A Sonar-Based Mapping System for an Unmanned Undersea Vehicle

by

Margaret F. Nervegna
B.S. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2001

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September 2002

Author ________________________________
Department of Electrical Engineering and Computer Science
August 9, 2002

Certified by ________________________________
Michael J. Ricard
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by ________________________________
Leslie P. Kaelbling
Professor, Computer Science and Engineering
Associate Director, MIT Artificial Intelligence Laboratory
Thesis Advisor

Accepted by ________________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

[This Page Intentionally Left Blank]

A Sonar-based Mapping System for an Unmanned Undersea Vehicle

by

Margaret F. Nervegna

Submitted to the Department of Electrical Engineering and Computer Science
on August 9, 2002, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

An unmanned undersea vehicle (UUV) must operate autonomously in a complex, dynamic environment and react intelligently to changing tactical, environmental, and mission variables with no outside intervention. To support these real-time, adaptive mission capabilities, the building and updating of an efficient and accurate *map* of the tactical scene is critical. The challenges are to obtain useful and comprehensive information about the environment, to represent and fuse this data into an on-board map, to update the map in real-time when new data is discovered, and to save the map for future use while maintaining both efficiency and accuracy.

This thesis presents the design and implementation of a sonar-based mapping system for a UUV, and discusses the elements of the mapping system design: representation of static and dynamic obstacles in a mapping system, the need for efficient data structures, the incorporation of sonar measurement uncertainty, and the assimilation of new information into the map. The mapping system consists of a static obstacle map that stores information about stationary objects and a dynamic obstacle map that stores information about moving objects in the underwater environment. The static obstacle map consists of a local map that represents the immediate mission area and a global map that represents the entire mission area. The combination of the separate maps forms an integrated mapping system that represents the UUV's tactical scene, supports a query for the presence or absence of an obstacle at any location, time, and level of certainty, and as such, can be used to support the UUV's mission objectives.

This thesis also discusses modeling of noise in the sonar measurements. Since the mapping system must handle noisy sonar measurements, a model of a noisy sonar measurement is an imperative part of the sonar simulation and the validation of the mapping system.

Technical Supervisor:	Dr. Michael J. Ricard
Title:	Principal Member of the Technical Staff, The Charles Stark Draper Laboratory, Inc.
Thesis Advisor:	Dr. Leslie P. Kaelbling
Title:	Professor of Computer Science and Engineering, Associate Director, MIT Artificial Intelligence Laboratory

[This Page Intentionally Left Blank]

Acknowledgements

There are many to whom I owe a great deal of gratitude.

First, I want to thank Dr. Michael Ricard of Draper Laboratory for his dedication to and genuine concern for his students. I am very grateful for his patience, his guidance, and his confidence in my ability.

I also want to thank Professor Leslie Kaelbling for taking the time out of her already busy schedule to offer helpful suggestions for this thesis.

Additionally, I would like to thank my family, friends, and all those who have supported and encouraged me over the years. I thank my family for their continual support and encouragement, for teaching me what is important in life and learning, and for always believing in me. I thank my MIT friends, especially my roommate Julie, for keeping me sane and focused and for all the good times and the memories throughout my time at MIT. And I thank my running teammates and friends for adding balance to my life and for always giving me new goals for which to strive.

Finally, I would like to thank Draper Laboratory for the opportunity to continue my education and to perform this research.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc., under Internal Research and Development.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings herein. It is published for the exchange and stimulation of ideas.

Margaret F. Nervegna

Margaret F. Nervegna
August 9, 2002.

[This Page Intentionally Left Blank]

Table of Contents

List of Figures

[This Page Intentionally Left Blank]

1 Introduction

An autonomous Unmanned Undersea Vehicle (UUV) must operate in a complex, dynamic environment and must react intelligently to changing tactical, environmental, and mission variables with no outside intervention. To allow effective operation of a UUV, significant increases in system autonomy and on-board autonomous processing are required. This thesis is part of a larger effort to develop and demonstrate the capability of intelligent autonomy in increasingly difficult tactical situations and operating scenarios for maritime reconnaissance. The larger project will assemble autonomous planning and execution, advanced data fusion, and situation awareness to enable the UUV to autonomously avoid dynamic threats and obstacles as well as static obstacles in a complex operational environment.

One of the fundamental components in the development of this sophisticated autonomous capability is an accurate and real-time *map* of the UUV's surrounding environment. The UUV must acquire, interpret, and integrate information about its environment and use this information to construct and maintain an on-board map of its mission space. The map can then be used to achieve real-time mission objectives and allow the UUV to adapt to changes in the battlespace, environment, and vehicle health. This thesis develops an on-board mapping system that fuses and represents the data composing the UUV tactical scene.

1.1 Next-Generation UUV Mission Objectives

UUVs will be used to perform various missions including intelligence, surveillance, and reconnaissance (ISR); mine countermeasures; tactical oceanography; and submarine track and trail in vast, partially known, shallow-water coastal regions [6]. A possible ISR mission could require that the UUV be used for intelligence gathering and battlespace preparation in the shallow littorals, as described in Dunn [6] and as depicted in Figure 1-1. For example, the UUV would be launched from a host and would proceed autonomously to a designated *observation area*. Having reached the observation area of interest, the UUV would collect information for a predetermined period of time while autonomously repositioning itself in the observation area to obtain better data and to avoid threats. Finally, the UUV would return with the data set or would transmit the data set to the host before proceeding to another observation area. The entire mission may consist of multiple observation areas, each of which is a few square miles in size, that are located within the entire mission space, which is hundreds or thousands of square miles in size.

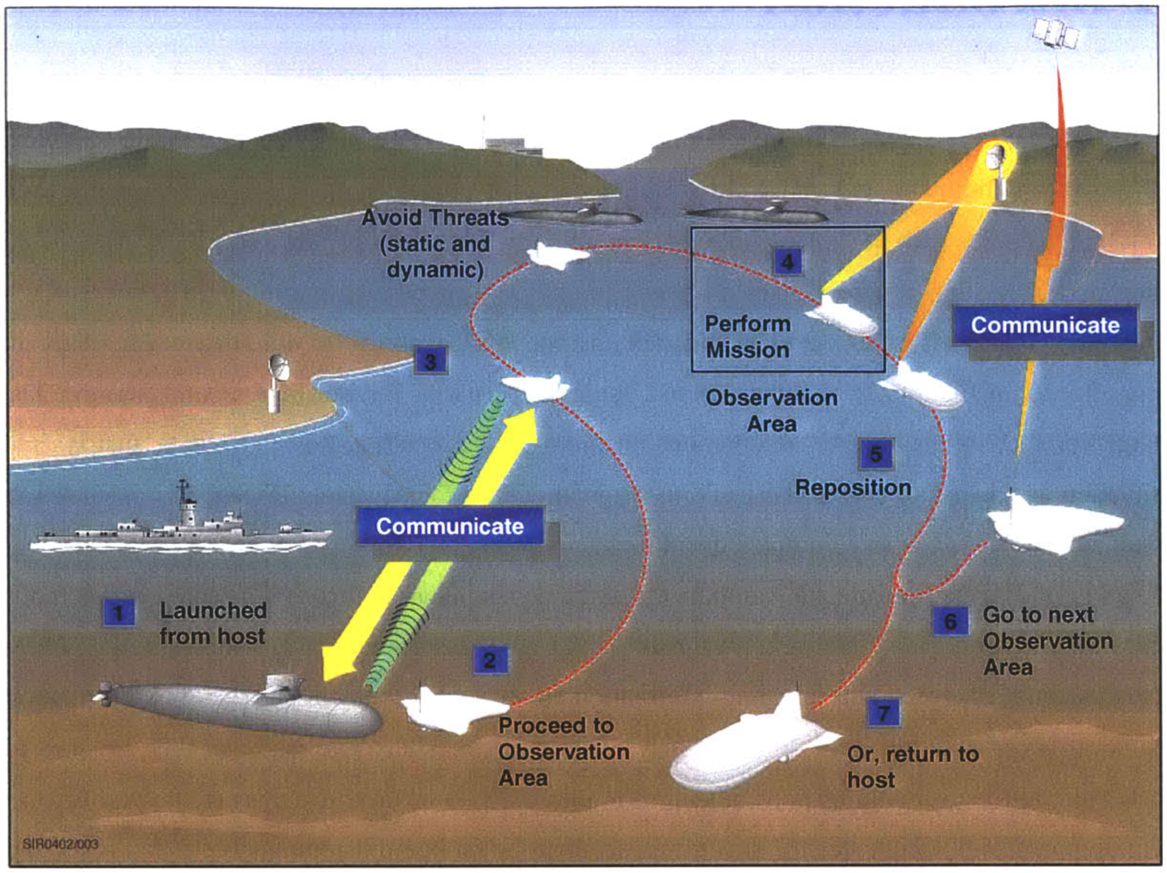

Figure 1-1: A possible ISR mission for a UUV

1.2 Mapping System Requirements

The UUV mission objectives generate derived requirements for the mapping system. The role of the mapping system is to capture the information composing the UUV's environment so that the UUV can use this information to safely and successfully complete its mission objectives. In order to successfully fulfill the objectives of the next-generation UUV missions, the mapping system must have the following functionality:

- **TRACK OBSTACLE LOCATIONS**

  Knowledge of the presence and location of all encountered obstacles is necessary for an accurate representation of the vehicle's environment and is necessary for mission objectives such as path planning, target localization, vehicle localization, situational awareness, and obstacle avoidance.

- **REPRESENT CONFIDENCE IN THE OBSTACLE'S LOCATION OVER TIME**
  Since the sonar measurements are inherently noisy, the map must have a way of interpreting and accumulating the measurements so that a measurement with a low level of confidence is given less credibility in the map than a measurement with a high level of confidence.

- **ALLOW QUICK, REAL-TIME UPDATES TO THE MAP**
  At every instant of time, the map must reflect an up-to-date view of the vehicle's current situation, since the vehicle operates in real-time and cannot adopt a stop-and-go strategy to wait until the processing of new information has finished.

- **REPRESENT UNDERWATER OBSTACLES**
  The underwater environment is composed of a set of threats, which may be either static or dynamic. Static obstacles consist of the underwater terrain, or bathymetry, such as landmasses, channels, and harbor entrances. Dynamic obstacles consist of other autonomous vehicles, submarines, and surfaces ships. The two primary types of information that are required to be modeled in the mapping system are bathymetry and threat data.

- **SUPPORT AN A PRIORI LOAD OF OBSTACLE DATA**
  Since an *a priori* map for a very shallow-water, coastal region may not be highly accurate, the mapping system must be able to handle any degree of accuracy in this *a priori* data and must be able to update the *a priori* data in an intelligent manner with the new sonar measurements over time.

1.3 Problem Statement

This thesis addresses the problem of representing and managing information about an underwater environment composed of both static and dynamic obstacles. The main objective of this thesis is to create an on-board mapping system that captures the information composing the UUV tactical scene. The challenges of the mapping system are to obtain useful and comprehensive information about the environment, handle sensor data quality issues, represent and fuse this data into an on-board map, update the map in real-time when new data is discovered, and save the map for future use while maintaining both efficiency and accuracy.

This thesis presents the design and implementation of a sonar-based mapping system for a UUV, and discusses the elements of the mapping system design: the representation of static and dynamic obstacles in a mapping system, the need for efficient data structures, the incorporation of sonar measurement uncertainty, and the assimilation of new information into the map. While the map should be able to support applications such as path planning, the map structure should not be too specific to one particular application such that it does not support other applications. Note that this thesis does not address the applications that could use the map, but only the mapping system itself. Applications for the mapping system are discussed only in brief detail in the next section as motivations for developing the capability of the mapping system.

Since the mapping system must handle noisy sonar measurements, this thesis also examines the stochastic modeling of sonar noise. This thesis presents a model for the noise in a sonar measurement and the incorporation of this model into the existing UUV simulation.

1.4 Motivating Applications for the Mapping System

The mapping system can be used for a variety of applications. While the mapping system developed in this thesis is not specific to a particular application, it is desired that the mapping system could be used for any or all of the typical mission applications shown in Figure 1-2 and discussed below.

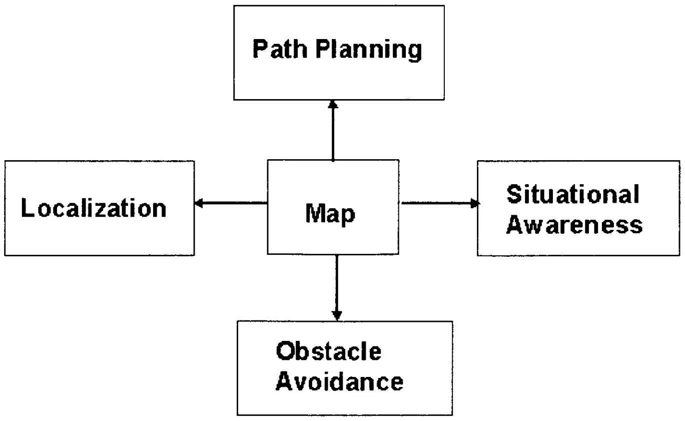

Figure1-2: Mission applications that can benefit from mapping system

1.4.1 Path Planning

Path planning is a frequently encountered and much studied problem, not only for UUVs but also for many types of robots. Path planning is the problem of finding an obstacle-free path from a source to a goal while minimizing a certain criterion, such as distance traveled, time traveled, energy expended, or time exposed to danger. The majority of the path planning literature assumes that the vehicle or robot has complete and accurate knowledge about its environment *a priori*, but less research focuses on the more realistic problem of path planning in dynamic, unknown, or partially known environments. In such a case, the vehicle must build and maintain a map of its environment, from which map-based path planning can be performed.

As part of DARPA's Autonomous Minehunting and Mapping Technologies (AMMT) program [26], the capability was developed at Draper Laboratory to successfully operate a UUV in an unknown environment, to generate and execute mission plans in a variety of situations, to survey and identify various underwater mines, and to generate a map of the environment [28]. During the mission, the planner receives terrain and obstacle data from a forward-looking sonar, and the planner uses this information to construct a map. This map allows the planner to adapt the mission plan to changes during the mission. However, the capability of the path planner is limited by the quality of the information in the map. AMMT uses a fixed map, such that for a given invocation of the path planner, the map is assumed to be fixed (i.e., unchanged). In addition, the map used for the path planning represents obstacles as either present or absent in a given area, with no allowance for the degree of uncertainty in the knowledge of an obstacle's presence. An important conclusion of the AMMT program is that continued development in improving the quality of the information in the map is critical to increasing the performance of the path planner [28].

McKeever [23] developed a path planning algorithm for an autonomous underwater vehicle for a track and trail mission, in which the vehicle is tasked to follow a moving target for an indefinite period of time. This path planner uses a two-dimensional map that is assumed to be fully known *a priori* and is unchanged throughout the course of the track and trail mission. As in AMMT, the fixed map only accounts for static obstacles, while it does not take into account that moving obstacles, such as surface traffic or enemy vehicles, may be present. The uncertainty in the presence of an obstacle is also not taken into account.

With improvements to the production and maintenance of the map, path planning can be expanded to plan in dynamic environments and to plan explicitly with moving obstacles and with stochastic data. Threat information and data quality issues can be factored into the path planning algorithms through the use of an improved on-board mapping system. With a useful and accurate map of its environment, a UUV could generate up-to-date plans based on the current external situation in a highly cluttered, uncertain, and dynamic environment.

1.4.2 Obstacle Avoidance

Avoidance of obstacles can be performed in a number of ways, one of which is map-based obstacle avoidance. When map-based obstacle avoidance is used, obstacles are stored in the map when they are detected. The map can be queried to determine if an obstacle is present in a certain vicinity of the vehicle, and the vehicle can use this information to avoid obstacles. Avoidance of moving targets can be accomplished by storing the moving obstacles as part of the map. Other methods of obstacle avoidance besides map-based obstacles avoidance can be used in the case where an obstacle becomes visible at the last minute and is not yet incorporated into the map.

1.4.3 Vehicle Localization

The vehicle localization problem is one of maintaining a reliable estimate of the vehicle's position within the environment. Map correlation (or map matching) is a vehicle localization technique that is often used with grid maps to determine an estimate of the vehicle's position [11]. A map of recent measurements is generated and then compared to a previously constructed map. The best map match can be found by correlating the recent map to the previous map, and this match provides the new estimate of position. This position estimate is used to merge the recent map with the previous map [10], [11].

Concurrent mapping and localization (CML) is a vehicle localization technique that is capable of generating the map at the same time as the navigation is being performed. CML uses information in the environment to provide an estimate of the vehicle's position, and as a result, CML allows an autonomous vehicle to build a map of an unknown environment while simultaneously using this map to improve the estimate of its own position. For interesting research on this problem, see [9], [10], [18].

1.4.4 Target Localization

Verifying the location of a target can be performed by matching the measured position of a target with a previous estimate of the position of the target. In a manner similar to that used in map correlation for vehicle localization, a map of recent measurements of the target can be generated and compared to the previously constructed map. The location of the target is reconciled between the maps, and the target position is estimated and incorporated into the previously constructed map.

1.4.5 Situational Awareness

Situational awareness, or tactical scene generation, involves the autonomous construction and maintenance of internally consistent, sophisticated environmental representations from many different sources, including various types of sensors and intelligence updates. The information available from a mapping system would form a large part of the information used by the situational awareness elements. Situational awareness can be integrated with the vehicle's planning and control system to increase the capability of vehicles.

1.5 Manta Test Vehicle and Simulation

The Manta Test Vehicle (MTV) is a test platform UUV designed and developed by the Naval Undersea Warfare Center (NUWC) to allow implementation of future UUV payloads, performance of at-sea demonstrations in shallow water, and evaluation of advanced UUV technologies [27]. The MTV is a large size (34.25 feet in length), modular, reconfigurable UUV that has a large payload capacity, high-accuracy navigation, acoustic communications, RF communications, low speed control, and is equipped with sophisticated sensors, including active forward-looking sonar, passive sonar, and ISR sensors.

To support the research, development, and evaluation at the subsystem and the system level of next-generation UUV technologies, Draper and NUWC have developed a high-fidelity simulation that is able to represent the unique operation of the MTV in a shallow-water operational environment. The simulation includes models of 6-degree-of-freedom vehicle dynamics, navigation sensors, active forward-looking sonar, and passive sonar. The simulation also supports the modeling of bathymetry and dynamic obstacles in the vehicle's operational environment [27].

Currently, static obstacles are modeled as terrain using bathymetry data from Narragansett Bay and can be detected with the forward-looking sonar. Dynamic obstacles are modeled as non-aggressive contacts that can be detected with the passive sonar. However, the forward-looking sonar and passive sonar models lack a model of the noise in a sonar measurement, and as such, the models do not give a measure of confidence in a particular sonar measurement. Since the mapping system must handle noisy sensor data and since this thesis uses the MTV simulation and its associated environmental models for the development and testing of the mapping system, this thesis must also address the modeling of the uncertainty in a sonar measurement.

1.6 Thesis Organization

This chapter has introduced the UUV's mission requirements (specifically for the ISR mission), the mapping system requirements, the problems addressed in and the scope of this thesis, the motivation and uses for the mapping system, and the test platform for the development and testing of the mapping system. Chapter 2 describes in detail map representations that have been used in mapping systems and other representations that could be applied to mapping systems. Chapter 3 discusses the sources of noise in a sonar system, describes a model for the uncertainty in a sonar measurement, and presents the integration of the sonar measurement uncertainty model into the existing UUV simulation. Chapter 4 presents the design and implementation of the static obstacle map, which stores information about stationary obstacles in the UUV's environment. Chapter 5 details the framework of the dynamic obstacle map, which stores information about moving threats in the UUV's environment. Chapter 6 describes the results and analysis of this mapping system. Chapter 7 summarizes the work presented in this thesis and discusses several areas for future research.

2 Map Representation

The representation of spatial information is important not only in mapping but also in robotics, computer vision, computer graphics, geographical information systems, image processing, and computer-aided design. Numerous representations are currently in use. Each representation has advantages and disadvantages in terms of the amount of memory required to store the representation, the speed to access the representation, and the compatibility of the representation to the goals and functionality of the particular application. Rather than an exhaustive literature review of every possible region representation, this chapter will instead focus on and discuss representations that have been used in or could be applied to mapping, as well as some of the tradeoffs involved in choosing a map representation.

Approaches to region representation can be characterized into two fundamental paradigms: (1) the grid-based, or metric, paradigm that represents the environment as a tessellation of discrete cells and (2) the topological, or feature-based, paradigm that represents the environment as a set of detectable objects or features.

2.1 Grid-based Paradigm

A grid-based representation, or cell decomposition, decomposes the environment into a grid of cells. The decomposition may divide the region into equal-size cells (e.g., a uniform grid), or it may divide the region into non-uniform cells based on the input (e.g., a hierarchical structure). The resolution of the decomposition process, or the size of the smallest cell in the decomposition, may be fixed beforehand, or it may be variable and dependent on properties of the input data.

Grid-based representations are commonly used in mapping because the complexity and computational requirements of the grid-based representation, for a given environment size, are determined solely by the number of grid cells in the representation, while the complexity of the feature-based representation is directly dependent on the number of obstacles or obstacle features that are stored [42]. Thus, the complexity of a grid-based map is independent of the environment complexity, while the complexity of a feature-based map will increase as the complexity of the environment increases.

Grid-based representations are also applicable for mapping because grids are amenable to data fusion from multiple sensors and because grids can store information about any obstacle, regardless of whether or not it can be represented as a recognizable feature or set of features. As a result, a grid representation is attractive for representing an unstructured environment such as the unstructured terrain in an undersea environment. However, since grid representations do not require assumptions about the source, nature, or properties of the data, grid representations do not directly capture the physical properties of the data and do not require the differentiation between different types of data. While it would be possible to augment the grid-based map with this extra information, the basic grid map does not assume knowledge about the type of obstacle in the map. In a basic grid map of an underwater environment, the differentiation between a terrain obstacle and another stationary obstacle such as a bridge is not made. While this lack of detail would not allow grids to explicitly characterize the geometry of the reflecting surfaces of the underwater obstacles, such that it would not be possible to accurately predict the values of individual sensor returns [18], the benefit of this lack of detail is that a basic grid-based representation may have a smaller representation cost than a feature-based representation [15].

A drawback to the use of a grid representation for mapping is that a grid does not directly and easily handle navigation error and uncertainty in the vehicle's position. The accuracy of a grid-based map representation depends crucially on the alignment of the vehicle with the map [39]. If a map is built assuming that the vehicle is located at one position and later the vehicle discovers that it is actually located in a different position, then the information in the map will be in error unless the information can be translated in some manner from the incorrect grid cell to the correct grid cell to account for the difference in the vehicle's real position. Thus, a grid-representation is best suited to a mapping system in which the vehicle can be fairly certain of its location at the time of the update to the map.

A grid-based representation is a discrete approximation to the environment because a grid cell represents information about a location that lies somewhere within the grid cell. As a result, another disadvantage of a grid-based representation is that this discretization involves a loss of detail that increases as the size of the grid cell increases. If the size of the grid cell is very small, then the grid will closely approximate the environment. If the size of the grid cell is large, such that the grid cell is much larger than the obstacles, then the grid will be only roughly approximate the locations of the obstacles, since the obstacle will be known only to lie somewhere within the grid cell.

A grid approach is well suited to represent static environments, but representing dynamic environments with a grid approach is more challenging [17]. One approach to representing a dynamic environment is to convert the problem of mapping a two-dimensional environment with dynamic obstacles to a new problem of mapping a three-dimensional environment with static obstacles [12], [13]. In this new problem, time is the third dimension, and the two-dimensional dynamic obstacles are transformed into the space-time domain as three-dimensional static obstacles. Thus, the number of dimensions in this grid representation for a dynamic environment will always be one higher than the number of spatial dimensions. A significant amount of memory will be required to store the augmented representation, as well as an added time-complexity for operations that use the augmented representation. This method of transforming a lower dimensional problem to a higher dimensional problem is limited by the "curse of dimensionality," since the memory requirements dramatically increase as the number of dimensions increases. Thus, even when the space is two-dimensional, the augmentation to three dimensions will require a significant increase in the memory required for the grid. Another approach is to use a distance map to keep track of the distance from the goal to each cell in the grid. (See § 2.1.1.3.)

