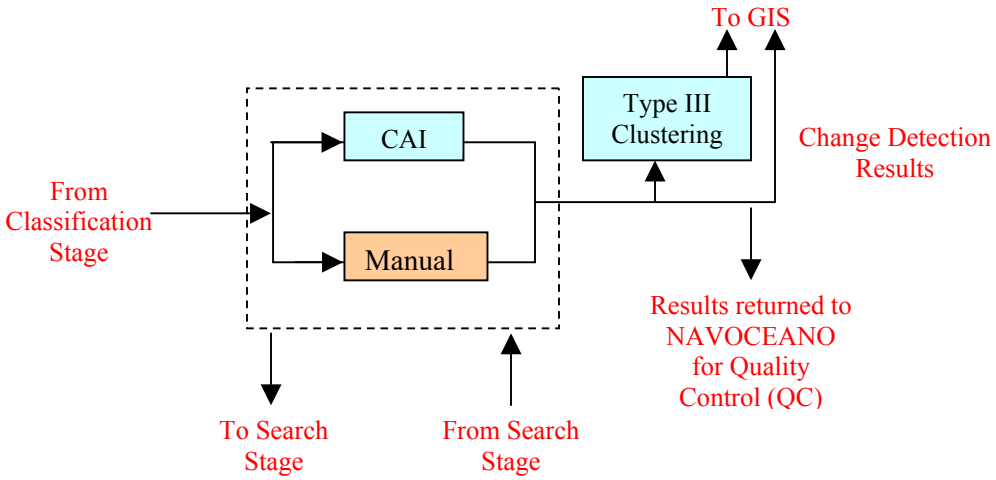


Figure 6-5. Identification Stage of ACDC

Figure 6-5 illustrates the identification stage of ACDC. The LAT/LON location of each feature from the classification stage is sent to the search stage one at a time. The search stage returns spatially close feature(s). Matching is performed to determine if the new feature has been seen in the past or is a new object (change detection). The results are sent to the GIS display. The center LAT/LON positions of features not observed in the past, i.e. not in the MCDB, can be clustered in a process called type III clustering which uses the algorithm developed by the author and discussed in Chapter 3. The resulting geographical polygons can be displayed in the GIS to help analysts estimate the resources and time needed to investigate new features that could be mines.

The results from the change detection process are sent to the GIS, but are also sent back to NAVOCEANO for quality control (QC). Expert analysts review the results and update the MCDB accordingly. The process of deconfliction is performed, in which analysts ensure that the same feature detected on different sonar passes is not stored in the database as multiple features.

Graphic Information System (GIS) Display

The ACDC system can be thought of as a GIS. GIS systems facilitate the storage, manipulation, and analysis of geographic data, and a GUI enables GIS users to input commands and output results (Rhind and Connolly, 1992; Worboys, 1995). Analysis of geographical data within a GIS is performed using a common map projection, such as Mercator, a common datum, such as WGS-84, and independently of map scale (Walker et al., 1996).

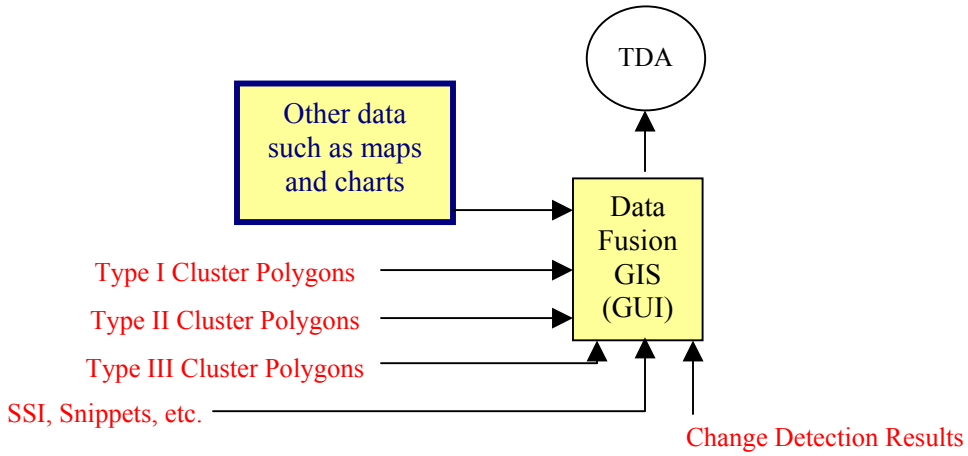


Figure 6-6. All components of ACDC are tied together in GIS.