2.1.1 Uniform Grids

Uniform grids, also referred to as regular grids, are a grid-based representation where the decomposition is uniform and the region is divided up into a grid of disjoint equal-size cells. A uniform grid is typically implemented as an array. A uniform grid cell does not need to store its index because its index can be determined easily from its position in the grid. A major benefit to a uniform grid representation is that a uniform grid cell can be updated in constant time, and information can be retrieved in constant time and used for real-time objectives. However, a fundamental disadvantage is that, for a fixed resolution, the representation size increases with the size of the environment. As a result, a uniform grid with a high resolution, or a high level of detail, is too memory-intensive for a large area. This is especially a concern with a three-dimensional area, for which a three-dimensional uniform grid representation is almost always impractical.

Each cell in the grid contains information about the cell at that location. This information can be, for example, an occupancy value that indicates the probability that the cell is occupied (e.g., an *occupancy grid*), a depth or elevation value that indicates the height at the location of the cell (e.g., a *height field*), a distance value that indicates the distance of the cell from the reference or goal cell (e.g., a *distance map*), a binary value that indicates either a black or a white image pixel in the cell (e.g., an *image array*), or some combination of values. Occupancy grids, height fields, and distance maps are potentially applicable for a mapping system. They are typically represented with uniform grids, as described below, although they can be extended to be represented with a hierarchical structure (see § 2.1.2).

2.1.1.1 Occupancy Grids

Occupancy grids, also referred to as certainty grids or evidence grids are a type of uniform grid where each cell stores a certainty value that represents the probability that the cell is occupied [7], [24], [25]. Cells that are definitely free correspond to a certainty value of 0, cells that are definitely occupied correspond to a certainty value of 1, and uncertain cells correspond to a value that can range between 0 and 1. Unknown cells typically are initialized to a certainty of 0.5. As multiple sensor readings of the area are obtained and accumulated, the certainty value is updated in proportion to the amount of confidence in the new information. Occupancy grids are robust and can function in the presence of noise or sensor errors. Since the accumulation of multiple measurements over time will resolve ambiguities, some inaccurate sensor readings will not affect the long-term overall integrity of the map.

Occupancy grids were originally proposed in [7], [24], and have since been implemented successfully in various systems. Much of the work that uses occupancy grids has been done with land robots in two-dimensional indoor environments [7], [17], [24], [25]. However, occupancy grids have also been applied to two-dimensional outdoor, terrestrial environments [37], to three-dimensional indoor environments [21], and even to three-dimensional aerial environments [33] and underwater environments [23], [28]. An occupancy grid is used as the map for the path planner in the AMMT program [28], and McKeever [23] uses an occupancy grid as the map for a track-and-trail path planner.

Occupancy grids have also been extended and used to incorporate measurements from various types of sensors (e.g., sonar, laser rangefinder, camera, etc.), as well as to fuse information from multiple sensors [21], [24]. Occupancy grids have also been used in combination with other representations, such as a topological approach, in order to take advantage of the benefits of each representation while attempting to minimize the disadvantages [39]. A detailed, annotated bibliography of the work that uses occupancy grids can be found in Martin and Moravec [21].

2.1.1.2 Height Fields

A *height field*, also referred to as an elevation map, is a discrete two-dimensional set of height samples $H(x, y)$ over a planar domain, where $H(x, y)$ is the height at position (x, y) [14]. While the set of height values can be non-uniform, a height field is typically associated with a uniform grid of points elevated above the x-y plane. Thus, a uniform height field is a two-dimensional uniform sampling of the three-dimensional data set.

Height fields are useful for representing images in computer graphics and for representing terrain in digital terrain models and in bathymetry maps for underwater environments [14], [33]. A uniform height field of the terrain in an underwater environment is created by dividing up the surface of the Earth in the area of interest into a two-dimensional uniform grid of cells.[1] Each cell in the grid stores a measure of the depth at the specified location. When there are multiple measurements for the same grid cell or when there is a measure of certainty available from the sensor measurement, the cell can store statistics for the depth, such as the mean and standard deviation of the depth. Thus, the height field measurement stored at grid location *(x, y)* is the mean and the standard deviation of the depth of the ocean floor beneath the surface position *(x, y)* [33].

A height field representation can only represent solid obstacles (or obstacles that can be treated as solid obstacles), since a height field representation stores the minimum depth of the obstacles at each grid location. As a result, a height field cannot represent hollow obstacles such as caves, overhangs, or other partially hollow terrain. In order to use a height field representation, the vehicle must be constrained to travel at or above the minimum depth value stored in the map, as opposed to under or through non-solid obstacles. For most applications of autonomous underwater vehicles, this constraint on the movement of the vehicle is satisfied.

2.1.1.3 Distance Maps

A *distance map* stores the distance from every cell in the map to the reference or goal cell [2]. Generating these distance values, or distance transforms, can be computationally expensive, although re-computing only the values that have changed can reduce this computation somewhat. A digital distance map can be used to represent moving obstacles by updating only the portions of the digital map that will potentially change as an object moves. A distance map is well suited for the path planning application, but the distance map is not as directly applicable to other objectives and applications.

A distance map is typically represented as a uniform grid, where each uniform cell stores the distance value [2]. However, Zelinsky [44] and Samet [30] extend the uniform distance map to use the quadtree hierarchical structure, where each non-uniform cell stores the distance value. (Quadtrees will be discussed in § 2.1.2.)

[1] The surface is assumed to be flat in the area of interest.

2.1.2 Hierarchical Structures

Hierarchical data structures are a type of variable resolution spatial data structure which recursively decompose the region with respect to the space occupied by it. Depending on the nature of the data and the particular implementation, hierarchical structures may require less memory than uniform grids, and they may require less computational time for operations such as search. Since the number of cells in a uniform grid can be quite large, it is desirable to combine equal-valued, or homogeneous, cells into the minimum set of homogeneous subspaces that can represent the region. This variable resolution decomposition allows the region to be represented with as few subspaces as possible and focuses the computational resources on the interesting sets of data. The number of subspaces in the hierarchical decomposition affects the amount of memory required to store the data and the amount of computation required to process and update the data. A smaller amount of subspaces generally results in a smaller amount of memory and a smaller amount of computation necessary for those algorithms.

Hierarchical data structures are conceptually clear and easy to implement. However, hierarchical methods are sensitive to the placement of the origin of the hierarchical structure within the region of interest. Thus, the placement of objects relative to decomposition lines of the space in which they are embedded affects their storage costs and the amount of decomposition that takes place.

The following sections will discuss a few of the different types of hierarchical data structures: the *quadtree*, the *octree*, and the *k-d tree*. Although these data structures are examples of grid-based hierarchical data structures, hierarchy is not specific to grid-based structures. A tree hierarchy could also be used in conjunction with topological representations, such as objects lists, to sort the objects according to their locations and to speed up the process of searching through the object list to find a desired object.

2.1.2.1 Quadtrees

The term *quadtree* is used in a general sense to describe a class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space. There are many variations of the quadtree, and these are discussed thoroughly in Samet [31], [32]. The specific type of quadtree structure that is applicable to spatial representation is the *region quadtree*. For simplicity, a region quadtree will be referred to here simply as a *quadtree*.

A quadtree recursively subdivides a grid into four quadrants until the largest quadrants are obtained where each cell in the quadrant contains the same value as the other cells in the quadrant (i.e., maximal square blocks) or until the smallest resolution cell is reached. This decomposition method is a regular decomposition of space. Since the quadtree subdivision produces disjoint quadrants, the union of these maximal square blocks can represent the entire region. While the decomposition of the space is not limited to be partitions of the same size, this is the case in the most common formulation of the quadtree [31], [42].

A tree of degree four represents the decomposition process. Each node is either a parent or a leaf. Each parent node always has four children nodes and corresponds to an inhomogeneous block. Each leaf node has zero children nodes and corresponds to a homogeneous block for which no further subdivision is necessary. Each of the four children nodes represents a quadrant (typically in the order NW, NE, SW, SE) of the region represented by its parent node. Figure 2-1 [31] shows an example of a quadtree representation for a binary region. (a) shows the region, where the grey area is occupied and the white area is free. (b) shows the uniform grid representation of the region as a binary array, where a 1 represents occupied and a 0 represents free. (c) shows the maximal block tesselation, where the smaller uniform cells in (b) are grouped into the largest possible homogeneous cells. (d) shows the quadtree representation, where a leaf node represents a homogeneous sub-region (either all 1's or all 0's) and a parent node represents a heterogeneous sub-region (a mixture of 1's and 0's).

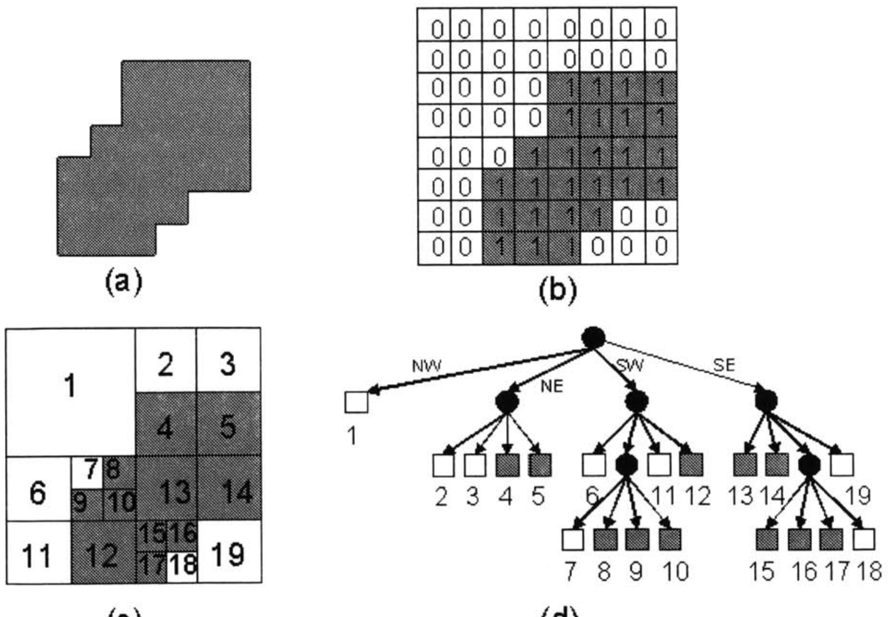

Figure 2-1: Example of a quadtree representation of a region
(a) Sample region, (b) its uniform grid representation as a binary array, (c) its quadtree representation in terms of maximal blocks, and (d) the corresponding quadtree data structure

A disadvantage of the quadtree representation, and all of the hierarchical structures, is that the resulting tree structure is dependent on the placement of the quadtree origin within the region. In other words, the structure of the quadtree is dependent on the placement of the obstacles with respect to the discrete grid boundaries. However, Kambhampati and Davis [15] point out that the savings in memory and computational time due to the use of a quadtree instead of a uniform grid are still high on average. In addition, Samet [31] points out that if the region is complicated, then the optimal positioning of the origin of the quadtree will generally not affect the amount of decomposition and storage cost required.

A quadtree representation can save memory as compared to a uniform grid representation of the same space. Kambhampati and Davis [15] and Samet [31] describe the memory savings of a quadtree structure compared to a uniform grid structure. The Quadtree Complexity Theorem states that the number of nodes in a quadtree region representation for a simple polygonal region (i.e. with nonintersecting edges and without holes) is $O(p+q)$ for a $2^q \times 2^q$ image with the obstacles' perimeter p measured in pixel-widths. Except for pathological case, q is negligible, and the number of nodes is proportional to the perimeter of the obstacles. In other words, if the resolution doubles (i.e., the perimeter approximately doubles), the number of nodes in the quadtree doubles [31]. This can be a significant memory savings, when compared to the number of cells in the uniform grid, which is proportional to the square of the number of smallest resolution cells on a side of the grid. When the resolution doubles, the number of cells in a uniform grid quadruples. However, depending on the distribution of the data and the particular implementation of the quadtree, a quadtree may not necessarily save space. Quadtrees are ideal for arbitrarily distributed data, and the quadtree will degenerate to a uniform grid (with the extra overhead required to store its index, location, and relationship to its parent and children cells), when the data is uniformly distributed. The most common formulation of the quadtree is a tree structure that uses pointers to its parent and children nodes, and this requires additional overhead to store the pointers to the internal nodes in the tree.

Many path planners, navigation systems, and mapping systems in the literature use occupancy grids for the map representation [7], [21], [24], [25], [37]. However, recent research is incorporating the quadtree structure into mapping systems and map-based path planners. Zelinsky [44] and Kambhampati and Davis [15] use the quadtree to represent a non-uniform distance map of the environment. Stentz et al. [42], [43] and Chen [4] use the *framed quadtree*, a modified version of the quadtree, as the map representation for finding shortest paths in a two-dimensional environment.

The *framed quadtree* is similar to the quadtree, except that cells of the highest resolution are added around the perimeter of each quadtree subspace [4], [42]. The framed quadtree is specifically suited to the path planning application, where the optimality of the path is a concern. The advantage of using a framed quadtree for a map representation that is specifically used for path planned is that a path is not constrained to go through the center of the cells, but can enter and exit the cells at almost any place due to the high-resolution perimeter cells. As a result, framed quadtrees produce more efficient paths than those produced from quadtrees. However, framed quadtrees require more memory than quadtrees, and they can even require more memory than uniform grids in uniformly, highly cluttered environments due to the overhead involved with storing the border cells. While framed quadtrees can be useful for the specific application of path planning, a mapping system that should be useful for a variety of purposes does not necessarily benefit from these border cells.

2.1.2.2 Octrees

The *octree* data structure is the extension of the quadtree to three dimensions and is used to represent three-dimensional data [5], [31]. A space in the form of a cubical volume is recursively subdivided into eight congruent disjoint cubes, or octants, until blocks are obtained of a uniform value or a predetermined level of decomposition is reached. Whereas the quadtree has a branching factor of four, the octree has a branching factor of eight. In general, a quadtree or octree can be generalized to d dimensions, where each node is split along all d dimensions leading to 2^d children. However, the memory requirements increase dramatically as the number of dimensions increases, since the branching factor increases as 2^d. The Quadtree Complexity Theorem can be extended to three-dimensional data, where the number of nodes is proportional to the surface area instead of the perimeter. Similarly, the theorem can be extended to d-dimensional data, where the number of nodes is proportional to the $(d-1)$-dimensional interfaces between the data [31].

An octree data structure can allow a significant reduction in the memory requirements compared to a three-dimensional uniform grid. However, octrees can use a lot of memory even when they are just a few levels deep because each subdivision creates eight new cells. Abramson et al. [1] uses the octree data structure to represent a three-dimensional underwater environment in the context of path planning, and Chen [5] uses the *framed-octree*, a variant of the octree, to find conditional shortest paths in three-dimensional environments.

2.1.2.3 K-d Trees

The *k-d tree* is closely related to the quadtree [33]. Both a quadtree and a k-d tree recursively subdivide a region into subspaces. However, the main difference is the method of decomposition of the region. For each level of the tree structure, each dimension of the k-d tree is divided into two subspaces, while a *d*-dimensional quadtree is divided into 2^d subspaces. In other words, the quadtree is not a binary search tree because each level in the tree has 2^d children. The k-d tree is a binary search tree where at each depth the direction of subdivision alternates along the dimensions. For example, in two dimensions, the *x*-coordinates are compared at odd levels, and the *y*-coordinates are compared at even levels. For two-dimensional space, a k-d tree divides space into rectangles. In three dimensions, the direction of the split alternates between the *x*-, *y*-, and *z*-coordinates.

The efficiency of a k-d tree decreases as the number of dimensions increases. An advantage of the k-d tree over the quadtree is that the k-d tree contains only two child pointers instead of four. For certain types of space, reducing the branching factor of a hierarchical representation can further reduce the total number of cells in a tree. As with the quadtree, the efficiency and memory usage of k-d trees is highly dependent on the implementation and the structure of the environment.

Another difference between a k-d tree and a quadtree is that the k-d tree was initially developed to deal with multidimensional point data, as opposed to region data. As a result, the k-d tree has not been as widely used in mapping as the quadtree has been. However, Sammon [33] uses a k-d tree in conjunction with a uniform occupancy grid representation to implement a mapping system for an autonomous helicopter in a three-dimensional aerial environment. The immediate three-dimensional area around the vehicle is represented with a three-dimensional uniform occupancy grid as a local map, and the larger global area is represented with a three-dimensional k-d tree as a global map. The local map is defined in terms of an area around the vehicle and moves with the vehicle. As the vehicle moves, the information in the local map is transferred to the global map. This combination of a small-area local map and a large-area global map incorporates the advantages of both data structures. The local map can be updated quickly, and the global map can store a large amount of data more efficiently than is possible with a three-dimensional uniform grid. (This idea of a local map for the immediate area and a global map for the entire area is similar to that which will be used in the mapping system in this thesis.)

2.2 Topological Paradigm

Topological representations exploit geometrical features of the environment, such as points, lines, and arcs. These representations assume that geometric features are present in the environment and can be reliably detected and tracked. Data association is a significant issue in the use of a topological paradigm, since each sensor measurement must be assigned to belong to a particular feature or object.

A topological representation is more amenable to navigation error than the grid-based representation. Since location is just a property of the feature, if the vehicle's location is found to be in error, the location of each object can easily be updated accordingly. This is in contrast to the way in which a grid-based representation handles navigation error. This can be an advantage for outdoor environments.

Leonard [18] utilizes geometric representation and models the environment with a set of geometric primitives – points, lines, and planes. The size of the representation depends on the number of features or objects in the environment, and not on the size of the environment itself. A disadvantage to this approach is that the raw sensor data must first be interpreted in a way to extract the geometric features. The crucial issues are the representation of uncertainty, the reliability of feature extraction, and the speed with which the model can be constructed. Unstructured, natural terrain such as the floor of the ocean cannot always be represented well with these geometric primitives.

2.2.1 Object Lists

An *object list* stores a list of identified objects and their positions [33]. This representation is very compact when the environment is sparse, because object lists only store obstacles, not free space. Object lists are well suited for environments that have typical, recognizable features that are some subset of a list of possible features [20]. However, the representation of uncertainty can be difficult to maintain in an object list. Often, the estimate of uncertainty is maintained in a covariance matrix that is updated with a Kalman filter.

2.2.2 Visibility Graphs

A *visibility graph* represents the environment as a graph of the connectivity between open regions [20]. Edges connect visible nodes – the node pairs that do not have an obstacle in between. A visibility graph representation assumes that obstacles can be represented by polygons so that every vertex of each polygon is used to define a node in the graph. Accurate sensors and significant preprocessing are necessary to identify the obstacles, extract the features, package the obstacles as polygons, and build the visibility relationships in the graph. Advantages to visibility graphs is that they can be very compact representations and that they already contain the connectivity information that is necessary for certain applications such as path planning, allowing very efficient path planning by traversing the graph of the environment.

2.3 Summary

Many types of region representations have been applied to mapping systems. This chapter has served to introduce the two main paradigms of representations – grid-based representations and topological representations, to present the fundamental advantages and disadvantages of each representation, to highlight some previous work using these representations in mapping systems, and to motivate the design choices that will be presented in the remainder of this thesis.

3 Stochastic Modeling of Uncertainty in a Sonar Measurement

A simple and useful model of a measurement from the sonar sensors is a crucial component for designing and testing the mapping system. Models of the active forward-looking sonar (FLS) and the passive sonar exist in the UUV simulation, but the models lack a measure of confidence in the sonar measurements. Since the mapping system must handle noisy sensor measurements, we need a way to simulate this noisy data. The modification of the sonar model in parallel with the design of the mapping system is necessary. The primary objective in modifying the sonar model is to model the uncertainty in the sonar measurements, for both the active FLS and the passive sonar models. For each sonar measurement, the sonar model should give a measure of confidence in that measurement.

We must first decide what information is important and what changes to the model are needed to produce only this set of information. Deciding on the level of detail in the model is very important and highly dependent on how the model is to be used. A thorough understanding of the basic principles of a sonar system, the sources and effects of noise in a sonar system and uncertainty in a sonar measurement, and the current capabilities of the sonar system simulation will help us to determine what changes should be made to the current sonar models.

3.1 Basics Principles of Sonar

Sonar is useful for confirming predicted obstacles, for discovering unexpected obstacles, and for confirming that no obstacle exists in a given area. Depending on the source of the sound energy that is detected, sonar can be classified as either active sonar or passive sonar. Active sonar operates by generating a directed sound pulse, or ping, and by timing the reflected return off of features in the environment. Sound energy is projected from the transmitter into a region of three-dimensional space and may be absorbed or reflected by one or more features, including the seafloor, targets, or any other object that it hits. Some portion of the signal may be returned to the receiver after being attenuated or distorted by the medium and corrupted by noise. With knowledge of the speed of sound and the duration of the sound pulse, the sonar system can calculate the distance (range) and the angle (bearing) from the sonar receiver to the reflecting surface of the feature that first reflects the sound pulse. Range to an obstacle is calculated as half the difference in time between the pulse transmission and the return detection, multiplied by the speed of sound in the water column.

Passive sonar does not generate energy, but instead receives, or listens for, the energy generated from targets. As a result, passive sonar can detect bearing only. However, range can sometimes be deduced by performing maneuvers and geometric analysis. For both types of sonar, the signal received at any moment is a one-dimensional time series of measurements and is generally a summation of the contributions from multiple sources or reflectors [38].

3.2 Noise in the Sonar System

Noise in an underwater environment generally refers to acoustic noise that causes attenuation of the sonar signal as it is transmitted through a medium. Attenuation is the process of weakening or reducing the amplitude of the sonar signal, and it is caused by numerous factors such as material dispersion, beam spreading, and absorption [8]. The attenuation of a sonar signal hinders the detection of the sonar signal at the sonar receiver, and reflected signals from far away are sometimes attenuated to such a degree that the signal may not even be detected. Acoustic noise includes ambient noise, reverberation noise, and self noise. Ambient noise is the residual sound level remaining after all identifiable, transient noise sources have been removed and includes surface weather, biological organisms, and commercial shipping traffic. Ambient noise models treat noise sources as variable densities distributed over large areas, and they predict the mean levels sensed by an acoustical sonar receiver. Reverberation, which consists of bottom reverberation, volume reverberation, and target reverberation, is transient sound caused by the echoing of a sonar signal off of an object, such as the seafloor or a submarine. Since the sonar must transmit energy in order to observe reverberation, reverberation is not present in passive sonar systems. Self noise is the near field noise introduced into the sonar system as the vehicle propels itself and is caused by the machinery of the vehicle and the flow of the current around the sonar. For more detail on acoustic propagation in an underwater environment and acoustic modeling, see Urick [40] and Etter [8].

3.3 Sources of Uncertainty in the Sonar Measurement

All of the noise that is introduced into the transmit-receive process in the sonar system becomes evident in the observed sonar measurement. Because the received sonar pulse waveform is represented as a one-dimensional function of time, while the sound energy has passed through a three-dimensional volume of space, there are several sources of uncertainty in a sonar measurement of a target.

- **ANGLE OF REFLECTION**

  Uncertainty exists in the angle of the reflection off of the target due to the sonar beam's non-zero width and the irregular surfaces typical of underwater obstacles.

- **DROPOUTS OF SONAR RETURNS**

  Uncertainty arises due to dropouts when the reflecting surface does not generate a reflection capable of being detected, since all targets may not be detected if they scatter only weakly (e.g., non-reflective targets) or reflect energy away from the sonar receiver (e.g., very reflective, or "shiny," targets).

- **MULTIPLE PATH FOR SONAR RETURN**

  Uncertainty exists in the measurement of the obstacle's location due to multi-path returns, which occur when a sonar pulse reflects off of multiple surfaces before being detected. When one sonar pulse can travel along multiple paths from the sonar transmitter to the target and back to the sonar receiver, this can produce an erroneous measurement of the duration of the sound pulse, corresponding to an erroneous estimate of the range to the target. Acoustic echoes can return to the sonar along three different paths – (1) sonar transmitter to target to sonar receiver, (2) sonar transmitter to target to sea surface to sonar receiver, and (3) sonar transmitter to sea surface to target to sea surface to sonar receiver [45]. Since these three return paths take three different durations of time, the result will be three different measurements, where only one measurement represents the true location of the target. The other measurements correspond to overestimates of the range to the target or the illusion of the presence of other targets.

- **CROSSTALK**

  Uncertainty can occur when an array of multiple sonar sensors is used. When another sonar receiver detects a sonar pulse that does not emanate from the corresponding sonar transmitter, this crosstalk results in an erroneous measurement of the range [10].

- **LOCATION AND DIRECTION OF SONAR**
  Uncertainty also exists in the estimate of where the sonar is located and the direction in which it points.

- **ENVIRONMENTAL DISTURBANCES**
  Uncertainty is introduced due to currents, tide variations, sound speed profile variations, acoustic noise, inhomogeneous medium, and other statistical characteristics of the sonar beam transmit-receive process [38].

- **SHALLOW-WATER ENVIRONMENTS**
  Even more uncertainty is introduced in shallow water environments. Repeated interaction with the sea floor, high attenuation caused by interaction with the bottom, and horizontal refraction caused by repeated boundary reflections over the sloping bottom can cause the sonar system to return a noisy measurement [8]. A shallow water environment with a flat sea surface is a typical environment for multi-path propagation.

3.4 Types of Measurement Errors

Measurement errors can be characterized as either systematic errors or random errors [46]. Systematic errors can be predicted from some physical law or rule. Random errors are generally small errors resulting from the limitations of measuring devices and processes, are equally likely to be positive or negative, and are governed by the laws of probability. Random errors result from the inability to perfectly measure any quantity or to perfectly model any systematic error.

Measurement errors in the measured depth include residual systematic and instrument error in the sonar system; error in the measurement of the heave, pitch, and roll of the vehicle; error in the measurement of the velocity of sound in the water; error due to static vessel draft and dynamic vessel draft; error in the detection of the sea floor due to the varying density of the bottom material; and any other sources of error in the actual measurement process, such as errors associated with tide variations [46].

3.5 Active Forward-looking Sonar Simulation

Since the existing active sonar simulation does not model noise, we will consider modeling the uncertainty in a measurement from the active FLS and incorporating this uncertainty model into the existing UUV simulation.

3.5.1 Active Sonar Model

The forward-looking mode of the active sonar model is the sonar simulation mode that will be used for detecting static obstacles. The forward-looking mode of the simulation is based on the properties of the Reson Seabat 8101 multi-beam swath bathymetry sonar. The active sonar model consists of 101 receive beams, where each beam is located 1.5 degrees apart, as shown in Figure 3-1. Beams 0 to 49 point to the port (left) side of the UUV starting at -75 degrees and ending at –1.5 degrees. The 50th beam is oriented directly below the UUV at 0 degrees and is perpendicular to its major and minor axes. Beams 51 to 101 point to the starboard (right) side of the UUV starting at 1.5 degrees and ending at 75 degrees. The sonar measures relative water depths across this 150 degree swath width perpendicular to the vehicle's path. The depth measurements are based on the exact position and attitude of the UUV at the time of each ping and the exact speed of sound in the water column. For each of the 101 receive beams, the active sonar simulation determines the depth measurement by intersecting a ray directed out from the receive beam with the true bottom, looking up the value of the true depth at that latitude and longitude in the true depth terrain database, and returning this true value of the depth. The terrain database consists of a grid of cells, or *terrain cells*, and each sonar beam will intersect one of these terrain cells[2]. The terrain cells in the sonar simulation line up with the *sonar cells*[3], or grid cells in the grid superimposed onto the area of the seafloor surrounding the sonar.

[2] Each sonar beam is modeled as having zero width. Therefore, a sonar beam will only intersect one terrain cell, as opposed to possibly multiple terrain cells.
[3] Note that the terrain cells exist only in the sonar simulation and not in the actual sonar, while the sonar cells exist in both the simulation and the actual sonar.

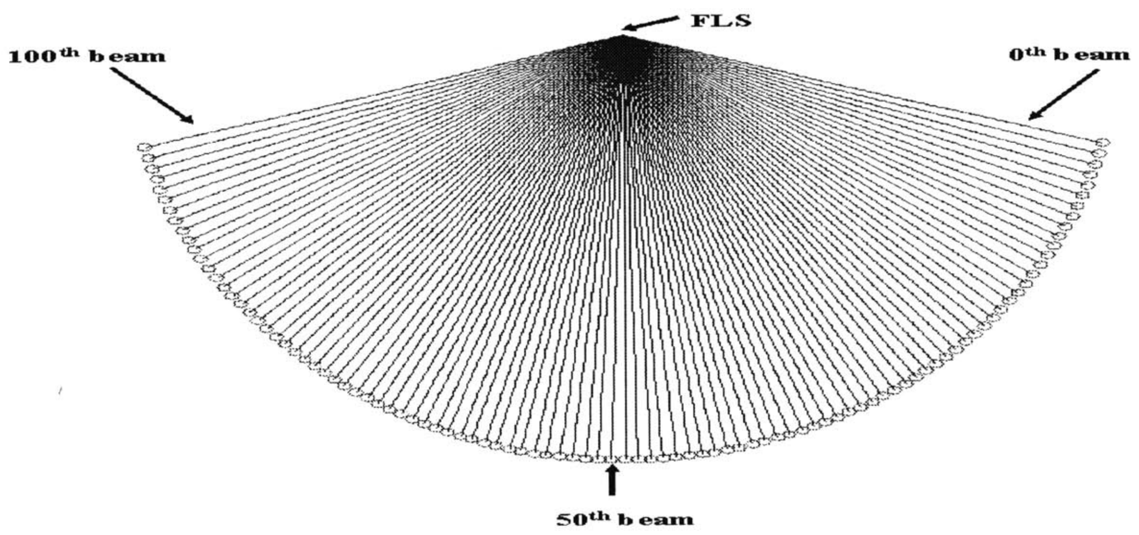

Figure 3-1: FLS beams

3.5.2 Modeling Uncertainty in an Active Sonar Measurement

The sonar subsystem consists of the physical sonar transmit-receive process and the sonar signal processing (see Figure 3-2). The sonar transmit-receive process runs at a faster rate than the sonar signal processing, and consequently, the sonar signal processing is able to calculate a statistical average of the sonar pings over the time series of noisy data. The output of the sonar system – and thus the input to the mapping system – is not only this statistical average as the measurement, but also a measure of confidence (standard deviation) associated with this measurement. We aim to model the output of the sonar subsystem. Each simulated active sonar measurement should represent the statistical average over the time series of noisy data. If the time series of data can be modeled in some manner, then the active sonar simulation can output a measure of confidence in its depth measurement.

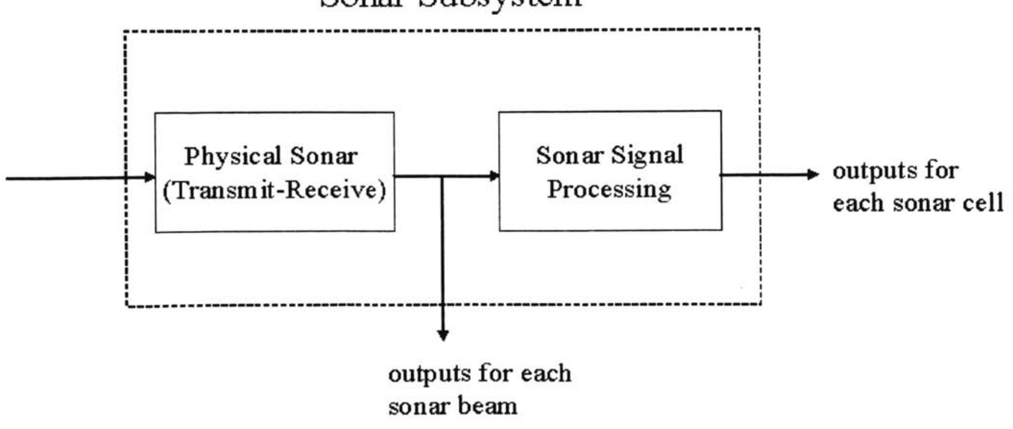
Sonar Subsystem
Physical Sonar (Transmit-Receive)
Sonar Signal Processing
outputs for each sonar cell
outputs for each sonar beam

Figure 3-2: Sonar subsystem

One possible approach to generating a measure of uncertainty for each sonar measurement is to model the mean and standard deviation for each of the 101 sonar beams. We could choose the trivial approach of assigning each beam the same default values for the mean and standard deviation, but this is useless for the mapping system. However, generating a useful value for the mean and standard deviation for each beam is impossible without detailed information about the distributions for the individual sources of noise, an elaborate model of the physics of the sonar, a detailed model of the shallow water environment, and a model for the reflection of the individual sonar beams off of the various obstacles in the environment. This is an analytical approach and would involve developing a sophisticated model for each of these sources of uncertainty.

Another approach to modeling the uncertainty in a sonar measurement is to model the mean and the standard deviation for a grouping of beams. This would model the noise in the sonar subsystem as a combination of the noise in the sonar transmit-receive process and the noise in the sonar processing of the receive beams, so that the standard deviation represents the uncertainty in a sample of depth measurements. Thus, we would be computing a statistical average over a small area of space instead of over time, but the result would be similar.

Separation of the sonar model into a model of the sonar transmit-receive process and a model of the sonar processing is unnecessarily complex for the purpose of generating a confidence value in the depth measurement for each sonar return. Instead, we only need a simple and useful model of a noisy measurement from the overall sonar subsystem. While an elaborate model of the physics of the sonar, the shallow water environment, and the interactions of the sonar beams reflecting off of the various obstacles in the environment may seem potentially useful at first, this level of detail is difficult and inefficient to model and not very applicable to the purposes of the mapping system. We do not need an analytical model that has its origins in the physics of underwater acoustics and the sonar equation. Instead of modeling every parameter and possible interaction to generate a mean and variance for each sonar beam, we take a probabilistic approach (the second approach described above) to generate a mean and a standard deviation for each *group* of sonar beams. Individual sonar beams are grouped according to the sonar cell that they intersect when the beam is projected as a ray from the sonar receiver to the surface of the terrain. Thus, each sonar measurement will be a measurement for a group of sonar beams, which corresponds to one sonar cell, instead of for an individual beam.

We assume that the uncertainty in a sonar measurement is only due to random errors and not to systematic errors. We model random errors as white Gaussian noise. Since systematic errors are a bias, we could choose to include systematic errors in the uncertainty model by adding a non-zero mean to the Gaussian random errors. However, we will assume that the systematic errors are accounted for with the sonar signal processing before the sonar measurement is output from the sonar subsystem. The particular assumptions that are we are making are dictated by the objectives of and the underlying motivation for the noise model being developed.

Since measurement noise in the physical sonar system is caused by a number of small errors, and since we do not know detailed information about the distributions of the individual sources of noise, we choose to model the aggregation of the noise. When a number of independent random variables are combined, the Central Limit Theorem states that a Gaussian distribution, regardless of the shape of the individual densities, can describe the summed effect very closely [22]. Gaussian distributions are also practical and mathematically tractable. Since the Gaussian density is completely determined by its first and second order statistics (mean and standard deviation), the Gaussian density is the best form to assume in the absence of any higher order statistics. Most other densities require a large number of higher statistics to determine their shape exactly [22]. Thus, a Gaussian distribution provides a practical and adequate representation for modeling the uncertainty in a sonar measurement.

Let X be a Gaussian random variable that describes the distribution of the noise in the active sonar measurement for a particular sonar grid cell. Let $(X_1,\ldots, X_n)$ be a random sample of size n of the noise, where n is the number of individual sonar beams that correspond to the particular sonar cell. Each X_i, for $i=1\ldots n$, has mean μ and variance σ^2 and is itself a measurement of the noise for this particular sonar cell. We assume that each sonar measurement X_i is equally likely to be corrupted with noise and that each measurement X_i is independent of the other measurements. Thus, we are assuming that the X_i's are independent and identically distributed. For each sonar cell, we will use the distribution of the mean of X and the distribution of the variance of X, which depend on n, to get a value for the mean of the noise and the standard deviation of the noise. The distribution of the mean is a Gaussian random variable with mean μ and variance $\frac{1}{n}\sigma^2$ [16]. (See Appendix A.1 for the derivation of the distribution for the sample mean.) We will then generate a random number from the distribution of the mean, and this will be the mean of the noise for the sonar grid. The distribution of the sample variance is a Gaussian random variable with mean σ^2 and variance $\frac{2}{(n-1)}\sigma^4$. (See Appendix A.2 for a derivation of the distribution for the sample variance.) Similarly, we will generate a random number from the distribution of the variance, and this will be the variance of the noise for the sonar grid cell. The simulated standard deviation of the noise is calculated as the square root of the simulated variance. The mean value for the noise is added to the true value of the depth, and the standard deviation of the noise represents the uncertainty in this sonar measurement. Thus, each sonar grid cell contains a mean depth and a standard deviation of the depth that represent the best estimate of the bottom topography from an accumulation of all the sonar beams that correspond to that cell. The FLS measurement algorithm is given in Figure 3-3. The inputs to this algorithm are the mean of the noise in an individual sonar beam (assumed to be zero), the standard deviation of the noise in an individual sonar beam (assumed to be known), the sonar receiver location in the North-East-Down coordinate frame, the terrain database of true depths in terms of terrain cells of a given size, and a scale factor used to divide up the terrain cells into smaller cells in order to simulate sonar outputs at better resolution, as well as the *x*-*y* position and the yaw of the UUV at the time of the measurement.

```
algorithm FLS-MEAS( $\mu, \sigma, rcvr\_loc, scale\_factor, terrain, x_{uuv}, y_{uuv}, \phi$ )
    for $b \leftarrow 0$ to 100        /* for each of the 101 sonar beams */
        $beam\_unit \leftarrow$ unit vector for beam $b$
        $\{x_{terrain}, y_{terrain}, depth_{true}\} \leftarrow env\_depth(rcvr\_loc, beam\_unit)$
        $no\_of\_cells \leftarrow 0$
        /* Calculate sonar cell $s$ (in terms of $index_x, index_y, row, col$ ). */
        $index_x \leftarrow$ ENV-X-INDEX( $x_{terrain}$ )  /* Calculate $x$-index in NED coordinate frame. */
        $index_y \leftarrow$ ENV-Y-INDEX( $y_{terrain}$ )   /* Calculate $y$-index in NED coordinate frame. */
        $border_{south} \leftarrow$ ENV-X-POS( $index_x$ ) /* Calculate position of west side of sonar cell. */
        $border_{west} \leftarrow$ ENV-Y-POS( $index_y$ ) /* Calculate position of south side of sonar cell. */
        $dx_{scaled} \leftarrow \frac{terrain.dx}{scale\_factor}$
        $dy_{scaled} \leftarrow \frac{terrain.dy}{scale\_factor}$
        $row \leftarrow \frac{x - border_{south}}{scale\_factor}$
        $col \leftarrow \frac{y - border_{west}}{scale\_factor}$
        $flag \leftarrow 0$   /* Signify that sonar cell has not been found in list (yet). */
        $s \leftarrow 0$       /* Start with first sonar cell in the list. */
        while ($flag = 0$ & $s < no\_of\_cells$)
                $s \leftarrow s+1$  /* Search for the correct sonar cell in the list. */
```

```
            /* If sonar cell is found, then update its average depth and hit count. */
17              if ( $index_x = fls[s].x\_index$ & $index_y = fls[s].y\_index$
        & $row = fls[s].row$ & $col = fls[s].col$ )
18                  $flag \leftarrow 1$ /* Signify that correct sonar cell is found. */
19                  $depth_{avg} \leftarrow \frac{depth_{true} + fls[s].avg\_depth * fls[s].hit\_count}{fls[s].hit\_count + 1}$
20                  $fls[s].hit\_count \leftarrow fls[s].hit\_count + 1$
21              end if
22          end while
            /* If sonar cell is not already in the list, then add it to the list. */
23          if ($flag = 0$)
24              $no\_of\_cells \leftarrow no\_of\_cells + 1$
25              $fls[no\_of\_cells].x\_index \leftarrow index_x$
26              $fls[no\_of\_cells].y\_index \leftarrow index_y$
27              $fls[no\_of\_cells].row \leftarrow row$
28              $fls[no\_of\_cells].col \leftarrow col$
29              $fls[no\_of\_cells].avg\_depth \leftarrow depth_{true}$
30              $fls[no\_of\_cells].hit\_count \leftarrow 1$
31          end if
32          for $s \leftarrow 0$ to ($no\_of\_cells - 1$)   /* for each sonar cell s in the list */
33              $depth_{sonarcell} \leftarrow fls[s].avg\_depth$
34              $n \leftarrow fls[s].hit\_count$
                /* Generate the mean of the noise from the distribution for the mean. */
35              $mean \leftarrow \mu$
36              $stdv \leftarrow \sqrt{\frac{1}{n}}\sigma$
37              $noise_{mean} \leftarrow \mathrm{RANDOM}(mean, stdv)$
                /* Generate standard deviation of noise from distribution for the variance */
38              $mean \leftarrow \sigma^2$
39              $stdv \leftarrow \sqrt{\frac{2}{n-1}}\sigma^2$
40              $noise_{\mathrm{var}} \leftarrow \mathrm{RANDOM}(mean, stdv)$
```

```
41        $noise_{stdv} \leftarrow \sqrt{noise_{\mathrm{var}}}$
42        $fls[s].mean\_depth \leftarrow depth_{sonarcell} + noise_{mean}$
43        $fls[s].stdv\_depth \leftarrow noise_{stdv}$
44        $border_{south} \leftarrow$ ENV-X-POS( $fls[s].x\_index$ )
45        $border_{west} \leftarrow$ ENV-Y-POS( $fls[s].y\_index$ )
46        $dx_{scaled} \leftarrow \frac{terrain.dx}{scale\_factor}$
47        $dy_{scaled} \leftarrow \frac{terrain.dy}{scale\_factor}$
48        $row_{terrain} \leftarrow dx_{scaled} * fls[s].row$
49        $col_{terrain} \leftarrow dy_{scaled} * fls[s].col$
50        $x_{terrain} \leftarrow border_{south} + row_{terrain}$
51        $y_{terrain} \leftarrow border_{west} + col_{terrain}$
52        $\Delta x \leftarrow x_{terrain} - x_{uuv}$
53        $\Delta y \leftarrow y_{terrain} - y_{uuv}$
54        $\theta = \tan^{-1}(\frac{y}{x})$
55        $fls[s].bearing \leftarrow \theta - \phi$
56        $fls[s].range \leftarrow \sqrt{\Delta x^2 + \Delta y^2}$
57    end for
58  return
```

Figure 3-3: Algorithm for a FLS measurement

3.6 Passive Sonar Simulation

Since the UUV simulation does not include a model of the ISR sensors which will be able to give a measure of the range to a target, we will instead use and augment the existing passive sonar model to give measurements of range in addition to bearing. Since the mapping system requires a measure of the range to the target, we are assuming that this range comes from the passive sonar model, for the purposes of this thesis.

As with the existing active forward-looking sonar simulation, the existing passive sonar simulation also does not model noise. We will expand the passive sonar model to include a model of the uncertainty in a passive sonar measurement. We will simulate noisy bearing and range measurements, rather than explicitly calculating a mean and standard deviation for each passive sonar measurement.

3.6.1 Passive Sonar Model

The passive sonar is modeled as an omni-directional sensor that can detect any of the passive targets that have been "activated" in the simulated environment. Thus, the simplistic passive sonar model assumes detection of a target at any bearing between 0 and 360 degrees and at any range. The targets are modeled as single points moving with constant velocity and independently of the UUV. With knowledge of the exact location and velocity of a target, the passive sonar simply outputs a noiseless measure of range and bearing to the target.

3.6.2 Modeling Uncertainty in a Passive Sonar Measurement

We assume that the bearing noise is white Gaussian noise with variance σ_B^2. Similarly, we assume that the range noise is white Gaussian noise with variance σ_R^2, where the distribution of the range is dependent on the distribution of the bearing and the true range:

$$\sigma_R = r * \tan(\sigma_B) \tag{3.1}$$

The dependence on the true range *r* is meant to reflect the likelihood that a measurement of a target closer to the passive sonar sensor would probably be more accurate than a measurement of a target farther from the passive sonar sensor. The dependence on the distribution of the variance is meant to reflect the likelihood that a bearing measurement with high uncertainty would probably result in a range measurement with high uncertainty, and vice versa.

The passive sonar measurement algorithm is given in Figure 3-4. The inputs to this algorithm are the mean of the bearing noise to a target (assumed to be zero), the standard deviation of the bearing noise to a target (assumed to be known), as well as the *x-y* position and the yaw of the UUV at the time of the measurement. The value of the bearing noise for each passive sonar measurement is a randomly generated value from the Gaussian bearing distribution (line 5). The value of the range noise for each passive sonar measurement is a randomly generated value from the Gaussian range distribution (line 8). The value for the bearing is the bearing noise added to the true bearing (line 9). The value for the range is the range noise added to the true range (line 10).

```
algorithm PAS-SONAR-MEAS ($\mu_\beta, \sigma_\beta, x_{uuv}, y_{uuv}, \phi$)
1  for each active target $t$
2      $r_{true} \leftarrow \sqrt{x_{uuv}^2 + y_{uuv}^2}$
3      $\theta = \tan^{-1}(\frac{y_{uuv}}{x_{uuv}})$
4      $\beta_{true} \leftarrow \theta - \phi$
5      $\beta_{noise} \leftarrow$ RANDOM($\mu_\beta, \sigma_\beta$)
6      $\mu_r \leftarrow 0$
7      $\sigma_r \leftarrow r_{true} * \tan(\sigma_\beta)$
8      $r_{noise} \leftarrow$ RANDOM($\mu_r, \sigma_r$)
9      $pas[t].bearing \leftarrow \beta_{true} + \beta_{noise}$
10     $pas[t].range \leftarrow r_{true} + r_{noise}$
11 end for
12 return
```

Figure 3-4: Algorithm for a passive sonar measurement

3.7 Summary

This chapter has presented a model for the uncertainty in an active sonar measurement and for the uncertainty in a passive sonar measurement. We have taken a probabilistic approach and modeled the sonar noise as white Gaussian noise. For the active forward-looking sonar simulation, we assume that we know the mean and standard deviation for an individual sonar beam. We group the individual beams into sonar cells, and we calculate the distribution of the sample mean based on the number of individual sonar beams that comprise the sonar cell. We generate the mean of the measurement noise for a sonar cell from the Gaussian distribution for the sample mean. We add the mean of the noise to the true depth to get the simulated noisy measurement. In a similar manner, we calculate the distribution of the sample variance based on the number of individual sonar beams that comprise the sonar cell. We generate the variance of the measurement noise for a sonar cell from the Gaussian distribution for the sample variance. The standard deviation of the measurement is equal to the standard deviation of the noise.

For the passive sonar simulation, we assume that we know the standard deviation of the bearing to a target, and we assume that the mean of the bearing noise distribution is zero. We generate a value for the bearing noise from this zero-mean Gaussian distribution. The simulated noisy measurement of the bearing is equal to the value of the bearing noise added to the true bearing. We use the standard deviation of the bearing noise distribution and the true range to calculate the standard deviation of the distribution of the range noise, and we assume that the mean of the range noise distribution is zero. We generate a value for the range noise from this Gaussian distribution. The value for the range noise added to the true range is equal to the simulated range measurement.

[This Page Intentionally Left Blank]

4 Static Obstacle Map

The mapping system consists of a static obstacle map and a dynamic obstacle map. This chapter presents the design, representation, and implementation of the static obstacle map. The static obstacle map stores information about stationary obstacles encountered with the UUV's sensors in the UUV's underwater environment. Although terrain is the primary static obstacle currently considered in the static obstacle map, other static obstacles such as bridge supports could also be incorporated into the static obstacle map.

4.1 Design of the Static Obstacle Map

The static obstacle map consists of two layers: a *local map* and a *global map*. The combination of two separate maps allows each map to use a representation that is best suited to store the data of interest at the desired level of detail. The local map will store data at a higher resolution for the immediate mission area (which is typically a few square miles), while the global map will compactly store data at a lower resolution for the entire mission space (which is hundreds or thousands of square miles). This two-layer approach is similar to that used in Sammon [33] and Steiner [35].

The combination of a local map and global map allows the information in the immediate mission area of interest to be kept separate from the data in the global map. The local map will be small enough so that it can be stored in the RAM on-board the vehicle and so that it can be accessed quickly for critical mission objectives. The global map can be stored on disk and can be retrieved as a permanent record of the entire mission after the mission has ended. The data in the local map will be used to update the global map after a mission activity has ended and when the vehicle is about to transition to another activity. Consequently, the time taken to update the global map is not a significant design concern. The major design motivation is the need for a compact and efficient representation of the entire mission space that preserves a high level of detail in the immediate mission area.