Figure 6-6 illustrates how SSI, snippets, cluster polygons (types 1, 2 and 3), results from change detection, background maps and auxiliary information are all displayed and can be fused together by GIS tools for input into a TDA.

Future Work

The author's algorithms and data structures presented in the main chapters of this dissertation do not represent all the components required for the ACDC system. Additional algorithms are needed to assist analysts in identifying and matching new features with spatially close, historical features. Reliable identification of features in SSI by analysts is difficult because imagery contains little three-dimensional feature information, and during most SSS surveys, only one pass is made over any given feature (Mayer et al., 2003).

At least half of the human cerebral cortex is dedicated to visual processing. Implementing computer algorithms to perform comparably or to aid the human visual system in a meaningful way is extremely difficult (Tsotsos, 2003). For the past two years, the author has been researching and developing FM and AM algorithms to aid analysts in accurately identifying and matching features in SSI.

Wavelets and neural networks are excellent at recognizing features, particularly faces (Krueger and Sommer, 2000). In support of the FM component of the identification stage of ACDC, the University of New Orleans (UNO) and the author are researching new methods to develop wavelet networks that can be trained to recognize basic shapes similar to extracted shadows and

brights produced by the LCM (Ioup et al., 2003). Research is ongoing to try to train specialized neural networks, called self-organizing maps, to classify and identify echo snippets and their corresponding filter echo images (Ioup et al., 2004), and to classify SSI snippets using a combination of Kalman filtering techniques and neural networks (McDowell et al., 2003).

Research into the AM component of identification stage is planned to begin in mid-2005. The authors plan to continue this research into FM and AM techniques and complete the GIS and GUI components and transition a fully-functional ACDC system in the next two years.

References

Booch, G. (1983). *Software Engineering with Ada*. Menlo Park: Benjamin/Coummings Publishing Company.

Collet, C., P. Thourel, P. Perez, and P. Bouthemy (1996). Hierarchical MRF Modeling for Sonar Picture Segmentation. Proceedings of the *3rd IEEE International Conference on Image Processing,* 3, Lausanne, Switzerland.

Cronin, D., M. Broadus, B. Reed, S. Byrne, W. Simmons, and L. Gee (2003). Hydrographic Work Flow – From Planning to Products. Proceedings of the *U.S. Hydro 2003 Conference*, March 24-27, Biloxi, Mississippi.

Ioup, J.W., M.L. Gendron, M.C. Lohrenz, G.J. Layne and G.E. Ioup (2004). Classifying Sidescan Sonar Images Using Self Organizing Maps. Abstract of the *J. Acoust. Soc. Am*, 116.

Ioup, J.W., M.L. Gendron, P.J. McDowell, B.S. Bourgeois and G.E. Ioup (2003). Wavelets and Neural Networks for Change Detection in Simulated Sidescan Sonar Data. Abstract of the *J. Acoust. Soc. Am*., 114.

Krueger, V. and G. Sommer (2000). Affine Real-Time Face Tracking using Gabor Wavelet Networks. *Internation Conference on Pattern Recognition*, Barcelona, Spain, September, 3-8.

Lingsch, W.C. and S.C. Lingsch (2001). Sonar Image Change Detection as a Mine Counter Measures Tactical Tool. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

MacLennan, B. (1987). *Principles of Programming Languages*. 2nd Edition, Austin: Holt Rinehart and Winston, January.

Mayer, L.A., Calder, B., J.S. Schmidt and C. Malzone (2003). Providing the Third Dimension: High-resolution Sonar as a Tool for Archaeological Investigations – An Example from the D-Day. Proceedings of the *U.S. Hydro 2003 Conference*, March 24-27, Biloxi, Mississippi.

McDowell, P.J., M.L. Gendron, P.M. McDowell, J.W. Ioup and G.E. Ioup (2003). Kalman Filtering With Neural Networks for Change Detection in Simulated Sidescan Sonar Data. Abstract of the *J. Acoust. Soc. Am*., 114.

Mignotte, M., C. Collet, P. Perez and P. Bouthemy (2000a). Markov Random Field and Fuzzy Logic Modeling in Sonar Imagery: Application to the Classification of Underwater Floor. *Computer Vision and Image Understanding*, 79.

Mignotte, M., C. Collet, P. Perez and P. Bouthemy (2000b). Sonar Image Segmentation Using an Unsupervised Hierarchical MRF Model. *IEEE Transactions on Image Processing*, 9-7, July.

Murino, V. (2001). Reconstruction and Segmentation of Underwater Acoustic Images Combining Confidence Information in MRF models. *Pattern Recognition*, 34:5.