4.2 Inputs to the Static Obstacle Map

The static obstacle map can be loaded with *a priori* data before the start of the mission. During the mission, the inputs to the static obstacle map at each sonar ping are the measurements and from the active FLS and the position of the vehicle (see Figure 4-1 and Figure 4-2). Each FLS measurement consists of:

r	- range to the obstacle from the vehicle (in the North-East plane)
β	- bearing to the obstacle from the vehicle (measured as degrees from the vehicle to the obstacle, in the North-East plane)
μ_{depth}	- mean of the depth of the static obstacle
σ_{depth}	- standard deviation of the depth of the static obstacle

The position of the vehicle at the time of the sonar ping consists of:

ϕ	- heading of the vehicle (measured as degrees East of North)
lat_{uuv}	- latitude of vehicle
lon_{uuv}	- longitude of vehicle

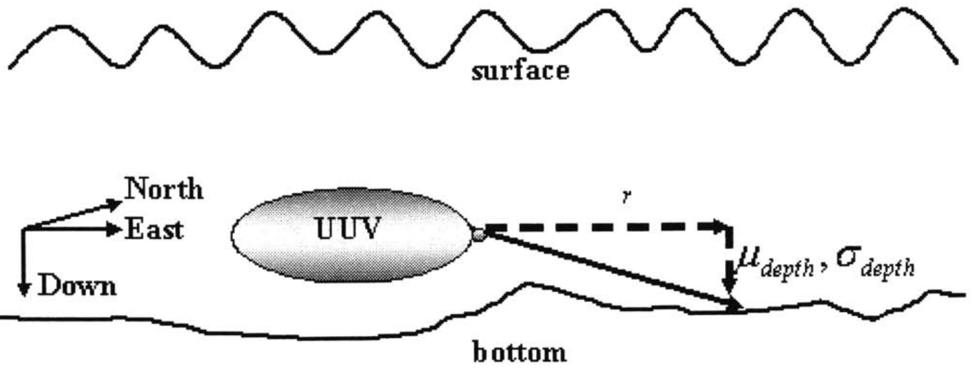

Figure 4-1: Inputs to the static obstacle map, in the North-East-Down coordinate frame

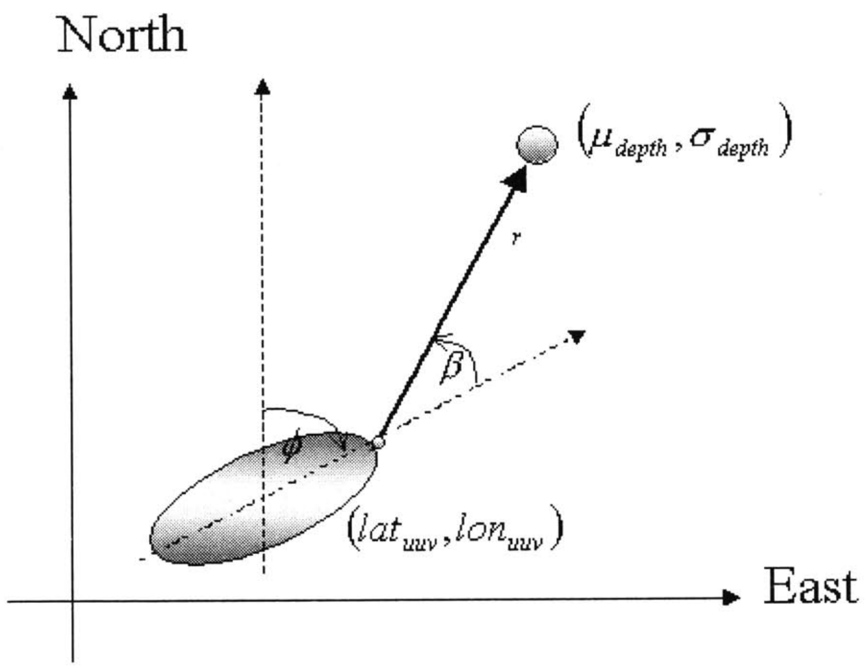

Figure 4-2: Inputs to the static obstacle map, in the North-East plane of the UUV

4.3 Design of the Local Map

The local map is represented as a uniform height field grid of the two-dimensional *x-y* surface of the Earth in the observation area of interest (see Figure 4-3). The surface is assumed to be flat in the observation area.[4] Each grid cell corresponds to a finite, equal-size, square area in the grid. The "height field" of each cell corresponds to the depth beneath that area, in terms of the mean and standard deviation of the depth in the area defined by the cell.

[4] This flat-Earth assumption is valid because the size of the Observation Area (and thus the size of the local map) is only a few square miles.

The static obstacles, such as the terrain, can be represented in two dimensions, since the vehicle is only concerned about passing over or around the static obstacles, as opposed to under or through them. Thus, the map only needs to store the mean and standard deviation of the minimum depth of the detected obstacles as the sufficient information for describing these obstacles in the underwater environment. The standard deviation is meant to model both the uncertainty about the depth measurements and the variability in the depth over the cell.

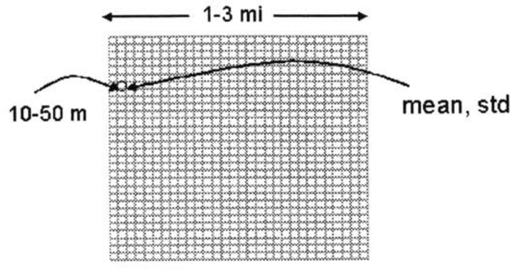

Figure 4-3: Local map represented as a uniform height field grid

Although a known disadvantage to grids is their large memory requirement, the memory required by the local map will be moderate. The size of the immediate mission area is typically a few square miles, and the resolution will be low (i.e., the size of each local cell will be large). For example, a local map that is 3 miles on each side with a cell resolution of 50m will have 97 cells per side, and approximately 10,000 cells in total. A local map that is 1 mile on each side with a cell resolution of 10m will have 161 cells per side, and approximately 26,000 cells in total. With the increases in computational speed and available memory, these numbers of cells are reasonable and will result in moderate memory usage.

Since the cells must be accessed many times because of the updating, filtering, and smoothing of the cell data, it is important to have quick access to the cells. An advantage to using uniform grids is that the grid cell structure allows fast, real-time updates to the local map during a mission, since each local cell can be looked up in constant time by its index into the grid cell array. If a hierarchical structure such as the quadtree were used, there would be a tradeoff between speed and memory. Either time linear in the height of the tree would be required for each search through the tree to find each neighboring cell, or a large amount of memory would be required to store the neighbors of each cell. Real-time performance is an important issue since a practical system cannot have a stop-and-go strategy, where the vehicle waits until the processing has finished to find the answer. Because of the high data rates in such a process, the efficiency was a major factor in the choice of representation of the local map. The local map has been designed to minimize cell access time.

4.3.1 Implementation of the Local Map

The local map is implemented as a one-dimensional fixed-length array (that represents this two-dimensional grid), where each element corresponds to an equal-size grid cell, which is called a *local cell* (see Figure 4-4). Each local cell stores the estimate of the mean depth and the standard deviation of the depth for this area. Furthermore, the local cell stores bits to indicate whether or not the local cell has been updated.

LOCAL MAP CELL

```
typedef struct localMapCell {
   double mean;          /* estimate of the mean of the depth at this local cell */
   double std;           /* estimate of the standard deviation of depth at this local cell */
   int updated;          /* {1 = has been updated,  = 0 has not} */
   int directUpdated;    /* {1 = has been updated directly from sonar, 0 = has not} */
   int smoothUpdated;  /*{1 = has been updated from neighbor smoothing, 0 = has not}*/
} localMapCell;
```

Figure 4-4: Structure of a local cell in the local map

For simplicity, the location of the local cell is defined to be the location of the southwest corner of the local cell. Without loss of generality, any point within the local cell, such as the midpoint, can be used to specify the location of the local cell, as long as the same point is consistently used to describe the location of every local cell. The local cell does not need to explicitly store an index number, since the cell can be indexed implicitly by its offset from the start of the local grid array. The local cell does not need to store its coordinates because the flat-Earth assumption allows the index of the local cell to be combined easily with the coordinates of a reference cell to calculate the coordinates of any cell in the local map.

4.3.2 Initialization of the Local Map

Upon the start of the vehicle's mission, space for the local map is allocated, and the local map is initialized. If the global map has already been initialized and loaded with *a priori* data, the local map will copy these values into the corresponding local cells. If the global map does not exist yet, then the local map will initialize the mean and standard deviation values for each local cell with default initialization values.

The local map can also be initialized at the start of each new mission activity. In this case, space for the local map is not allocated again. The local map will just load the data from the section of the global map that corresponds to the area spanned by the local map.

4.3.3 Updating the Local Map

At each ping of the active forward-looking sonar, the sonar returns a collection of sonar measurements that will correspond to a subset of the local cells in the local map. For each sonar measurement, the range and bearing from the vehicle to the static obstacle are combined with the vehicle's location at the time of the sonar ping to determine the obstacle's absolute location. This absolute location is combined with the position of the local map to find the local cell in the local map to which this obstacle location corresponds. Figure 4-5 shows the relationship between the position of the vehicle and the position of the obstacle in a North-East-Down (NED) coordinate frame.

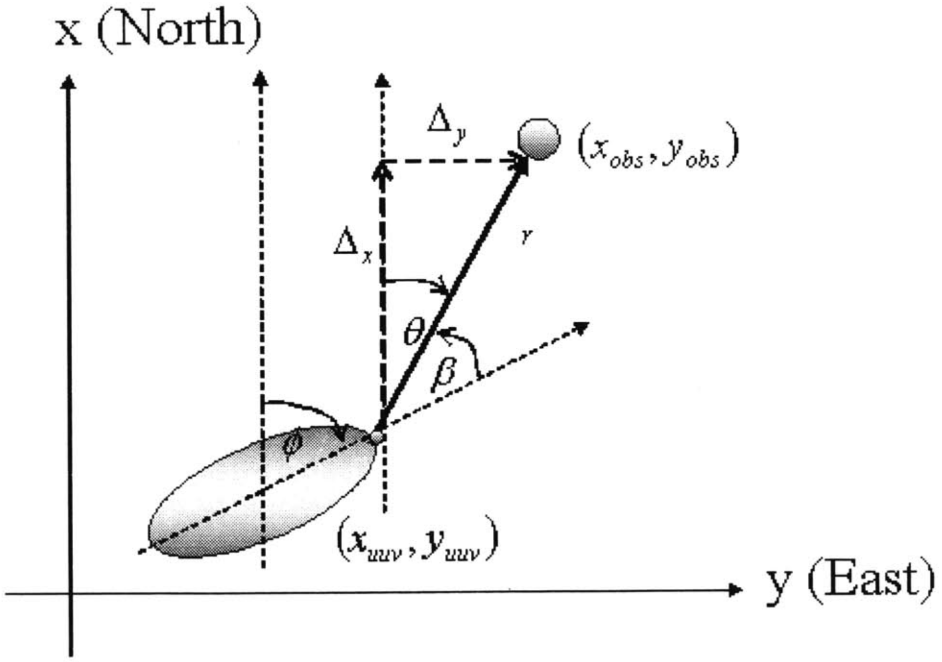

Figure 4-5: Relationship between the position of the vehicle and the position of the obstacle in a North-East-Down coordinate frame

The angle of the obstacle, θ, (measured as degrees East of North in the North-East-Down coordinate system) is calculated as the sum of the heading of the vehicle, ϕ, and the bearing to the obstacle from the vehicle, β.

$$\theta = \phi + \beta \tag{4.1}$$

The components of the difference in the position of the obstacle relative to the vehicle, Δ_x and Δ_y, are calculated as

$$\begin{aligned} \Delta_x &= r\cos(\theta) \\ \Delta_y &= r\sin(\theta) \end{aligned} \tag{4.2}$$

where r is the range from the vehicle to the obstacle. lat_{ref} and lon_{ref} are the latitude and longitude, respectively, that define the origin of the local map. Using the flat-Earth assumption, the x-y position of the vehicle in the local map is calculated as

$$\begin{aligned} x_{uuv} &= (lat_{uuv} - lat_{ref})R_{Earth} \\ y_{uuv} &= (lon_{uuv} - lon_{ref})R_{Earth}\cos(lat_{ref}) \end{aligned} \tag{4.3}$$

where R_{Earth} is the equatorial radius of the Earth. The x-y position of the obstacle in the local map is calculated as

$$\begin{aligned} x_{obs} &= x_{uuv} + \Delta_x \\ y_{obs} &= y_{uuv} + \Delta_y \end{aligned} \tag{4.4}$$

The x-component of the index, i, the y-component of the index, j, and the index of the local cell that corresponds to the position of the obstacle are determined by

$$\begin{aligned} i &= \left\lfloor \frac{x_{obs}}{res_{lcell}} \right\rfloor \\ j &= \left\lfloor \frac{y_{obs}}{res_{lcell}} \right\rfloor \\ index &= j + i * num_{y\text{-dim}} \end{aligned} \tag{4.5}$$

where res_{lcell} is the resolution (or size) of the local cell and $num_{y\text{-}dim}$ is the number of local cells that make up the local map in the y-direction. The local cell that corresponds to this index is simply looked up in the grid array:

$$lcell = grid[index] \tag{4.6}$$

After the correct local cell has been found, the input data is ensured to be within some range of values. This is described in the next section. If the data is valid, the local cell will be updated with the corresponding sonar measurement, as described in § 4.3.3.2. In addition, the updated local cell will be used to *smooth* the surrounding cells, as described in § 4.3.3.3.

4.3.3.1 Thresholding the Input Data

Range readings above a certain maximum R_{max} are discarded, since sonar can return bad measurements. The range returns near R_{max} are likely due to specular reflections that cause inaccurate sonar returns. The value of R_{max} depends on the expected maximum range of the sonar. Similarly, range returns below a certain minimum R_{min} are discarded. It is logical to choose R_{min} to be 0, because any negative range values would obviously be due to a problem in the sonar. Range measurements with the range (R_{min}, R_{max}) are considered to be valid sonar measurements.

4.3.3.2 Updating the Local Cells

After each valid sonar measurement is matched to its corresponding local cell in the local map, the previous depth information in the local cell must be combined with the new depth information from the sonar measurement. Since the local cell stores the mean and standard deviation of the depth, these two values completely specify a Gaussian density for the depth of the static obstacle at that local cell location. Furthermore, the input sonar measurement contains new values for the mean and standard deviation for the depth, so this measurement also specifies a Gaussian distribution. Now the problem of updating the previous information in the local cell with the new information reduces to the problem of combining two Gaussian densities.

The theory of maximum likelihood estimation is used to combine the previous depth information in the cell with the current measurement of depth from the sonar. Maximum likelihood estimation determines the "best" estimate based on the criterion of minimizing the variance of the error between the estimate and the true value [41].

Let z_1 correspond to the previously accumulated measurements ($\ldots$, z_{-2}, z_{-1}, z_0) of the depth in the local cell, such that $z_1 = f(\ldots, z_{-2}, z_{-1}, z_0)$ is a combination of the previous measurements, we can consider z_1 as a measurement itself. Let z_2 correspond to the new measurement from the sonar. Due to the inherent noise associated with the sonar system, the measurements z_1 and z_2 are uncertain. Let σ_1^2 be the variance associated with measurement z_1, and let σ_2^2 be the variance associated with measurement z_2. The conditional probability density of x, the true depth, conditioned on the observed value of the measurement being z_1, is

$$f_{X|Z_1}(x \mid z_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left[-\frac{(x-z_1)^2}{2\sigma_1^2}\right] \tag{4.7}$$

This is a Gaussian density with mean z_1 and variance σ_1^2. Based on this conditional density, the maximum likelihood estimate $\hat{x}_1$ of the true depth x, given the measurement z_1, is

$$\hat{x}_1 = z_1 = \mu_1 \tag{4.8}$$

and the variance of the error in the estimate is

$$\sigma_{\hat{x}_1}^2 = \sigma_1^2 \tag{4.9}$$

Similarly, the conditional density of the true depth x, conditioned on the observed value of the measurement being z_2 is

$$f_{X|Z_2}(x \mid z_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left[-\frac{(x-z_2)^2}{2\sigma_2^2}\right] \tag{4.10}$$

This is a Gaussian density with mean z_2 and variance σ_2^2. Based on this conditional density, the maximum likelihood estimate $\hat{x}_2$ of the true depth x, given the measurement z_2, is

$$\hat{x}_2 = z_2 = \mu_2 \tag{4.11}$$

and the variance of the error in the estimate is

$$\sigma_{\hat{x}_2}^2 = \sigma_2^2 \tag{4.12}$$

At this point, two measurements are available for estimating the true depth. Maximum likelihood estimation dictates that the conditional density of the true depth x, given measurements z_1 and z_2, is a Gaussian density

$$f_{X|Z_1,Z_2}(x \mid z_1, z_2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right] \tag{4.13}$$

with mean μ and variance σ^2 as defined below.

$$\mu = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2 \tag{4.14}$$

$$\sigma^2 = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\sigma_2^2 \tag{4.15}$$

Given this density, the maximum likelihood estimate is

$$\hat{x} = \mu \tag{4.16}$$

with an associated error variance σ^2. This estimate is also the weighted least squares estimate and the linear estimate whose variance is less than that of any other linear unbiased estimate. Thus, it is considered the "best" estimate for the assumed criteria.

We will define k to be a weighting factor as follows:

$$k = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{4.17}$$

k ranges between 0 and 1 and will decrease as the input variance, σ_2^2, increases. Thus, $(1-k)$ will increase as the input variance, σ_2^2, increases. Figure 4-6 shows how k varies with the input variance σ_2^2, where σ_1^2 is set to 0.5.

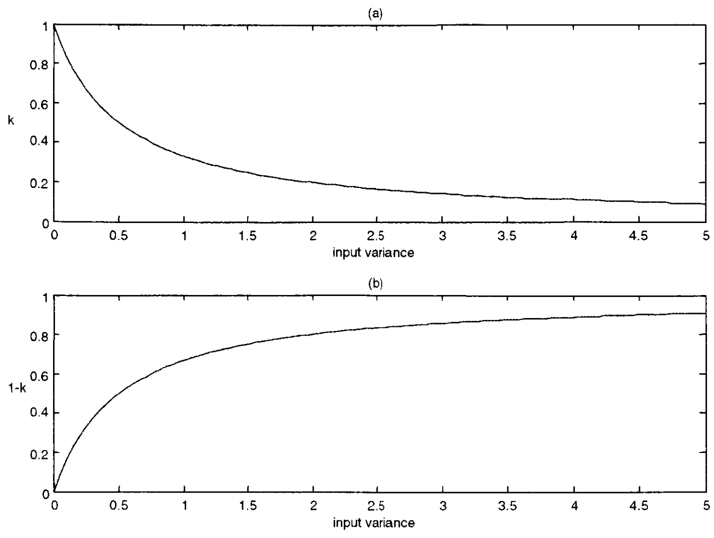

Figure 4-6: Relationship between weighting factor and input variance

(a) k decreases as σ_2^2 increases, (b) and consequently, (1-k) increases as σ_2^2 increases (Note that σ_1^2 is set to 0.5)

Combining equations (4.14), (4.16), and (4.17), we have:

$$\hat{x} = \mu = (1-k)\mu_1 + k\mu_2 \tag{4.18}$$

This can be rewritten as:

$$\hat{x} = \mu = \mu_1 + k(\mu_2 - \mu_1) = \hat{x}_1 + k(z_2 - \hat{x}_1) \tag{4.19}$$

Equation (4.19) says that the optimal estimate $\hat{x}$ is equal to the best estimate of its value before measurement z_2 is taken, $\hat{x}_1$, plus a correction term of an optimal weighting value times the difference between measurement z_2 and the best estimate before it is taken, $\hat{x}_1$.

If the input measurement has a large variance (i.e., σ_2^2 is large, such that k is small), then equation (4.18) will give less weight to the input measurement, $z_2 = \mu_2$, and more weight to the existing measurements, $z_1 = \mu_1$. Thus, the updated mean will be dominated by the existing measurements in the cell. However, if the input measurement has a small variance (i.e., σ_2^2 is small, such that k is large), then equation (4.18) will give more weight to the input measurement, $z_2 = \mu_2$, and less weight to the existing measurements, $z_1 = \mu_1$. In this case, the updated mean will be dominated by the new input measurement. If σ_2^2 were equal to σ_1^2 (i.e., $k = 0.5$), then equation

(4.18) says that the optimal estimate of the mean is simply the average of the two values, as would be expected since the measurements have equal accuracy.

To help understand the update equation for the variance (4.15), we can rewrite it as:

$$\sigma^2 = k\sigma_2^2 = \frac{(1-k)\sigma_1^2 + k\sigma_2^2}{2} = \sigma_1^2 \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{4.20}$$

As σ_2^2 increases (i.e., k approaches 0), the updated variance σ^2 approaches σ_1^2. On the other hand, as σ_2^2 decreases (i.e., k approaches 1), the updated variance σ^2 approaches 0. In the case where σ_2^2 is equal to σ_1^2, the updated variance σ^2 is equal to $0.5\sigma_2^2$ (see Figure 4-7, where σ_1^2 is set to 0.5). Thus, equation (4.20) results in a value for the updated variance σ^2 that is less than the input variance σ_1^2 (as long as σ_2^2 is not infinity). This property is based on the idea that each new measurement provides more information, so that the accuracy of the estimate increases with each new measurement. While this property works well when the series of measurements is fairly consistent, this property is not desirable in all practical situations, such as when the sonar gives bad, or inconsistent, measurements of the variance. If the sonar measurements are not trustworthy, then we want to decrease our confidence in the sonar measurements and we want to increase our resulting uncertainty in the updated estimate.

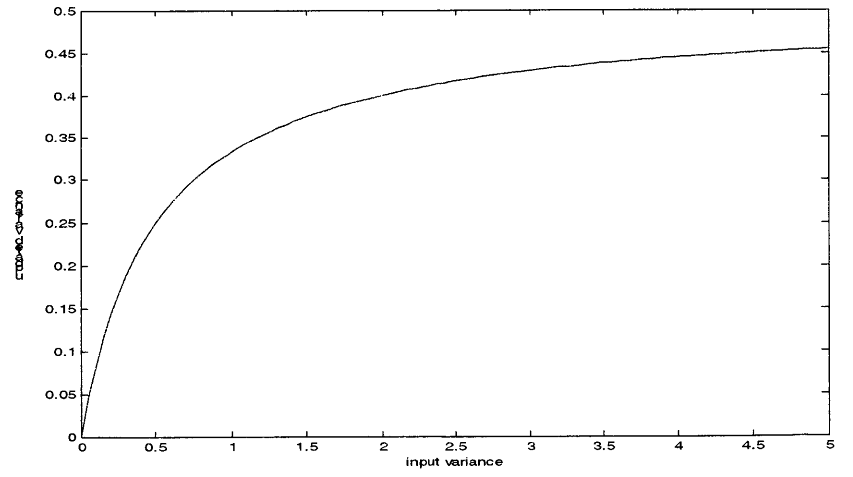

Figure 4-7: Relationship between input variance and updated variance (Note that σ_1^2 is set to 0.5)

Equation (4.20) does not work well for inconsistent measurements because it does not reflect the movement of the mean between the measurements. Consider the pathological example where the sonar system reports multiple measurements with variance equal to 0 but with different means. The updated variance should reflect that there is some inconsistency and ambiguity in the value of the mean. However, if the variance were updated with equation (4.20) as the variance update equation, the updated variance would be equal to 0. Consider an alternative choice for the update equation:

$$\sigma^2 = (1-k)\sigma_1^2 + k(\mu_1 - \mu_2)^2 \tag{4.21}$$

This equation would give a non-zero value for the updated variance in the example above, and this is desirable. Capturing the movement of the mean between measurements is an important property because the variance should reflect not only the confidence reported for each measurement but also the consistency between measurements. Since our criterion is to reflect the movement of the mean in addition to minimizing the variance between the truth and the estimate, we will use equation (4.21) instead of equation (4.20) as the equation for updating the variance. We will use equation (4.18) as the equation for updating the mean.

4.3.3.3 Smoothing Neighboring Cells

Since terrain is likely to change gradually and to lack large, sharp fluctuations within a very small area, it is likely that the true depth is similar within a small area. Thus, the values of the mean depth should be fairly similar in the cells surrounding the updated cell. Due to the inherent noise in the sonar measurements and due to the varying degree of accuracy in the a priori map, there may be large differences in the depth values over a small area in the map. Thus, it is desirable to *smooth* the values of the mean depth in the *neighborhood* of the updated cell to help even out the possible large deviations within a small area and to provide a more realistic prediction of what the vehicle will expect when it encounters those cells in the future. The smoothing also indirectly reflects some of the uncertainty that is associated with the exact position of the UUV.

The "small area" surrounding the local cell of interest is defined in terms of the distance d in each direction from the newly updated local cell of interest to the boundary of the neighborhood region, and it takes the shape of a symmetric bounding box centered on the local cell of interest. The number of neighboring cells n that will be smoothed in each direction (North, East, South, West) around the local cell of interest with resolution (or size) res_{lcell} is given by:

$$n = \left\lceil \frac{d}{res_{lcell}} \right\rceil \tag{4.22}$$

The indices of the outermost neighboring cells at the boundaries are calculated as:

$$\begin{aligned} index_{North} &= i + n \\ index_{East} &= j + n \\ index_{South} &= i - n \\ index_{West} &= j - n \end{aligned} \tag{4.23}$$

where i is the x-coordinate of the index of the updated local cell of interest, and j is the y-coordinate of the index. In addition, we ensure that these bounds all lie within the local map. The neighborhood includes all of the cells with indices that are within the bounds of this box.

The value in the newly updated local cell of interest is assumed to be more accurate than the value in the non-updated neighboring cells, and the mean depth value in the newly updated local cell will be used to smooth the existing value in each of the non-updated neighboring cells. The smoothing uses the same update equations that are used for updating a local cell directly with a sonar measurement, except that the weighting factor is calculated differently. The weight given to the information in the newly updated local cell is:

$$k_{smooth} = \frac{s\sigma_1^{\,2}}{\left(\sigma_1^{\,2} + \sigma_2^{\,2} + n_{ij}\right)} \tag{4.24}$$

s is a scale factor that can be set to a value in the range from 0 to 1, and it represents the amount of smoothing that should be done. n_{ij} is a function of the difference between the index of the local cell and the index of the neighboring cell. We can use any function of distance, such as Euclidian distance, but we choose to define n_{ij} as the maximum number of grid cells from the newly updated local cell to the neighboring cell of interest:

$$n_{ij} = \max\left(\left|j_{neighbor} - j_{lcell}\right|, \left|i_{neighbor} - i_{lcell}\right|\right) \tag{4.25}$$

where $j_{neighbor}$ is the y-component of the index of the neighboring cell, j_{lcell} is the y-component of the index of the local cell of interest, $i_{neighbor}$ is the x-component of the index of the neighboring cell, and i_{lcell} is the x-component of the index of the local cell of interest.

The addition of the n_{ij} term in the denominator of equation (4.24) effectively increases the variance of the variance of the input measurement, σ_I^2, and as a result, decreases the value of k_{smooth} and the confidence in the input measurement. The weighting factor, k_{smooth}, decreases as the distance from the updated local cell, n_{ij}, increases. Thus, each neighboring cell is updated by an amount that depends on its distance from the updated local cell. Closer neighbors are updated with a greater weight than farther neighbors. In the case where s equals 1 and n_{ij} equals 0, the weighting factor degenerates to the same weighting factor that is used to update a cell directly from a sonar measurement. This makes sense and is consistent, because in this case the "neighboring" cell is actually the same cell as the newly updated local cell. The case where s equals 1 corresponds to the maximum smoothing. The case where s equals 0 corresponds to the minimum smoothing (which is no smoothing at all, since k_{smooth} would be equal to 0), and the values of the neighboring cells remain unchanged. Since it is desirable to weight the smoothing updates less than a sonar update directly to that cell, it makes sense that s should be less than 1.

4.4 Design of the Global Map

The global map is represented as a quadtree structure (similar to the quadtree presented in [31]). Since natural terrain often consists of semi-flat areas, the hierarchical nature of the quadtree will allow areas that are of the same or similar depth to be grouped into larger cells, thereby minimizing the memory required to store the same amount of information. The use of a compact global map will allow the storage of the entire mission data, where the entire mission may be on the order of hundreds or even thousands of square nautical miles. This large area would be too great to store in a uniform grid.

The global map is capable of storing the mean depth and standard deviation with a lower amount of memory than that required by a uniform grid local map of the same size, in exchange for a lower level of detail in the values. The level of resolution of the global map can be chosen depending on the size of the entire space and on the level of detail necessary. The global map represents a general picture of the entire area for the vehicle, and it is sufficient to store the information at a lower resolution than the information in the local map. If the vehicle needs to survey a small area with high resolution, it can travel through the area and store its measurements in the local map with higher resolution. Figure 4-8 shows an extension of the quadtree idea to store non-binary data, such as depth values.

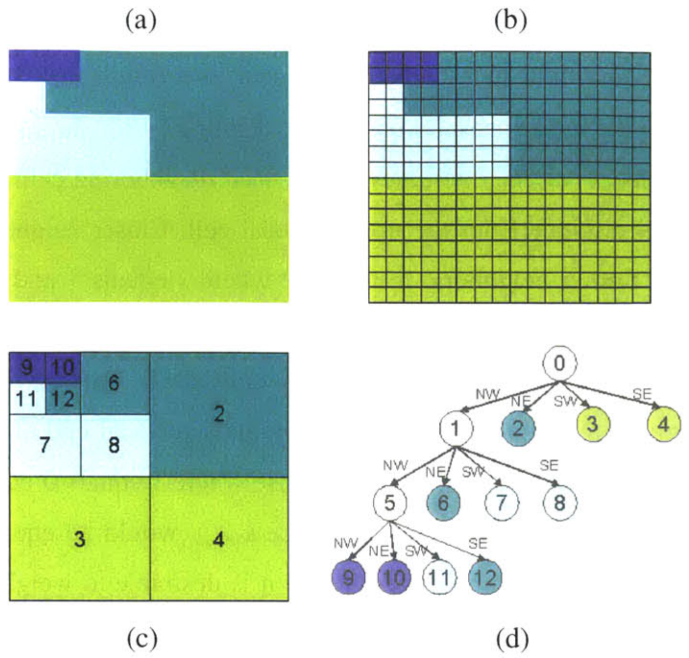

Figure 4-8: Extension of the quadtree to non-binary data

(a) Sample region, where the different colors represent different depth values, (b) its uniform grid representation as an array, (c) its quadtree representation in terms of maximal blocks, and (d) the corresponding quadtree data structure

4.4.1 Implementation of the Global Map

The global map is implemented as a one-dimensional fixed-length array, where each element corresponds to a global cell. Since each global cell has a variable size, there is no implicit order of the global cells in the array. Thus, each global cell stores an index number, its node type (leaf or parent), its location and size, a pointer to its parent node, and a pointer to each of its four children nodes, as well as the mean value of the depth and the standard deviation of the depth. The size of the array is fixed so that the global map can only use up to a maximum amount of memory. The index number corresponds to the index of the global cell in the array.

We must remember that the global map is representing the underwater terrain for a large area. Since the flat-Earth model is not valid for this large area, we must recognize that the Earth's surface is modeled as a spherical surface rather than a planar surface[5]. We can superimpose the planar quadtree structure onto the spherical surface. Figure 4-9 shows the superposition of the global map onto the spherical surface of the Earth. The shape of the resulting global map will be a quadrilateral with sides equal in distance and approximately perpendicular to each other. The global map will be an approximate square in the *lat-lon* geodetic coordinate frame, rather than an

[5] In reality, the Earth is an ellipsoid rather than a sphere, but a spherical-Earth model is assumed.

exact square, because of the issue of the projection onto a spherical surface (see Figure 4-10). The resulting global map better approximates an exact square as the size of the global map decreases. This is as expected, since the planar surface will better approximate the spherical surface as the size of the global map decreases.

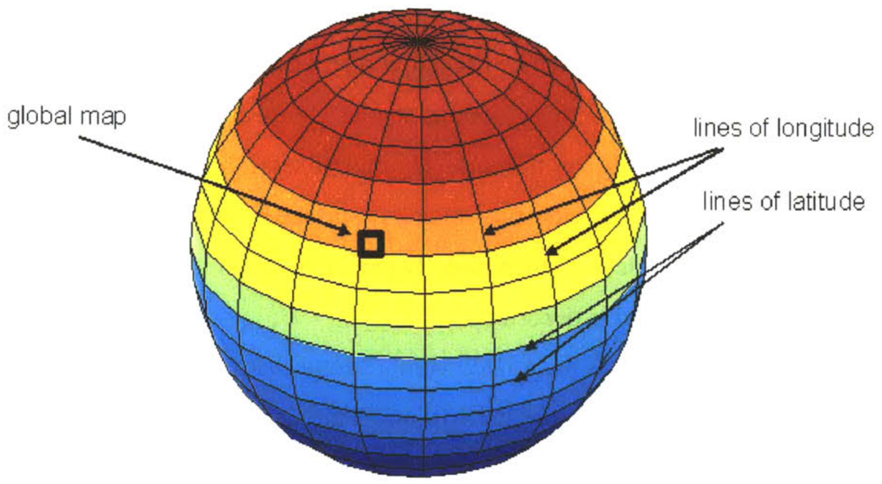

Figure 4-9: Global map superimposed on non-planar surface of the Earth

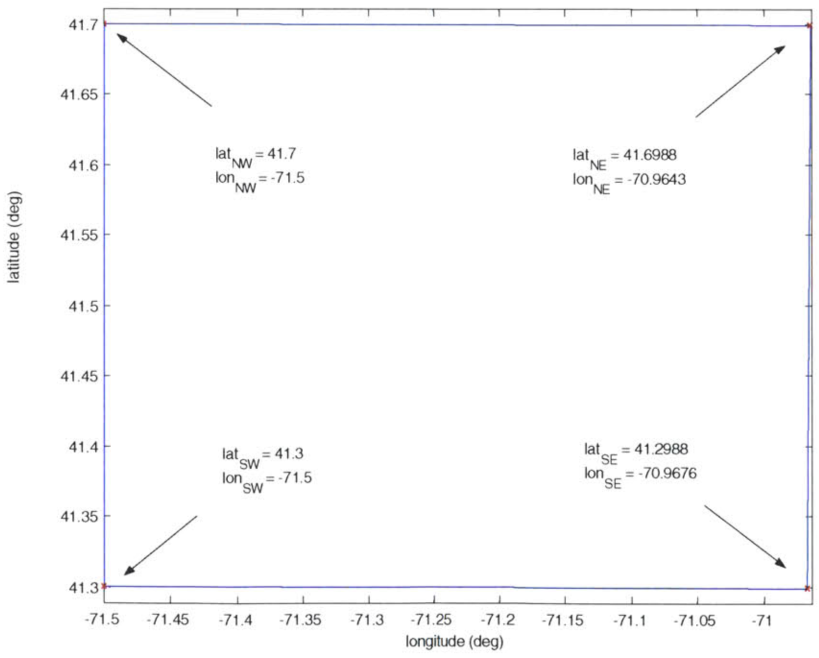

Figure 4-10: Global map as approximate square in the lat-lon geodetic coordinate frame

4.4.2 Location and Size of a Global Cell

Since the decomposition of space for a quadtree is not uniform, each cell must store its location and size. It is desirable to store the location in a form that is easy to maintain and that is directly applicable for the global map operations. When the quadtree is subdivided, the location and size of the parent cell must also be divided to form the location and size of the four new children cells.

Storing the latitude and longitude does not allow direct division of the location, since the lines of longitude are not parallel but come together at the North and South poles. Storing the position in an *x-y-z* Earth-fixed coordinate frame seems more useful, but upon further inspection, the question arises about how to subdivide the spherical surface of the earth into squares. The issue of how to subdivide the spherical surface directly can be avoided if the location is stored in a less conventional way. This method is described below.

Each corner of the global cell is associated with a vector that points from the center of the Earth to the corner in an Earth-fixed coordinate frame. Thus, there are four vectors: the *NW* corner vector, the *NE* corner vector, the *SW* corner vector, and the *SE* corner vector. We consider the planes that are formed by these vectors. The *north plane* is the plane containing the *NW* vector and the *NE* vector. Similarly, the *east plane* is the plane containing the *NE* vector and the *SE* vector, the *south plane* is the plane containing the *SW* vector and the *SE* vector, and the *west plane* is the plane containing the *SW* vector and the *NW* vector. The unit vector normal to the plane can represent the plane. Instead of storing the locations of the corners of the global cell, each global cell stores the four unit vectors that are normal to the four planes. Thus, the global cell stores the unit vector *N*[3] normal to the north plane, the unit vector *E*[3] normal to the east plane, the unit vector *S*[3] normal to the south plane, and the unit vector *W*[3] normal to the west plane. In addition, the sign of the unit vectors is maintained so that the unit vectors point inward to the center of the cell. Storing the location of the unit normal to each of the four planes will also indirectly give the size of the global cell. The structure of the global cell is given in Figure 4-11.

PARENT TYPE
```
typedef struct parentType {
   struct globalMapCell *childNW;          /* 1st child cell */
   struct globalMapCell *childNE;  /* 2nd child cell */
   struct globalMapCell *childSW;  /* 3rd child cell */
   struct globalMapCell *childSE;  /* 4th child cell */
} parentType;
```

LEAF TYPE
```
typedef struct leafType {
   double mean;                   /* mean of the depth */
```

```
  double std;                    /* standard deviation of the depth */
} leafType;
```

PARENT OR LEAF

```
typedef union parentOrLeaf {
  leafType l;                    /* leaf node*/
  parentType p;                  /* parent node*/
} parentOrLeaf;
```

GLOBAL MAP CELL

```
typedef struct globalMapCell {
  int id;                        /* unique identification number assigned to each gcell */
  int nodetype;                  /* {0 = parent, 1 = leaf} */
  double N[3];                   /* unit normal to north plane */
  double E[3];                   /* unit normal to east plane */
  double S[3];                   / unit normal to south plane */
  double W[3];                   /* unit normal to west plane */
  parentOrLeaf value;            /* either mean and std, or 4 children, depending on
                                    whether gcell is leaf or parent */
  struct globalMapCell *parent; /* pointer to the parent gcell */
} globalMapCell;
```

Figure 4-11: Structure of a global cell in the global map

4.4.3 Initialization of the Global Map

The global map can be initialized from a file at the start of the mission, or it can be initialized with default values at the time of the initialization of the local map. The file can contain detailed information (i.e., many global cells), or it can contain hardly any information at all (i.e., just the root global cell). The file also contains all of the parameters needed for the initialization of the global map, such as the latitude and longitude of the southwest corner and the northwest corner, the maximum number of global cells in the global map, and the minimum cell size.

Given the latitude and longitude of the southwest corner and the northwest corner of the global map as parameters, the latitude and longitude of the northeast corner and the southeast corner are calculated. The northeast latitude and longitude and the southeast latitude and longitude are initialized such that the resulting global map will have equal-distance sides, and where the sides will be approximately perpendicular to form an approximate square. Figure 4-12 shows an illustration of this initialization, and some notation is defined below. The algorithm for the initialization of the position of the global map is non-trivial and is described in Figure 4-13. The *NW* vector and *SW* vector are computed from the latitude and longitude of the given northwest corner and southwest corner, respectively, of the global map (lines 1-2). The vector *b* is the computed as the vector from the center of the Earth that bisects the *NW* and *SW* vectors (line

3). The line about which to rotate, r, and the angle to rotate, α, are computed (lines 4-6). NW is rotated about r by α to get NE (line 7), and SW is rotated about r by α to get SE (line 8). Finally, the normals to the planes are computed (lines 9-12).

Notation

lat_{NW}	- latitude of the northwest corner of the global map
lon_{NW}	- longitude of the northwest corner of the global map
lat_{SW}	- latitude of the southwest corner of the global map
lon_{SW}	- longitude of the southwest corner of the global map
NW	- vector from the center of the Earth to the northwest corner
NE	- vector from the center of the Earth to the northeast corner
SW	- vector from the center of the Earth to the southwest corner
SE	- vector from the center of the Earth to the southeast corner
ψ	- angle between NW and SW
b	- vector from the center of the Earth that bisects ψ
r	- unit vector rotation line (line about which to rotate NW to get NE, and about which to rotate to SW to get SE)
α	- rotation angle (angle about which to rotate NW to get NE, and about which to rotate SW to get SE)
N	- unit normal to the north plane, pointing inward to the center of the cell
E	- unit normal to the east plane, pointing inward to the center of the cell
S	- unit normal to the south plane, pointing inward to the center of the cell
W	- unit normal to the west plane, pointing inward to the center of the cell

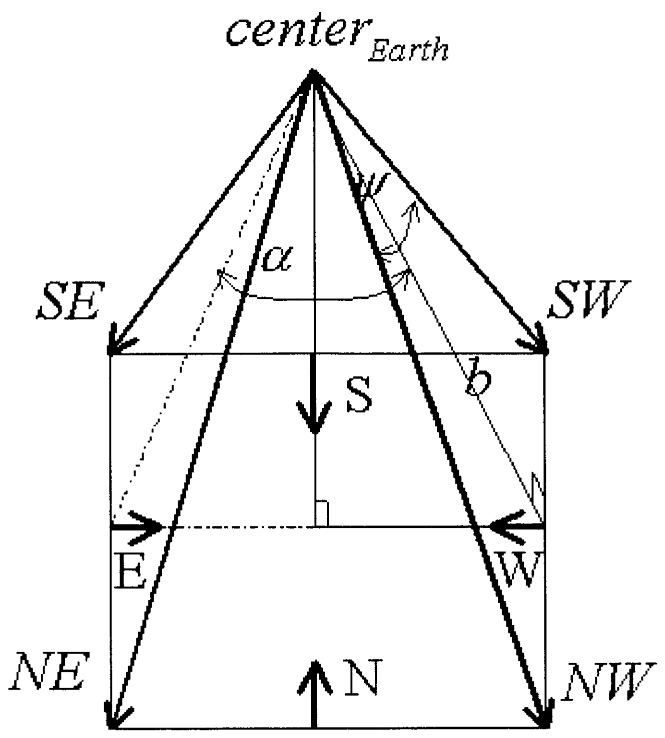

Figure 4-12: Initialization of the position of the global map

```
algorithm INITIALIZE-POSITION ( $lat_{NW}, lon_{NW}, lat_{SW}, lon_{SW}$ )
1  $NW \leftarrow$ EF-TO-GEODETIC $(lat_{NW}, lon_{NW})$
2  $SW \leftarrow$ EF-TO-GEODETIC $(lat_{SW}, lon_{SW})$
3  $b \leftarrow$ BISECTOR$(NW, SW)$
4  $r \leftarrow$ UNIT((UNIT$(NW \times SW)) \times b)$
5  $\psi \leftarrow \cos^{-1}(NW \bullet SW)$
6  $\alpha \leftarrow 2 * \sin^{-1}(\tan(\frac{\psi}{2}))$
7  $NE \leftarrow$ UNIT(ROTATE-POSITION$(NW, r, \alpha)) * R_{Earth}$
8  $SE \leftarrow$ UNIT(ROTATE-POSITION$(SW, r, \alpha)) * R_{Earth}$
9  $N \leftarrow$ UNIT$(NE \times NW)$
10 $E \leftarrow$ UNIT$(SE \times NE)$
11 $S \leftarrow$ UNIT$(SW \times SE)$
12 $W \leftarrow$ UNIT$(NW \times SW)$
13 return
```

Figure 4-13: Algorithm for initializing the position of the global map

After the position of the global map is initialized, the global map is initialized to contain at least one global cell – the root cell. The root cell is allocated and initialized to be a leaf node with no parent node. The position of the root global cell is initialized to have the same position as the global map itself, so that the root global cell represents the space of the entire global map. If the initialization is from a file, then the data in the file will be used to initialize as many global cells in the global map as there is global cell data in the file. In this way, the global map can support an *a priori* load of data, at any level of detail.

4.4.4 Updating the Global Map

The global map does not receive information directly from the sonar. Instead, the global map is updated with the information in the local map. For each cell in the local map that has been updated with a sonar measurement, the corresponding cell in the global map is found and updated. The midpoint of the updated local cell is used to direct the search for the corresponding global cell in the global map. Since the size of a corresponding global cell can be larger than the size of the local cell of interest, the global cell must be decomposed to the approximate size of the local cell before the updating can occur. The global cell that contains the midpoint of the local cell is broken down until a global cell of the approximate same size or of the minimum allowed size is reached. The process of finding the global map cell that corresponds to the local map cell is explained in more detail in § 4.4.4.1. After all of the updated local cells have been added to the global map, the global map is conglomerated to merge similar global cells. This is described in §4.4.4.4. Figure 4-14 shows an illustration of the process of updating the global map from the local map.

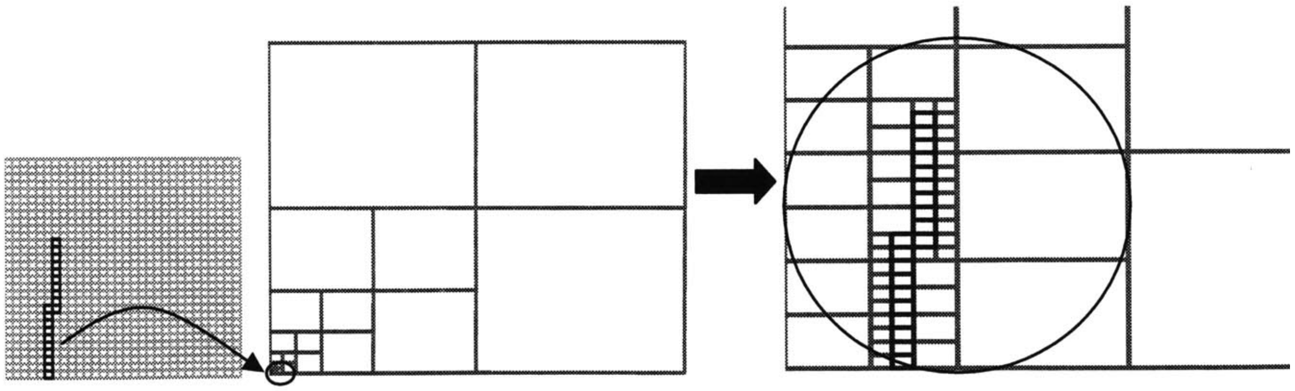

Figure 4-14: Updating the global map from the local map

4.4.4.1 Searching for a Local Cell in the Global Map

In order to find a cell in the global map, the search must continue down the relevant branches of the tree until the cell of interest is found. As a section of the global map is subdivided into smaller and smaller global cells, the spherical-surface global map approaches a planar local map. The global map cells should look almost like planar squares, and should approximately line up with the local cells.

The algorithm for this searching process is given in Figure 4-15. As the global cell is divided into smaller global cells and the size of the resulting global cell that contains the local cell of interest approaches the size of the local cell, the angle between *N*, the normal to the north plane, and *S*, the normal to the south plane approaches 180 degrees. Note that the angle approaches 180 degrees, since the unit vectors point in the opposite direction (i.e., to the center of the global cell). Thus, the cosine of this angle approaches -1, so that the negative of this angle approaches 1 (line 1). The angle between the corners of the local cell will be used as the test angle that determines whether the global cell has been subdivided small enough. This test angle, which is approximately equal to the length of the side of the local cell divided by the radius of the Earth (line 2), is close to 0 degrees, and thus, the cosine of the test angle is close to 1. When the negative of the cosine between the north normal and the south normal becomes greater than or equal to the cosine of the test angle, then the global cell has been subdivided to approximately match the resolution of the local cell (lines 4-6).