Neill, R. (1999). Non-Acoustic Tracking of Towfish Location. Proceedings of the *Shallow Survey 1999 Conference*, Defence Science and Technology Organisation (DSTO), Australia.

Reed, B. (2001). Data, Data Everywhere and Nary a Bit to Drop. Abstract for the *Shallow Survey 2001 Conference*, Sidney, Australia.

Reed, S., E. Dura, D.M.Lane, Y. Petillot, J. (2002). Extraction and Classification of Objects from Sidescan Sonar Bell. *IEEE Workshop on Nonlinear and Non-Gaussian Signal Processing*, 8-9 July.

Reed, S., Y. Petillot, and J. Bell (2001). Unsupervised Mine Detection and Analysis in Sidescan Sonar: A Comparison of Markov Random Fields and Statistical Snakes. Processings of *CAD/CAC 2001*, Halifax.

Reed, S., Y. Petillot, J. Bell (2003). An Automatic Approach to the Detection and Extraction of Mine Features in Sidescan Sonar. *IEEE Journal of Oceanic Engineering,* 28-1, January.

Rhind, D. and T. Connolly (1992). *Understanding GIS: The ARC/INFO Method*. Redlands: Environmental Systems Research Institute.

Tsotsos, J. (2003). Encyclopedia of Cognitive Science. *Computer Vision*.

Schwab, W.C., Webb, R.M.T., Danforth, W.W., O'Brien, T.F., and Irwin, B.J. (1991). *High-Resolution Sidescan-Sonar Imagery of the Manchas Interiores - Manchas Exteriores Coral Reef Complex, Mayagüez, Puerto Rico*. U.S. Geological Survey Open-File Report.

Walker, J.D., Black, R.A., Linn, J.K., Thomas, A.J., Wiseman, R., and D'Attilio, M.G. (1996). Development of Geographic Information Systems-Oriented Database for Integrated Geological and Geophysical Applications. *GSA Today: A Publication of the Geological Society of America* 6:3.

Worboys, M. (1995). *GIS: A Computing Perspective*. London: Taylor & Francis.

Appendix: Co-Author Permission Letters

December 9, 2004

Marlin L. Gendron
NRL Code 7440.1
Stennis Space Center, MS 39529

Dear Mr. Gendron,

As a co-author on six chapters comprising your dissertation, I have read the final drafts of all chapters, and I authorize you to use the portions of the chapters to which I contributed. If you have any questions, I may be contacted at mlohrenz@nrlssc.navy.mil or (228) 688-4611.

Sincerely,

Maura C. Lohrenz

Maura C. Lohrenz

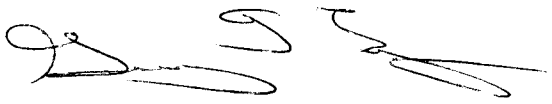
December 9, 2004

Marlin L. Gendron
NRL Code 7440.1
Stennis Space Center, MS 39529

Dear Mr. Gendron,

As a co-author on six chapters comprising your dissertation, I have read the final drafts of all chapters, and I authorize you to use the portions of the chapters to which I contributed. If you have any questions, I may be contacted at glayne@nrlssc.navy.mil or (228) 688-4693.

Sincerely,



Geary J. Layne

Vita

In 1986, Marlin Gendron graduated first in his class in the Industrial Electronics Technology program at the Mississippi Gulf Coast Junior College. After graduation, he worked as the Audiovisual Library Supervisor for an Air Force contractor located on Keesler Air Force Base (KAFB), Mississippi. Two years later, Mr. Gendron left KAFB and accepted a part-time audiovisual position at the University of Southern Mississippi (USM). While working at USM, he began pursuing an undergraduate degree in computer science. A year before his expected graduation, Marlin applied and received a cooperative education position at the Naval Research Laboratory (NRL) Mapping, Charting and Geodesy Branch, Marine Geosciences Division, at the Stennis Space Center, Mississippi.

Mr. Gendron graduated in 1991 with highest honors and received a Bachelors of Science degree in computer science with a minor in mathematics from USM. Upon graduation, he accepted a position at Planning Systems, Inc., as a computer scientist. In 1997, Mr. Gendron was hired by NRL. In 1999, Mr. Gendron received a Masters of Science in Applied Physics from the University of New Orleans (UNO), and shortly after graduation, he began pursuing a Ph.D in Engineering and Applied Science at UNO. Mr. Gendron is currently the principal investigator on a project to develop an Automated Change Detection and Classification (ACDC) system for sidescan imagery.