It is also possible that the minimum cell size of the global map is larger than the size of the local cell, so the cosine of this minimum angle must also be tested against the north-to-south angle to avoid dividing the global cell too far (lines 7-9). It should be noted that if the minimum size of the global cell is reached, such that the global cell cannot be divided further and this resulting global cell is larger than the local cell used to update it, the local cell will be providing information for only a fraction of the area in the global cell. The obvious way to avoid this situation is to allocate enough memory for the global map and to set the minimum global cell size to be the size of a local cell. However, this may not always be practical. While uncertainty may exist for the rest of the global cell, we take the view that any information for the global cell from the local cell is better than no information. Thus, in the case where the global cell cannot be further divided to be the size of the local cell, we choose to update the global cell with the information in the local cell.

```
algorithm SEARCH-GLOBAL-MAP (gcell, loc)
1   $negcosangle_{N-S} \leftarrow -(gcell.N \bullet gcell.S)$
```

```
$cosangle_{test} \leftarrow \cos(\frac{size_{lcell}}{R_{Earth}})$
$cosangle_{min} \leftarrow \cos(\frac{size_{\text{min cell}}}{R_{Earth}})$
if ( $negcosangle_{N-S} \geq cosangle_{test}$ )
    UPDATE-GLOBAL-CELL(gcell, mean, std)
    return
else if ( $negcosangle_{N-S} \geq cosangle_{min}$ )
    UPDATE-GLOBAL-CELL(gcell, mean, std)
    return
else if $gcell.nodetype = "leaf"$
    if $totalNumCells > maxCells - 4$
        UPDATE-GLOBAL-CELL(gcell, mean, std)
        return
    else
        DIVIDE(gcell)
    end if
end if
/* Get the 4 children of the parent global cell. */
$childNW \leftarrow gcell.childNW$
$childNE \leftarrow gcell.childNE$
$childSW \leftarrow gcell.childSW$
$childSE \leftarrow gcell.childSE$
/* Continue the directed search in the quadrant in which loc lies. */
if $(childNW.S \bullet loc \geq 0)\ \&\ (childNW.E \bullet loc \geq 0)$
    SEARCH-GLOBAL-MAP(childNW, loc)  /* search northwest quadrant */
else if $(childNE.S \bullet loc \geq 0)\ \&\ (childNE.W \bullet loc \geq 0)$
    SEARCH-GLOBAL-MAP(childNE, loc)  /* search northeast quadrant */
else if $(childSE.N \bullet loc \geq 0)\ \&\ (childSE.W \bullet loc \geq 0)$
    SEARCH-GLOBAL-MAP(childSE, loc)  /* search southwest quadrant */
else if $(childSW.N \bullet loc \geq 0)\ \&\ (childSW.E \bullet loc \geq 0)$
    SEARCH-GLOBAL-MAP(childSW, loc)  /* search southeast quadrant */
end if
```

```
31 return
```

Figure 4- 15: Algorithm for searching for a local cell in the global map

4.4.4.2 Dividing a Global Map Cell

When new global cells are added to the global map, the current global cell must be divided into four new children cells (see Figure 4-16). Space for the new global cells must be allocated, and the global cell must be converted from a parent cell to a leaf cell. The position of the parent global cell must also be divided to determine the position of the children cells. Since the global map itself is not an exact square at the start, the children cells will not be exact squares. Even though the children cells do not have exactly the same size, the children cells are disjoint, and the union of the children cells is equal to the parent cell. Therefore, no information is lost in the division process. The division algorithm is presented in Figure 4-17.

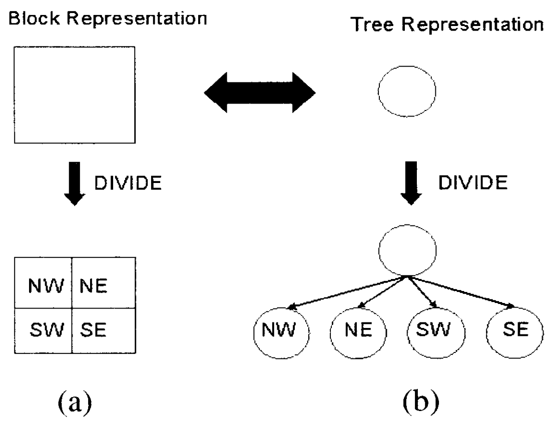

Figure 4-16: Global cell divided into four new children cells: (a) block representation and (b) tree representation

```
algorithm DIVIDE (gcell)
1  $mean \leftarrow gcell.value.l.mean$
2  $std \leftarrow gcell.value.l.std$
3  $midNandS \leftarrow$ bisector$(-gcell.N, gcell.S)$
4  $midEandW \leftarrow$ bisector$(-gcell.E, gcell.W)$
5  $gcell.nodetype \leftarrow$"$parent$"
   /* Create the northwest child and initialize its properties. */
6  $childNW \leftarrow gcell.childNW \leftarrow$ ALLOCATE($sizeof(globalMapCell)$)
7  $childNW.N \leftarrow gcell.N$
8  $childNW.E \leftarrow -midEandW$
```

```
9  childNW.S ← midNandS
10 childNW.W ← gcell.W
11 childNW.nodetype ←"leaf"
12 childNW.value.l.mean ← mean
13 childNW.value.l.std ← std
14 childNW.parent ← gcell
   /* Create the northeast child and initialize its properties. */
15 childNE ← gcell.childNE ← ALLOCATE(sizeof(globalMapCell))
16 childNE.N ← gcell.N
17 childNE.E ← gcell.E
18 childNE.S ← midNandS
19 childNE.W ← midEandW
20 childNE.nodetype ←"leaf"
21 childNE.value.l.mean ← mean
22 childNE.value.l.std ← std
23 childNE.parent ← gcell
   /* Create the southwest child and initialize its properties. */
24 childSW ← gcell.childSW ← ALLOCATE(sizeof(globalMapCell))
25 childSW.N ← −midNandS
26 childSW.E ← −midEandW
27 childSW.S ← gcell.S
28 childSW.W ← gcell.W
29 childSW.nodetype ←"leaf"
30 childSW.value.l.mean ← mean
31 childSW.value.l.std ← std
32 childSW.parent ← gcell
   /* Create the southeast child and initialize its properties. */
33 childSE ← gcell.childSE ← ALLOCATE(sizeof(globalMapCell))
34 childSE.N ← −midNandS
35 childSE.E ← gcell.E
```

```
$childSE.S \leftarrow gcell.S$
$childSE.W \leftarrow midEandW$
$childSE.nodetype \leftarrow "leaf"$
$childSE.value.l.mean \leftarrow mean$
$childSE.value.l.std \leftarrow std$
$childSE.parent \leftarrow gcell$
return
```

Figure 4-17: Algorithm for dividing a global map cell

4.4.4.3 Updating a Global Cell

When the global cell is broken down into small enough size so that it is the same size as the local cell, or that the minimum size of the global cell has been reached, the global cells will be updated. We assume that the size of the local cell is equal to or less than the size of the global cell, so it will always be the case that the global cell that corresponds to the local cell of interest will always be a leaf cell when the global cell has been broken down into small enough size.

The global cell is updated with the same update equations as those used for updating the local cell, where σ_1^2 and μ_1 are defined to be the existing variance and mean depth values in the global cell, respectively, and σ_2^2 and μ_2 are defined to be the values in the local cell that are being incorporated into the global cell. Updating the global map is a process of using each updated local cell to update the information in the corresponding global cell. In the case where the global map already contains a useful a priori map, it is expected that the values in the updated local cells will be similar to the values in the corresponding global cells, such that the values in the global cells should not change significantly. However, in the case where the global map contains little or no prior information, the updates to the global map cells should strongly reflect the information provided from the updated local map.

Since each update to the global map will split the global cells in the neighborhood of the global cell of interest until the smallest global cell that corresponds to the local cell has been obtained, the global map will need to be conglomerated to combine the same or similar cells again, such that the conglomerated global map will contain the minimum number of cells needed to represent the area at the given resolution. Without conglomeration, the storage benefit of the quadtree structure is lost. Conglomeration is described in detail below.

4.4.4.4 Conglomerating the Global Map

The global map is conglomerated after each update to the global map in order to combine cells that have the same or similar value (see Figure 4-18). The conglomeration starts at the root. While the global cell is a parent, the conglomerate function is recursively called on the four children of the global cell. When the four children are all leaves, the conglomerate function will combine the global cells by combining the four children nodes into one node if they share the same mean value or a similar mean value within a specified tolerance. Then the memory for the children nodes can be freed. This results in one cell taking the place of four cells, or a net decrease of three cells. This conglomeration algorithm is similar to the approach presented in Sammon [33] for conglomeration of similar blocks in a 3-dimensional k-d tree.

```
algorithm CONGLOMERATE (gcell)
1  if $gcell.nodetype = "leaf"$
2     return
3  end if
4  $childNW \leftarrow gcell.childNW$
5  $childNE \leftarrow gcell.childNE$
6  $childSW \leftarrow gcell.childSW$
7  $childSE \leftarrow gcell.childSE$
8  CONGLOMERATE(childNW)
9  CONGLOMERATE(childNE)
10 CONGLOMERATE(childSW)
11 CONGLOMERATE(childSE)
12 if ( $childNW.nodetype = "leaf"$ & $childNE.nodetype = "leaf"$ &
       $childSW.nodetype = "leaf"$ & $childSE.nodetype = "leaf"$ )
13    if ( $|childNW.mean - childNE.mean| \leq tol$ &
         $|childNW.mean - childSW.mean| \leq tol$ &
         $|childNW.mean - childSE.mean| \leq tol$ &
         $|childNE.mean - childSE.mean| \leq tol$ &
         $|childNE.mean - childSW.mean| \leq tol$ &
         $|childSE.mean - childSW.mean| \leq tol$ )
14       $gcell.mean \leftarrow \frac{1}{4}(childNW.value.l.mean + childNE.value.l.mean +$
            $childSW.value.l.mean + childSE.value.l.mean)$
15          $gcell.std \leftarrow \frac{1}{4}(childNW.value.l.std + childNE.value.l.std +$
            $childSW.value.l.std + childSE.value.l.std)$
16          $gcell.nodetype \leftarrow$ "leaf"
17          RELEASE($childNW$)
18          RELEASE($childNE$)
19          RELEASE($childSW$)
20          RELEASE($childSE$)
21     end if
22  end if
23  return
```

Figure 4-18: Algorithm for conglomerating the global map

4.5 Summary

This chapter has presented the design and representation of the static obstacle map. The static obstacle map stores information about stationary obstacles encountered in the UUV's underwater environment, and it maintains a measure of confidence in the measurement of the location of the obstacles.

The static obstacle map is represented with a two-layer approach. The local map is a uniform height field grid and stores the information in a small observation area in high detail in RAM on-board the vehicle. The local map allows quick updates to the information in the observation area, so that at every instant of time, the local map reflects the best estimate of the vehicle's current situation in this particular observation area. The global map is a quadtree and stores the information for the entire mission area in lower detail on disk. The global map allows a compact and efficient storage of the information in the UUV's entire mission area, so as to use as little of the limited memory and computational resources on-board the UUV. Algorithms were presented for updating the local map from the sonar inputs and for updating the global map from the information in the local map.

While the static obstacle map currently receives inputs during the mission only from the forward-looking sonar, the static obstacle map is versatile enough to be able to incorporate measurements from other sensors or even to fuse measurements from multiple sensors. Furthermore, the static obstacle map can be easily modified and expanded upon in future development of the mapping system.

[This Page Intentionally Left Blank]

5 Dynamic Obstacle Map

This chapter presents the design, representation, and implementation of the dynamic obstacle map. The dynamic obstacle map must be able to store information about the moving obstacles found in an underwater environment. The primary dynamic obstacles of concern are surface ships which can be detected with passive sonar and ISR sensors. However, the dynamic map structure should be general enough to incorporate other types of dynamic obstacles in the future.

Since the actual sensor data is noisy, the dynamic map must be able to filter the noisy sensor data to accurately estimate the location of the obstacle. In addition, the dynamic obstacle map should maintain a measure of uncertainty that reflects the uncertainty in the location of the obstacles.

The dynamic obstacle map must be able to support inputs from on-board sensors such as sonar and ISR sensors, as well as from off-board sources such as C4I updates[6]. The structure of the map should be independent of the source of the dynamic obstacle information.

5.1 Design of the Dynamic Obstacle Map

The dynamic obstacle map is represented with an object list, where the objects are the dynamic obstacles detected with the sensors. Each object stores information about the best estimate of its current position, the best estimate of its current velocity, and the uncertainty in its position and velocity estimates. We are assuming that the dynamic obstacles are traveling at constant velocity, and we are assuming that the sensors can detect any dynamic obstacle, regardless of its direction or range from the sensor.

The separation of the dynamic obstacle map from the static obstacle map allows each map to use a representation that is best suited to its type of stored information. The two separate but appropriate representations enable a more efficient way of representing the information than one all-purpose representation. A list of dynamic obstacles is much more memory efficient than storing a 3-dimensional grid of the environment that contains mostly free space. Furthermore, this object list representation is general and can store any type of dynamic obstacle.

The separation of the dynamic obstacle map from the static obstacle map will keep each map less cluttered, so that the relevant information can be accessed and interpreted easily. For

[6] We will consider a C4I (Command, Control, Communications, Computers, and Intelligence) update to be any update from sources outside of the UUV. The information provided through these updates is considered to be the "truth."

instance, if dynamic obstacles were stored in the same map as the static obstacles, it would be unclear whether the measure of confidence in the depth measurement would indicate an uncertainty in the depth measurement at that location due to non-flat terrain or imperfections in the sonar, or an uncertainty in the depth measurement due to a moving object passing through the area over time. In this case, there would be a problem of determining which measurements signify dynamic obstacles and which measurements simply signify noisy measurements. If the static obstacles and moving obstacles were to be represented in the same map, a more sophisticated measure of confidence or multiple different gauges of confidence would be needed.

We assume that the dynamic obstacles are located on or near the surface, out of view of the forward-looking sonar, and are detected with the passive sonar and ISR (intelligence, surveillance, reconnaissance) sensors. Thus, the dynamic obstacle map stores information orthogonal to that in the static obstacle map, in the sense that any obstacle detected with the passive sonar and ISR sensors is assumed to be moving and is stored in the dynamic obstacle map, while any obstacle detected with the forward-looking sonar is assumed to be stationary and is stored in the static obstacle map. The dynamic obstacle map does not need to decide whether or not an obstacle is stationary or moving. By storing the data in a manageable and intelligent manner, the mapping system can be easily extended in the future to deal with information from multiple sensors and about other types of obstacles.

5.2 Inputs to the Dynamic Obstacle Map

The dynamic obstacle map receives inputs from on-board sensors such as sonar and ISR sensors, as well as from off-board sources such as C4I updates. The C4I updates can be received at any time, and they will be treated as truth information, since this update will be able to provide the most accurate estimate of the true state of the obstacles at the current time.

The simulation does not have an explicit model for the ISR sensors, so we will assume that the identification, classification, and range information are available from the passive sonar simulation. The C4I updates will provide the same information as the sensors, but they will also be able to provide information about the velocity of the obstacle. Thus, the inputs to the dynamic map at each update time are:

t	- time of update
ϕ	- heading of the vehicle at time t
lat_{uuv}	- latitude of vehicle at time t

lon_{uuv}	- longitude of vehicle at time t
num_{dynObs}	- number of dynamic obstacles detected with sensors at time t

For each dynamic obstacle, the map will input:

r	- range to the dynamic obstacle from the vehicle at time t
β	- bearing to the dynamic obstacle from the vehicle at time t
$class$	- number that indicates the type or classification of the obstacle
id	- number that uniquely identifies the obstacle
(v_x, v_y)	- velocity of obstacle at time t (if available)

Velocity (v_x, v_y) is included as an input in order to allow the dynamic map structure to be independent of the source of the information updates. Only a C4I update will give velocity for the obstacle, and in this case, the velocity input will be a valid value. However, in the case of an update from the on-board sensors, the velocity input will be 0 (to indicate that the velocity input is invalid).

5.3 Implementation of the Dynamic Obstacle Map

The dynamic obstacle map is implemented as a one-dimensional, fixed-length array. The length of the array is set to be the maximum number of obstacles, which is a parameter that is selected before the initialization of the dynamic map. Each obstacle in the list stores its estimated position, estimated velocity, classification type, the time at which the observation was made, and an identification number that is unique for each obstacle. In addition, each obstacle stores a flag that indicates whether or not the obstacle is active, or in the field of view of the sensors. An obstacle is marked as active when it is detected for the first time and added to the map. The active flag can be turned off to simulate that the obstacle has moved outside the field of view of the sensors.

A dynamic obstacle will also store a variance and covariance terms, for both the x-coordinate and the y-coordinate, that are necessary for filtering and accurately estimating the obstacle's position and velocity. These variance and covariance terms form the measure of uncertainty in the obstacle position. The filtering will be discussed in the next sections. The structure of a dynamic obstacle is summarized in Figure 5-1.

```
DYNAMIC OBSTACLE
struct DynamicObs {
   int active;                          /* {1 = active, 0 = not active} */
   int id;                              /* unique identification for obstacle */
   int class;                           /* classification of obstacle */
   double time;                         /* time of the current update */
   double pos_x, pos_y;                 /* estimate of position of obstacle */
   double vel_x, vel_y;                 /* estimate of velocity of obstacle */
   double var_pos_x, var_vel_x, cov_x; /* estimate of covariance/variance in x-pos/vel */
   double var_pos_y, var_vel_y, cov_y; /* estimate of covariance/variance in y-pos/vel */
} dynamicObs
```

Figure 5-1: Stucture of a dynamic obstacle in the dynamic obstacle map

5.4 Estimating the Position and Velocity of an Obstacle

If the sensor measurements were perfectly accurate, then the range and bearing measurements would exactly correspond to the position of the obstacle. If we assume that the obstacle is traveling at a constant velocity, then we could infer the velocity of the obstacle at time step k as the difference of the obstacle's position, p_k and p_{k-1}, divided by the difference of the time, t_k and t_{k-1}.

$$v_k = \frac{p_k - p_{k-1}}{t_k - t_{k-1}} \tag{5.1}$$

However, the sensor measurements of range and bearing are noisy, such that the corresponding position of the obstacle is not accurate. We want to estimate the state of the obstacle (the position and velocity), such that our estimate is an accurate estimate of the true state even though we cannot measure the true state directly. We want an unbiased estimate, such that the average value of the state estimate is equal to the average value of the true state. We want to use an estimator that minimizes the estimation error variance, so that the state estimate varies from the true state as little as possible. The linear Kalman filter satisfies these criteria, under the assumption of Gaussian noise.

5.4.1 Kalman Filter

The Kalman filter is a multiple-input, multiple-output digital filter with time-varying gains that can optimally estimate the state of a system in real time based on the observations of its noisy outputs [19], [34], [41]. The estimates are optimal in the sense that they minimize the variance of the estimation error. Starting with an initial predicted state estimate and its associated covariance obtained from past information, the filter combines the predicted estimate with the new measurement to obtain the optimal estimate. As the filter recursively processes more measurements, the estimate of the state should more closely approximate the true state of the system. The Kalman filter has the desirable property of maintaining the first two moments of the state distribution.

$$E[x_k] = \hat{x}_k \tag{5.2}$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k \tag{5.3}$$

In order to obtain an optimal estimate with the Kalman filter, the system that is to be estimated must satisfy certain assumptions (or approximately satisfy the assumptions). Since the Kalman filter algorithm is linear, the process generating the measurements is assumed to be linear[7]. A linear system can be described by the following equations.[8]

State Equation:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{5.4}$$

Output Equation:

$$y_k = Cx_k + z_k \tag{5.5}$$

A, B, and C are matrices, x is the state vector of the system, u is the known input vector to the system, y is the measured output vector of the system, w is the process noise vector, z is the measurement noise vector, and k is the time index. The state vector x contains all of the information that describes the behavior of the system, but x cannot be measured directly. Instead,

[7] Here we are discussing a Linear Kalman Filter. An Extended Kalman Filter allows relaxation of the linearity constraint.
[8] We use the notation presented in Simon [34] for the Kalman filter.

y can be measured, and the noisy information in y can be used to accurately estimate x. Because noise is corrupting the system, equations (5.4) and (5.5) are linear stochastic difference equations. In the absence of the random noise w and z, x and y would simply follow linear deterministic transformations.

The process noise w and the measurement noise z are assumed to be Gaussian white noise with process noise covariance S_w and measurement noise covariance S_z. Thus,

$$f_W(w) \sim N(0, S_w) \tag{5.6}$$

$$f_Z(z) \sim N(0, S_z) \tag{5.7}$$

The process noise covariance S_w is defined as

$$S_w = E[w_k w^T{}_k] \tag{5.8}$$

Similarly, the measurement noise covariance S_z is defined as

$$S_z = E[z_k z^T{}_k] \tag{5.9}$$

While in reality S_w and S_z many vary in time, we assume that they remain constant. Furthermore, we assume that w and z are independent and uncorrelated with each other at all times i and j, such that

$$E[w_i z_j] = 0 \tag{5.10}$$

Provided that the constraints can be met, the recursive Kalman filter algorithm allows for efficient real-time processing. The updated estimate is based solely on the current measurement at time step k and the information from the one previous time step k-1. To prepare for the next measurement, the filter projects the updated state estimate and its associated covariance to the next measurement time. Starting with an initial predicted state estimate and its associated covariance obtained from past information, the filter calculates the weights to be used for combining this estimate with the measurement. This predicted state is combined with the measurement to produce the optimal state estimate and covariance estimate. This predict-update process at time step k can be described by the following three equations.

$$K_k = AP_kC^T\left(CP_kC^T + S_z\right)^{-1} \tag{5.11}$$

$$\hat{x}_{k+1} = \left(A\hat{x}_k + Bu_k\right) + K_k\left(y_{k+1} - C\hat{x}_k\right) \tag{5.12}$$

$$P_{k+1} = AP_kA^T + S_w - AP_kC^TS_z^{-1}CP_kA^T \tag{5.13}$$

K is the Kalman gain matrix, and it indicates how much to weight the new measurement when computing the next state estimate. If the measurement noise is large, S_z will be large, so K will be small and we will give the measurement y a small weight in the computation for the next state estimate, $\hat{x}_{k+1}$. If the measurement noise is small, S_z will be small, so K will be large and the measurement y will be given a large weight in the computation for the next state estimate, $\hat{x}_{k+1}$. The first term in equation (5.12) is the propagation in time of the state estimate at time step k to the time step $k+1$. In the absence of a measurement, this linearly propagated state would be the new state estimate. However, the presence of a measurement requires the addition of a correction term, the second term in equation (5.12), which reflects the amount by which to correct the propagated state estimate due to the measurement. P is the estimation error covariance matrix, or the estimate of the uncertainty in the state estimate. Each diagonal term in P is the variance of a state variable, and the off-diagonal terms are the covariance terms that describe the correlation between pairs of state variables.

5.4.2 Setup of the One-dimensional Obstacle Tracking Problem

We start with an example of tracking an obstacle in one dimension, and we will later consider the modifications for tracking an obstacle in two dimensions. We can express the relation for the one-dimensional position and velocity as:

$$T = t_{k+1} - t_k \tag{5.14}$$

$$p_{k+1} = p_k + Tv_k + \tilde{p}_k \tag{5.15}$$

$$v_{k+1} = v_k + \tfrac{1}{2}T^2 u_k + \tilde{v}_k \tag{5.16}$$

The position at time step *k+1*, p_{k+1}, is equal to the deterministic position at time step *k+1* (the previous value of the position at time step *k* plus the incremental position change in the time *T* due to the velocity) plus the position noise, $\tilde{p}_k$, which is unknown. The velocity at time step *k+1*, v_{k+1}, is equal to the deterministic velocity at time *k+1* (the previous value of the velocity at time step *k* plus the incremental velocity change in the time *T* due to the acceleration) plus the velocity noise, $\tilde{v}_k$, which is unknown.

We define the state of the system to be:

$$x_k = \begin{bmatrix} p \\ v \end{bmatrix}_k \tag{5.17}$$

and the output of the system to be:

$$y_k = p_k \tag{5.18}$$

Then we can write equations (5.15) and (5.16) in matrix form.

$$x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \tfrac{1}{2}T^2 \\ T \end{bmatrix} u_k + w_k \tag{5.19}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + z_k \tag{5.20}$$

Equation (5.19) is in the form of equation (5.4), where *A* is the *2x2* matrix

$$A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{5.21}$$

B is the *2x1* column vector

$$B = \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} \tag{5.22}$$

C is the *1x2* row vector

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{5.23}$$

Using equations (5.8) and (5.9), S_w and S_z are

$$S_w = E\left[ww^T\right] = E\left[\begin{bmatrix} w_p \\ w_v \end{bmatrix} \begin{bmatrix} w_p \\ w_v \end{bmatrix}^T\right] = E\begin{bmatrix} w_p^{\ 2} & w_p w_v \\ w_p w_v & w_v^{\ 2} \end{bmatrix} \tag{5.24}$$

$$S_z = E\left[zz^T\right] = E\left[\left[z_p\right] \ \left[z_p\right]^T\right] = E\left[z_p^{\ 2}\right] \tag{5.25}$$

We have set up the one-dimensional filtering problem, and we will now consider the two-dimensional filtering problem.

5.4.3 Setup of the Two-dimensional Obstacle Tracking Problem

We assume that the obstacle is free to move in two dimensions (i.e., planar motion). Since we are only tracking surface ships, the assumption of a constant depth is valid. We want to track the obstacle based on range and bearing measurements over time. The passive sonar will report a measurement of the range r_k and bearing β_k at time *k*. The range and bearing are related to the position of the obstacle (p_x, p_y) as:

$$r = \left(p_x - x_{uuv}\right)^2 + \left(p_y - y_{uuv}\right)^2 \tag{5.26}$$

$$\beta = \tan^{-1}\left(\frac{p_y - y_{uuv}}{p_x - x_{uuv}}\right) - \phi \tag{5.27}$$

Since the range and bearing are related non-linearly to the state of the system and since the linear Kalman filter assumes that the measurement is related linearly to the state of the system, we will first transform the range and bearing measurement of the obstacle relative to the vehicle to the corresponding position of the obstacle, as in Figure 5-2. We can represent the obstacle's position in Cartesian coordinates in the local coordinate frame at time step k as $(p_x, p_y)_k$ and its velocity as $(v_x, v_y)_k$. f is the non-linear transformation function that relates (r, β) to (p_x, p_y) by the combination of equations (5.26) and (5.27). Now, we will use this transformed position as the measurement, and we can use the linear Kalman filter to estimate the state of the system.

$$\begin{tikzcd}(r,\beta) \arrow[r] & \boxed{f_{non\text{-}linear}(r,\beta)} \arrow[r] & (p_x, p_y)\end{tikzcd}$$

Figure 5-2: Non-linear transformation of range and bearing measurements to x-y position measurement

This process of linearizing the measurement is one method of linearization. The use of the Extended Kalman Filter is another method of linearizing the measurement. Both methods make a linear approximation of a non-linear process. It is important to note that the non-linear transformation does not preserve the Gaussian distribution of the noise. This is also an issue with the use of the Extended Kalman filter. Therefore, instead of assuming that the range and bearing measurements are corrupted with Gaussian noise, we are actually assuming that the transformed measurements of x-y position are corrupted with additive Gaussian white noise. This is equivalent to assuming that the range and bearing are corrupted with some noise that when transformed to the x-y position becomes Gaussian white noise.

5.4.4 Filtering in Two Dimensions

We will further assume that the x-coordinate and the y-coordinate of the state are independent. This enables us to use a simple one-dimensional implementation of the filter to filter each coordinate separately. We define the state of the system as two separate vectors, one for the x-coordinate and one for the y-coordinate.

While it is possible to combine the two separate states into one larger state, it is also possible to decouple the system and to approach the filtering of the position as two separate one-dimensional filtering problems. This approach consists of two steps, as described below.

STEP 1: FILTERING THE X-COORDINATE

State of the system:

$$x_{x,k} = \begin{bmatrix} p_x \\ v_x \end{bmatrix}_k$$

Measurement:

$$y_{x,k} = p_{x,k}$$

STEP 2: FILTERING THE Y-COORDINATE

State of the system:

$$x_{y,k} = \begin{bmatrix} p_y \\ v_y \end{bmatrix}_k$$

Measurement:

$$y_{y,k} = p_{y,k}$$

While the assumption of independence will limit the possibilities for the joint probability density function of the x-coordinate and y-coordinate of position, this implementation is sufficient to demonstrate the fundamental properties of the dynamic obstacle map in this thesis.

5.5 Updating the Dynamic Obstacle Map

At each time $t_k = kT$, where T is the time-step interval and k is an integer time index, the sensors return a list of the detected obstacles. The data contained in the list of obstacles is used to update the dynamic map. For each detected obstacle, the range and bearing measurements are used to calculate the position of the obstacle in Earth-fixed coordinates.

For each obstacle, the identification of the obstacle is used to determine whether or not the obstacle already exists in the dynamic obstacle map (see Figure 5-3). If the identification of the obstacle is found to match the identification of an existing obstacle in the map, and if the obstacle is flagged as active in the map, the properties of the existing obstacle are updated. If the update is from C4I information, then the obstacle's properties are directly overwritten. Otherwise, the update is from the onboard sensors, and the measurements are used with a Kalman filter to estimate the true state of the obstacle.

In the case that the obstacle was not found in the map, or if the obstacle is not active in the map, the new obstacle will be added to the dynamic obstacle map, as long as there is space available. If the update is a C4I update, the new obstacle's properties will be initialized with the values from the C4I information. Otherwise, the obstacle's position and velocity estimates will be initialized to be the values from the Kalman filter initialization. If there is no space left in the dynamic obstacle map, then the new obstacle cannot be added, and there will be an error message. The algorithm is given in Figure 5-4, and a more detailed line-by-line description follows.

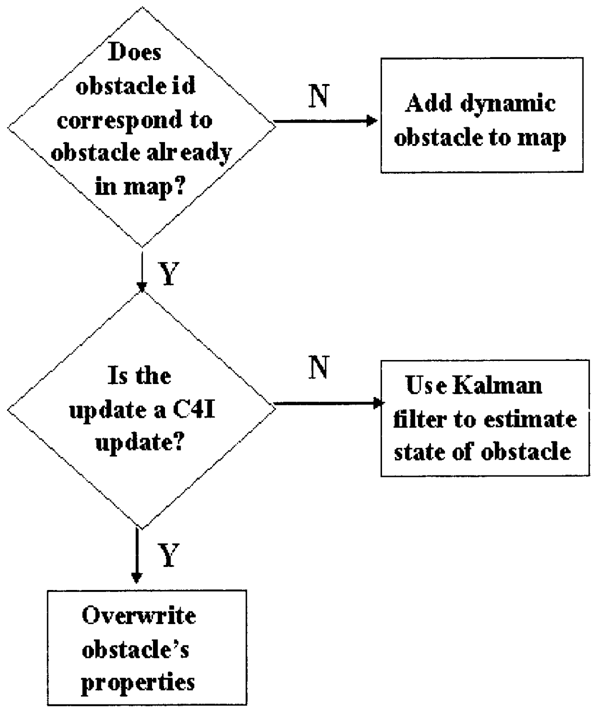
Does obstacle id correspond to obstacle already in map?
N
Add dynamic obstacle to map
Y
Is the update a C4I update?
N
Use Kalman filter to estimate state of obstacle
Y
Overwrite obstacle's properties

Figure 5-3: Updating the dynamic obstacle map

```
algorithm UPDATE-DYNAMIC-MAP
1 for each active dynamic obstacle $i$
2     $\theta \leftarrow \phi + \beta$
3     $\Delta_x \leftarrow r\cos(\theta), \Delta_y \leftarrow r\sin(\theta)$
4     $\{x_{uav}, y_{uav}\} \leftarrow env\_xy(lat_{uav}, lon_{uav})$
```

```
$x_{obs} \leftarrow x_{uuv} + \Delta_x, y_{obs} = y_{uuv} + \Delta_y$
$j \leftarrow 0$
while $id \neq list(i).id$ & $j < num_{dynObs}$
    $j \leftarrow j+1$
end while
    if $j < num_{dynObs}$ & $list(i).active = TRUE$
        if update = "C4I"
            $list(j).pos\_x \leftarrow x_{obs}, list(j).pos\_y \leftarrow y_{obs}$
            $list(j).vel\_x \leftarrow v_x, list(j).vel\_y \leftarrow v_y$
            $list(j).id \leftarrow id, list(j).class \leftarrow class$
        end if
        else
            $init \leftarrow FALSE$
            $M \leftarrow \{x_{obs}\}$
            $S \leftarrow \{list(j).pos\_x, list(j).vel\_x\}$
            $C \leftarrow \{list(j).\mathrm{var}\_pos\_x, list(j).\mathrm{var}\_vel\_x, list(j).\mathrm{cov}\_x\}$
            $\{S', C'\} \leftarrow kalman\_filter(M, S, C, init)$
            $\{list(j).pos\_x, list(j).vel\_x\} \leftarrow S'$
            $\{list(j).\mathrm{var}\_pos\_x, list(j).\mathrm{var}\_vel\_x, list(j).\mathrm{cov}\_x\} \leftarrow C'$
            $M \leftarrow \{y_{obs}\}$
            $S \leftarrow \{list(j).pos\_y, list(j).vel\_y\}$
            $C \leftarrow \{list(j).\mathrm{var}\_pos\_y, list(j).\mathrm{var}\_vel\_y, list(j).\mathrm{cov}\_y\}$
            $\{S', C'\} \leftarrow kalman\_filter(M, S, C, init)$
            $\{list(j).pos\_y, list(j).vel\_y\} \leftarrow S'$
            $\{list(j).\mathrm{var}\_pos\_y, list(j).\mathrm{var}\_vel\_y, list(j).\mathrm{cov}\_y\} \leftarrow C'$
        end else
    end if
    else if $j < num_{\max}$
        $num_{dynObs} \leftarrow num_{dynObs} + 1$
```

```
34          $list(j).active \leftarrow TRUE$
35          $list(j).id \leftarrow id, list(j).class \leftarrow class$
36          **if** update = "C4I"
37              $list(j).pos\_x \leftarrow x_{obs}, list(j).pos\_y \leftarrow y_{obs}$
38              $list(j).vel\_x \leftarrow v_x, list(j).vel\_y \leftarrow v_y$
39          **end if**
40          **else**
41              $init \leftarrow TRUE$
42              $M \leftarrow \{x_{obs}\},\ S \leftarrow \{\},\ C \leftarrow \{\}$
43              $\{S', C'\} \leftarrow kalman\_filter(M, S, C, init)$
44              $\{list(j).pos\_x, list(j).vel\_x\} \leftarrow S'$
45              $\{list(j).\mathrm{var\_}pos\_x, list(j).\mathrm{var\_}vel\_x, list(j).\mathrm{cov\_}x\} \leftarrow C'$
46              $M \leftarrow \{y_{obs}\},\ S \leftarrow \{\},\ C \leftarrow \{\}$
47              $\{S', C'\} \leftarrow kalman\_filter(M, S, C, init)$
48              $\{list(j).pos\_y, list(j).vel\_y\} \leftarrow S'$
49              $\{list(j).\mathrm{var\_}pos\_y, list(j).\mathrm{var\_}vel\_y, list(j).\mathrm{cov\_}y\} \leftarrow C'$
50          **end else**
51      **else**
52          ERROR: print("Out of memory in the dynamic map")
53      **end else**
54      **return**
```

Figure 5-4: Algorithm for updating the dynamic obstacle map

Description

1. Loop through each dynamic obstacle that was detected with the on-board sensors or a C4I update.
2. Calculate the heading of the obstacle as the sum of the bearing to the obstacle and the heading of the vehicle.
3. Calculate the difference in the *x-y* position from the obstacle to the vehicle.
4. Convert the *lat-lon* position of the vehicle into the *x-y* position of the vehicle.
5. Calculate the *x-y* position of the obstacle from the *x-y* position of the vehicle and the

difference in the position.

6. Set j to be 0, to signify the beginning of the dynamic obstacle map.

7-9. While the detected dynamic obstacle is not the same as the obstacle at map element j, look at the next dynamic obstacle in the dynamic obstacle map and repeat. When the while loop terminates, either the obstacle has been found in the map at element j, or the entire map has been searched without success.

10. If the obstacle was successfully found in the map

11. If the update is a C4I update

12-14. Overwrite its position, velocity, and identification and classification values.

15. End the C4I update.

16. Otherwise, if the update is from the on-board sensors

17. Set the filter initialization flag to be false to indicate that the filter has already been initialized.

18-20. Save the input x-position as the measurement, save the previous x-position and x-velocity as the state, and save the previous x-covariance for the x-position and x-velocity.

21. Filter to get new estimate of x-position, x-velocity, and x-covariance

22-23. Update the obstacle's x-position and x-velocity with the new estimate of the state in the x-dimension and update the obstacle's x-covariance with the new estimate of the covariance in the x-dimension.

24-29. Filter the y-dimension in the same manner as 17-21, and update the y-dimension properties of the obstacle in the same manner as 22-23.

30. End the on-board sensor update.

31. End of the update for an existing obstacle in the map.

32. If the obstacle was not found in the map but there is still space available in the map

33-34. Increment the count of the number of obstacles stored in the map and set the active flag as true.

35. Initialize the identification and classification properties of the new obstacle.

36-39. If the update is a C4I update, initialize the position and velocity properties with the inputs.

40-42. If the update is from the on-board sensors, set the filter initialization flag to be true to indicate that the filter has not already been initialized, set the x-position as the measurement, set the previous state to be null, and set the previous covariance to be null.

43. Filter to get new estimate of the state and covariance for the x-coordinate.

44. Update the obstacle's x-position and x-velocity with the new estimate of the x-state.

45. Update the obstacle's x-covariance with the new estimate of the x-covariance.

46-49. Filter the y-coordinate in the same manner as 41-44, and update the y-coordinate of the obstacle's properties.

50. End of the on-board sensor update.

51-52. If the obstacle was not found in the map and there is no more space available to add the obstacle to the map, then print an error message.

53-54. End of the updating of the dynamic map.

5.6 Summary

This chapter has presented the design and implementation of the dynamic obstacle map. The dynamic obstacle map is represented as a list of the dynamic obstacles. This representation can store any type of obstacle and is more memory-efficient than a 3-D uniform grid of the operational environment. Since the range and bearing measurement is a non-linear function of the state, we linearize the measurement by transforming the *range-bearing* measurement to an x-y position measurement. Then the linear Kalman filter is used to estimate the state of the dynamic obstacles over time, where the state is the position and the velocity of the obstacle and the measurement is the x-y position of the obstacle. The linear Kalman filter gives an estimate of the current position and velocity of the obstacle, as well as a measure of confidence in this estimate. The filtering assumes that the x-position and the y-position are independent, and two-dimensional filtering is accomplished by separate applications of the one-dimensional filter to each coordinate.

6 Results

The mapping system has been integrated into the MTV simulation. The simulation makes calls to the mapping system, in addition to calls to the guidance, navigation, control, and sonar systems. Simulated noisy measurements of the Narragansett Bay terrain are output from the forward-looking sonar model and input to the static obstacle map. Simulated noisy measurements of constant-velocity moving targets are output from the passive sonar model and input to the dynamic obstacle map. This chapter will present some results and analysis of the simulation-based testing of the mapping system.

6.1 Results of Static Obstacle Map

The static obstacle map was tested in the simulation with simulated noisy measurements of the Narragansett Bay truth data as inputs. The vehicle was driven around in various small areas, and local maps were constructed of the areas. Comparison with the truth depth data showed that the local cells were quickly updated to reflect accurate depth estimates, even in the case of a bad *a priori* local map. The neighbor smoothing was especially useful in the case of a bad *a priori* local map. The local maps were transferred to the global map, and the corresponding section of the global map was updated.

6.1.1 Performance of the Local Map

Figure 6-1 shows the accumulated depth error in the local map for a portion of a row of local cells after several updates from the forward-looking sonar. In each of these local cells, the accumulated error between the true value of depth and the accumulated mean value of the depth is less than 3 standard deviations (99.9% confidence).

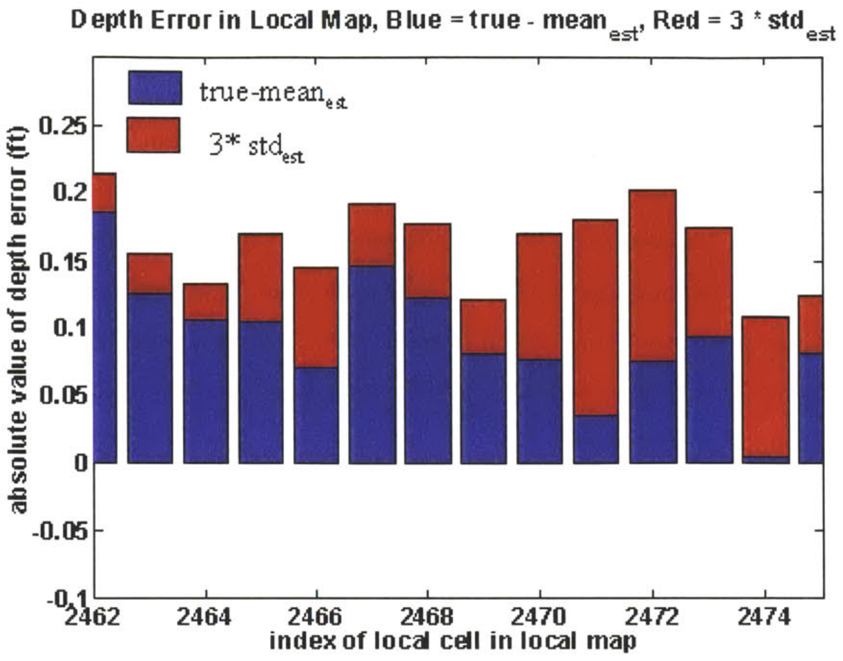

Figure 6-1: Accumulated error in the depth measurement in local cells

6.1.2 Performance of the Global Map

The global map produced an efficient representation for the data in the larger mission area. We can quantify a measure of the efficiency of the global map as the number of resulting global cells, which is directly related to the amount of memory used. For typical simulation tests for updating the global map with a local map, where the global map minimum cell size was set to 10m, the merging tolerance to 0.5m, and the global map size to 1267 nmi x 1267 nmi, and where the local map has a size of 500m x 500m with a local cell size of 10m, the resulting global map contained approximately 220 global cells. For the implementation in this thesis, each global cell requires 128 bytes of memory, resulting in a total of 28.2kB of memory to store the global map. For a given global map, this memory usage can be further reduced by choosing a larger merging tolerance. In addition, we can further reduce the memory usage if necessary by using less memory to store each global cell. For example, we could use floats instead of doubles for the values stored in the global cell.

6.2 Results of Dynamic Obstacle Map

The dynamic obstacle map was tested in the UUV simulation to ensure that the position and velocity estimates of the dynamic obstacles were indeed accurate. The dynamic obstacles were modeled in the simulation as having constant velocity. The amount of noise was varied in the sonar model in the simulation, and the dynamic obstacle map showed that the error between the estimate and the truth data decreases over time. The larger the amount of noise, the longer it took for the error to decrease, and the larger was the uncertainty in the estimate.

As a specific example, we modeled a dynamic obstacle that was moving at a constant speed of 3.4ft/s and at a heading of 290 degrees. The UUV was traveling toward this obstacle at first, but then was moving away for the remainder of this test. Figure 6-2 shows that while the measured range is quite noisy, the estimated range is close to the true range. The estimated range is much better approximation to the true range than is the measured range, especially in this case where the simulated range noise is large. Figure 6-3 shows a similar result for the bearing. Figure 6-4 and Figure 6-5 show that the estimate of the x-position and the estimate of the y-position approach a better approximation of the true x-position and the true y-position as time increases. Initially, there is a deviation between the estimate and the truth because the initial estimate is the initial noisy measurement of the position. However, as more measurements are taken, the estimate begins to better approximate the truth.

Figure 6-6 shows that the error in the estimate of the x-position is decreasing over time, and Figure 6-7 shows that the error in the estimate of the y-position is similarly decreasing over time. The error in the position estimate is within the steady-state error bounds[9]. Again, the initial values for the error are large, but as more measurements accumulate, the error decreases and is bounded by the steady-state error bounds. Figure 6-8 and Figure 6-9 show that the measured position is very noisy and that the estimated position is a much better indicator of the true position.

Figure 6-10 and Figure 6-11 show a comparison between the true velocity and the estimated velocity of the dynamic obstacle. Figure 6-12 and Figure 6-13 show the error between the estimated velocity and the true velocity.

[9] We define the steady-state error bounds as three standard deviations of the steady-state error. The standard deviation is obtained from the variance that is maintained in the covariance matrix for each dynamic obstacle.

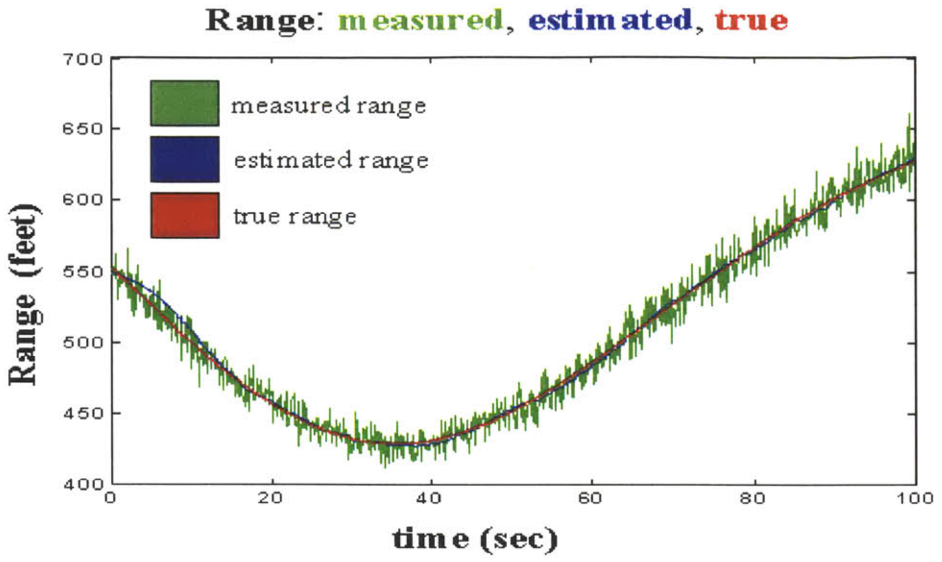

Figure 6-2: Comparison of the measured, estimated, and true range to a dynamic obstacle

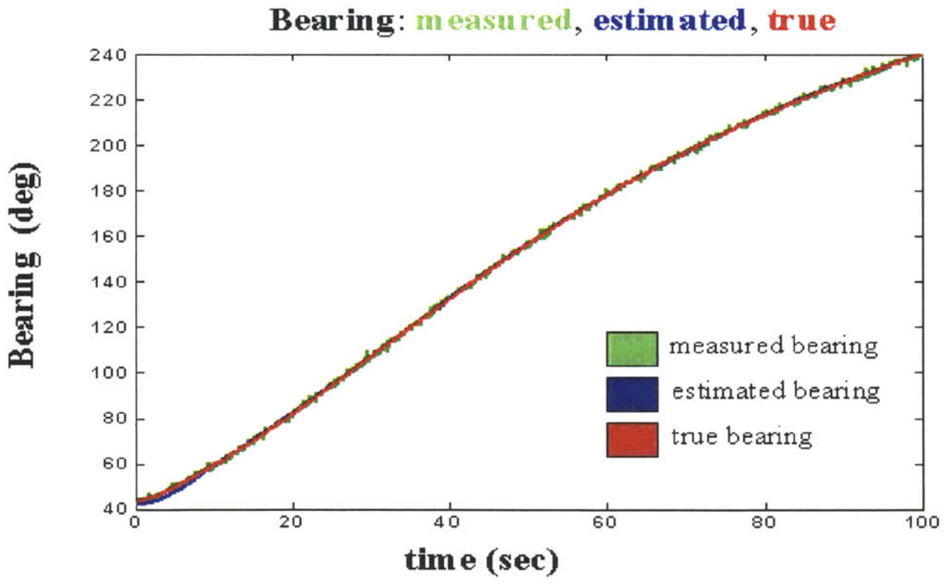

Figure6-3: Comparison of the measured, estimated, and true bearing to a dynamic obstacle

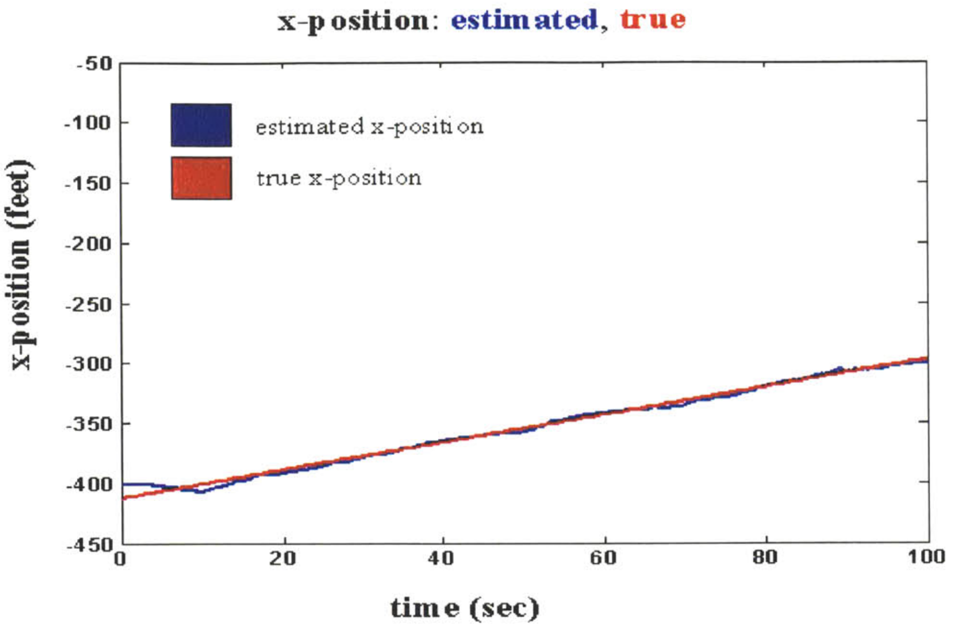

Figure 6-4: Comparison of the estimated and true x-position of a dynamic obstacle

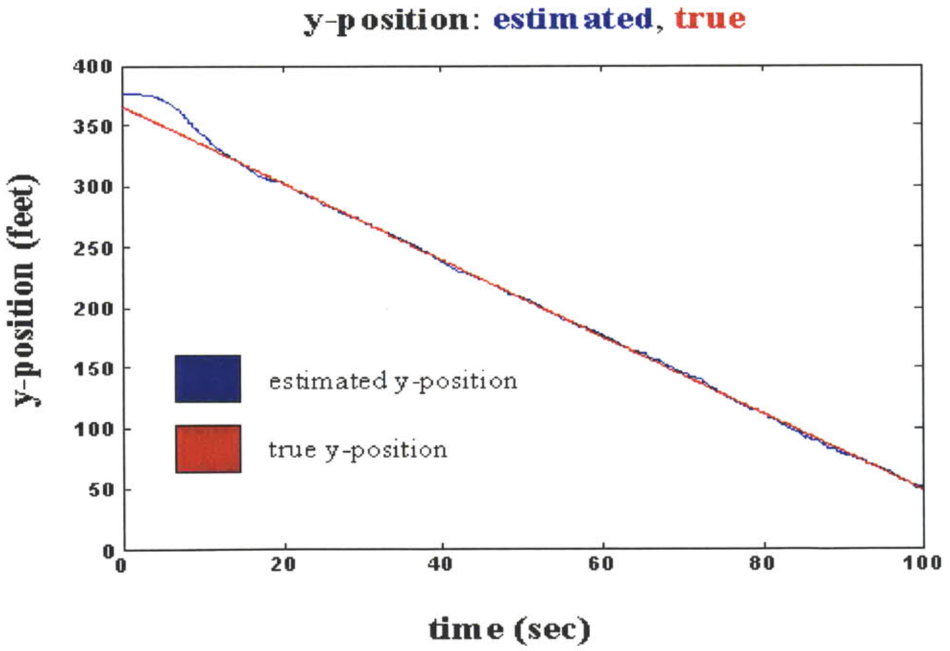

Figure 6-5: Comparison of the estimated and true y- position of a dynamic obstacle

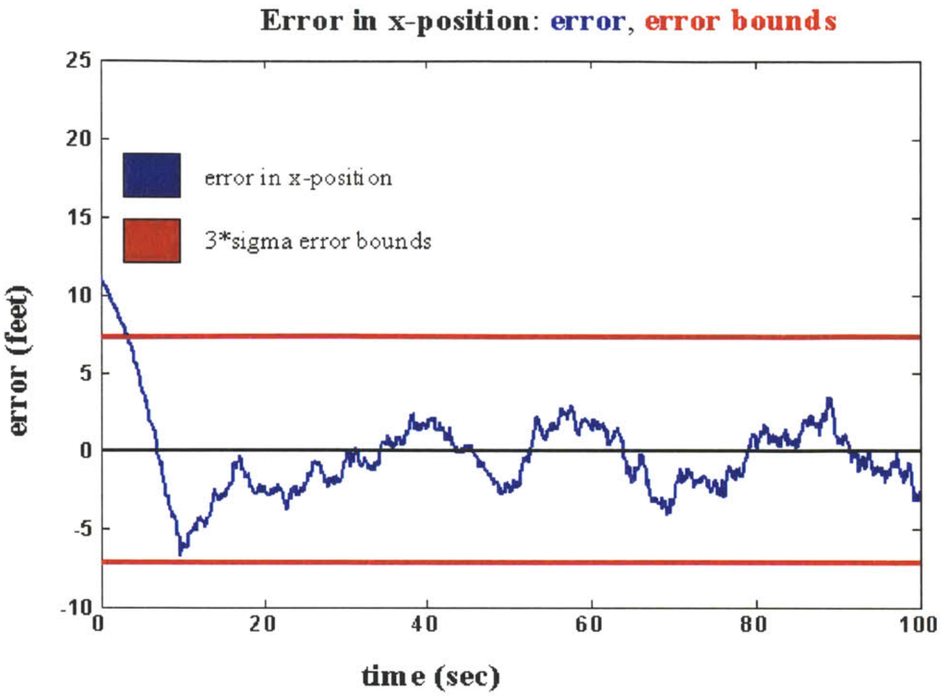

Figure 6-6: Error in the estimate of the x-position of a dynamic obstacle compared to the error bounds

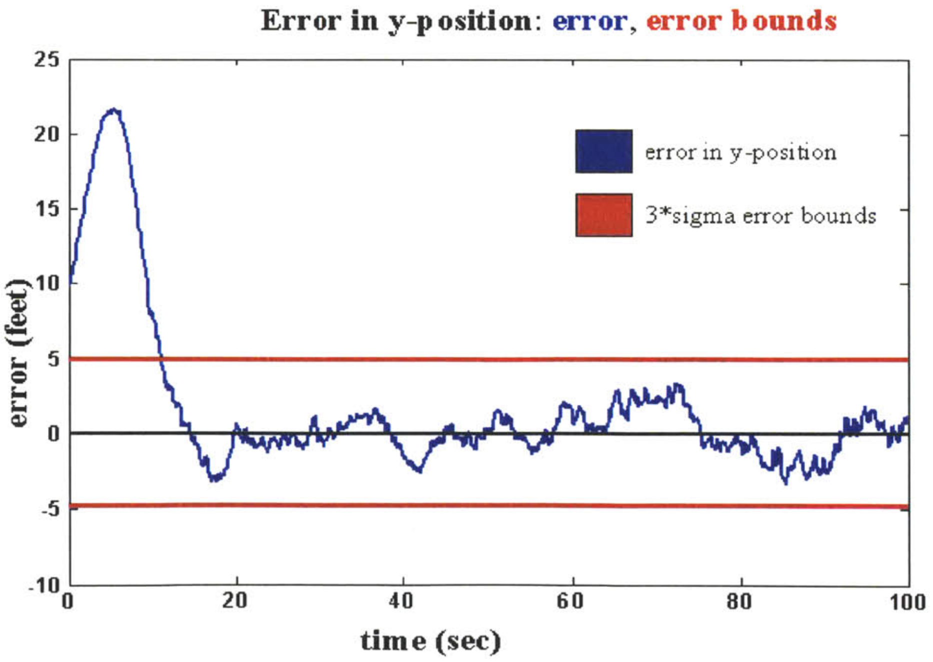

Figure 6-7: Error in the estimate of the x-position of a dynamic obstacle compared to the error bounds

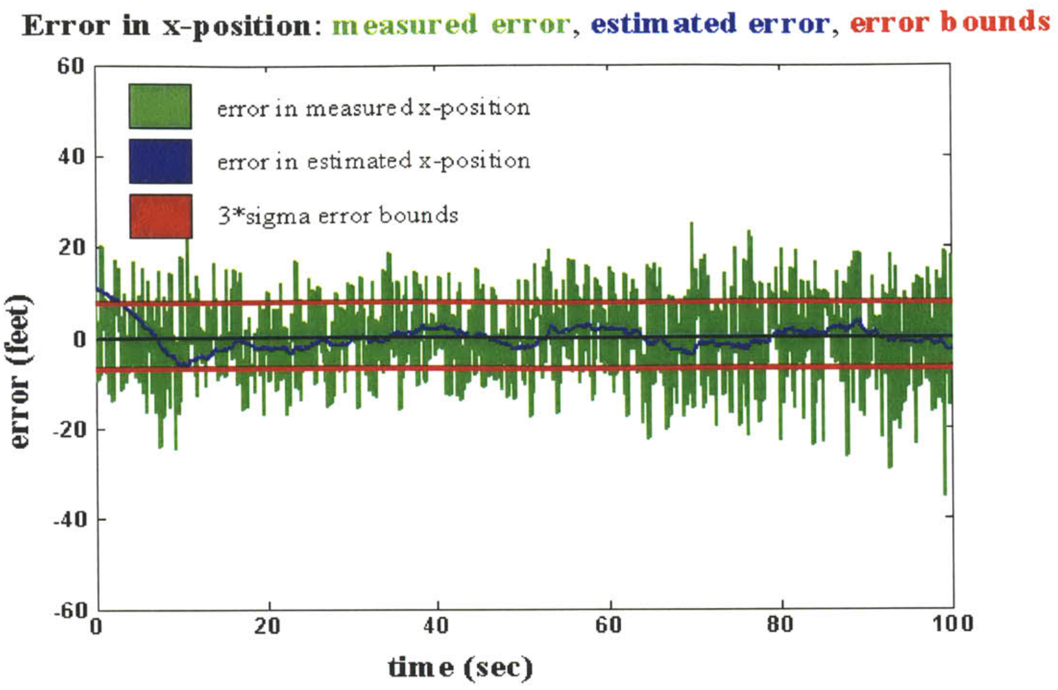

Figure 6-8: Comparison of the error in the measured, estimated, and true x-position of a dynamic obstacle

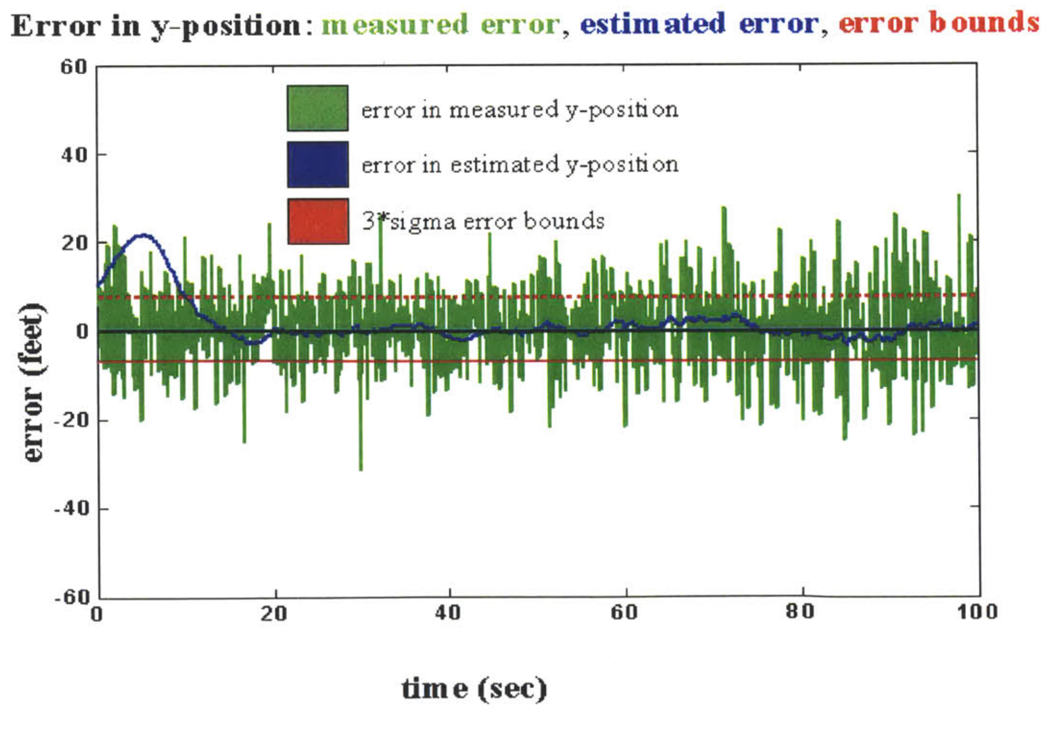

Figure 6-9: Comparison of the error in the measured, estimated, and true y-position of a dynamic obstacle

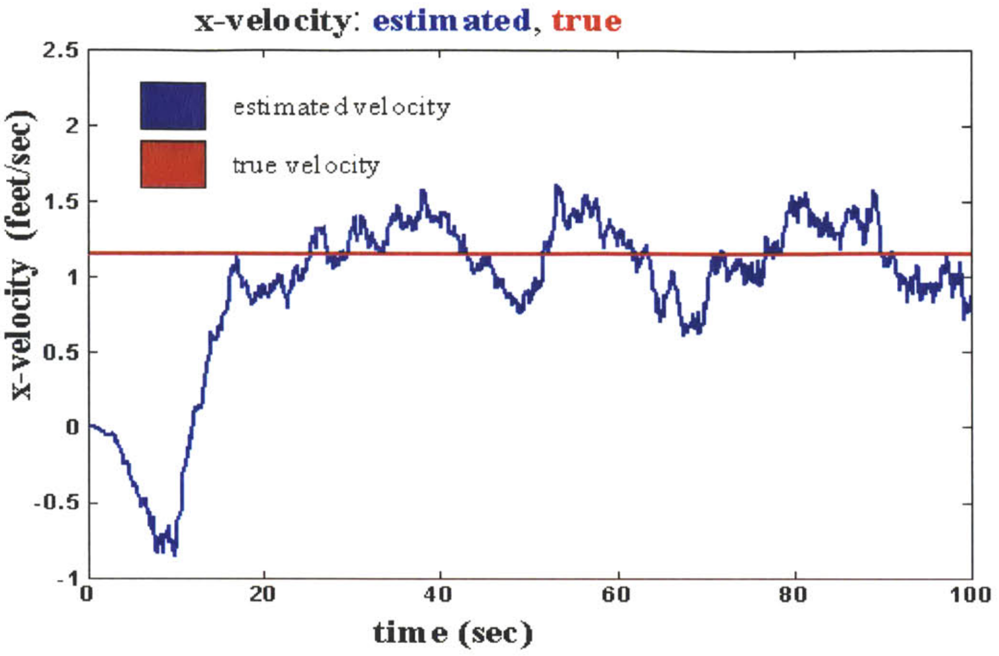

Figure 6-10: Error in the estimate of the y-velocity of a dynamic obstacle

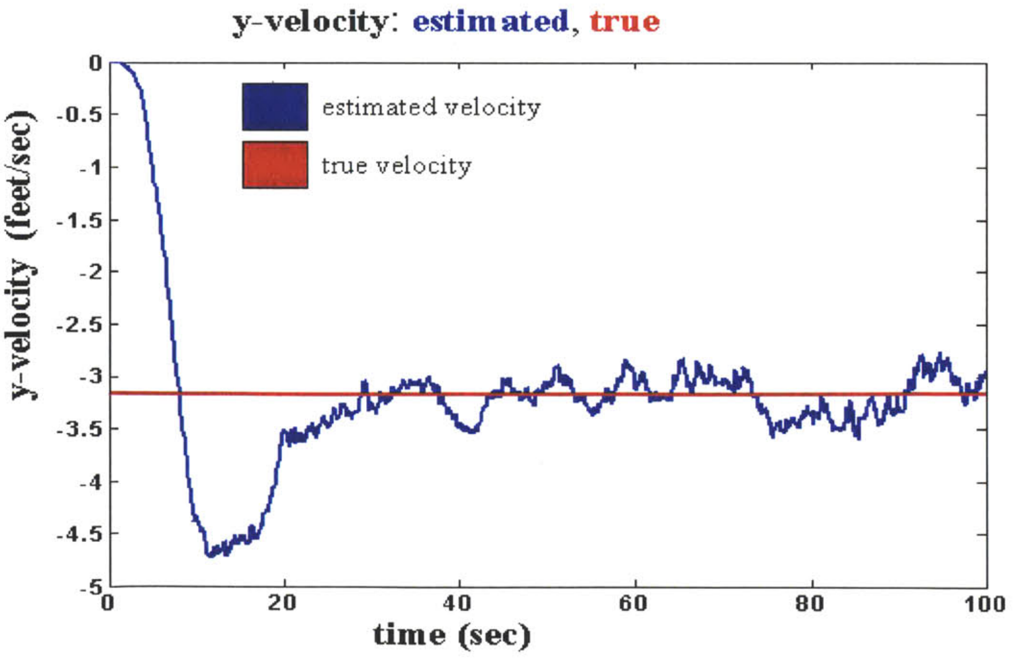

Figure 6-11: Comparison of the estimated and true y-velocity of a dynamic obstacle

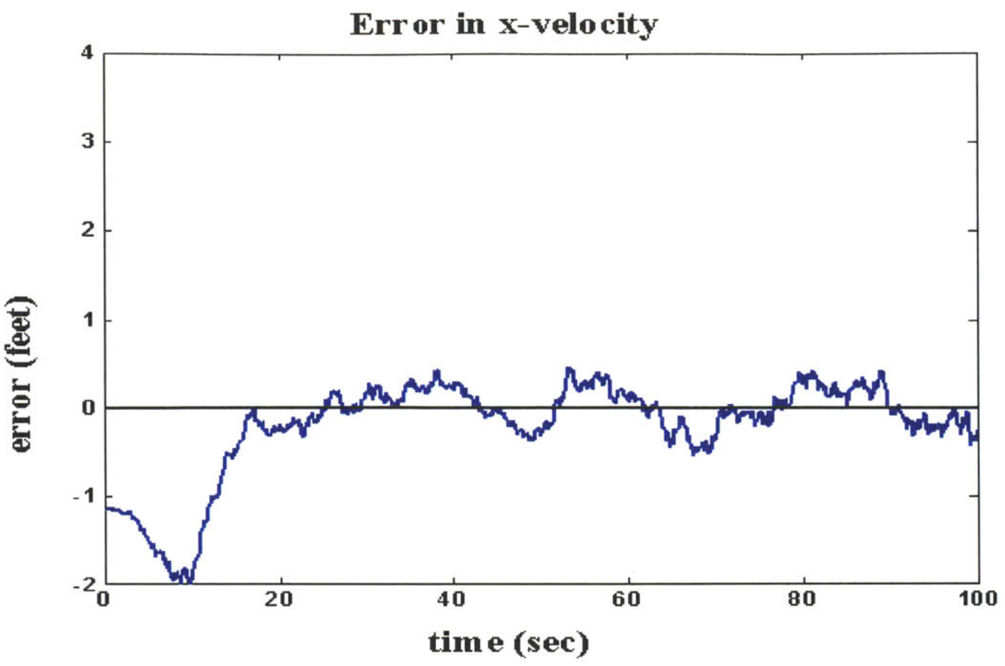

Figure 6-12: Error in the estimate of the x-velocity of a dynamic obstacle

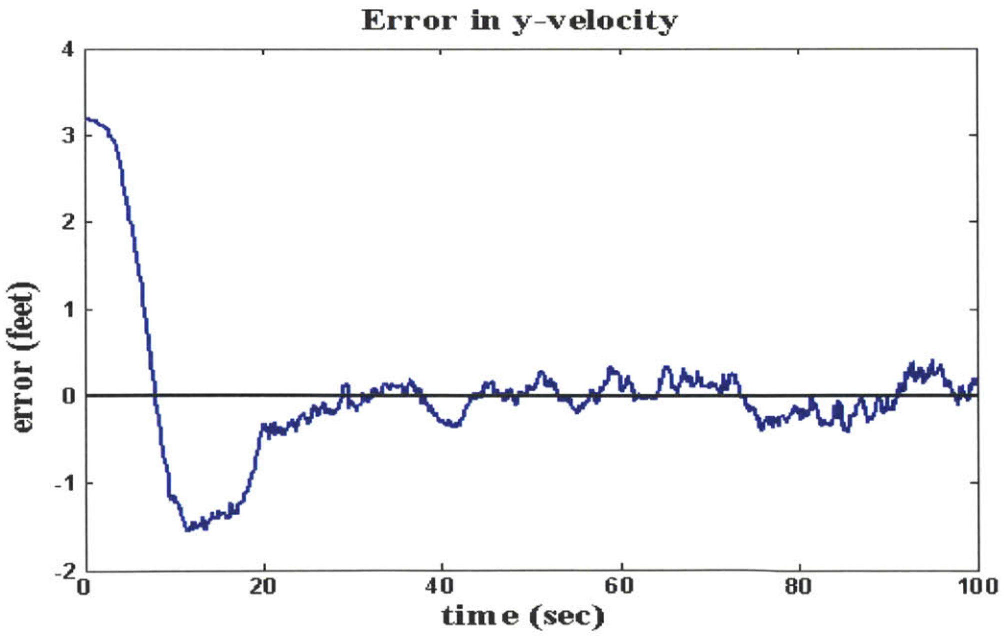

Figure 6-13: Comparison of the estimated and true x-velocity of a dynamic obstacle

6.3 Summary

While it is difficult to quantify the exact efficiency and accuracy of the mapping system, this chapter has demonstrated that the local map can assimilate over time the noisy measurements into an accurate representation of the UUV's observation area and that the resulting global map can store this information efficiently and effectively. Together, the local map and the global map satisfy the requirements for the static obstacle map. While the assumption of constant velocity limits the testing of the dynamic obstacle map, this chapter has shown that this implementation of the dynamic obstacle map works well in this case of constant-velocity targets and is a solid first step toward tracking accelerating or aggressive dynamic obstacles in a UUV's environment. Given the assumptions made in the development of this mapping system, this combination of the two-layered static obstacle map and the dynamic obstacle map forms an integrated on-board mapping system that can be used to represent the UUV's environment and that provides a solid foundation for future extensions to its capability.

7 Conclusion

The objective of this thesis was to design and implement an on-board mapping system that would capture useful and comprehensive information about the UUV's dynamic underwater environment and would represent this information both efficiently and accurately. This chapter presents a summary of the work presented in this thesis and several areas for future research and improvements.

7.1 Thesis Summary

In Chapter 1, we presented the objectives, requirements, and motivations for this thesis. We described mission objectives for a next-generation UUV, and we outlined requirements for a useful mapping system. We defined the scope of this thesis to be the development of an on-board mapping system that should be able to support a variety of mission applications, and we briefly pointed out some of the possible mission applications that could benefit from a useful mapping system.

In Chapter 2, we gave background information on map representations that have been used in mapping systems and other representations that could be applied to mapping systems. We presented advantages and disadvantages of each representation, and we laid the groundwork for the tradeoffs involved in the design decisions for the development of the mapping system in the future chapters.

In Chapter 3, we discussed the sources of noise in a sonar system, and we used these various sources of noise to motivate the use of a probabilistic model for the uncertainty in a sonar measurement. We presented the integration of the sonar measurement uncertainty model into the existing UUV simulation.

In Chapter 4, we presented the design and implementation of the static obstacle map, which stores information about stationary obstacles in the UUV's environment and maintains a measure of confidence in the measurement of the location of the obstacles. The static obstacle map consists of two layers: the local map and the global map. The local map is a uniform height field grid, stores the information in a small observation area in high detail in RAM on-board the vehicle, and allows quick updates, so that at every instant of time the local map reflects the best estimate of the vehicle's current situation in this particular observation area. The global map is a quadtree, stores the information for the entire mission area in lower detail on disk, and allows a compact and efficient storage of the information in the UUV's entire mission area, so as to use as little of the limited memory and computational resources on-board the UUV. We presented algorithms for updating the local map from the sonar inputs and for updating the global map from the information in the local map.

In Chapter 5, we presented the development of the dynamic obstacle map, which stores information about moving threats in the UUV's environment and uses a Kalman filter to estimate the obstacle's position and velocity, given a measurement from the sonar. We assume that the *x*-coordinate and the *y*-coordinate of the obstacle's position are independent of each other, and we filter the state for each coordinate separately.

In Chapter 6, we presented some results of the mapping system. The static obstacle map is shown to accumulate accurate depth measurements over time. The dynamic obstacle map is shown to keep accurate estimates of the position and velocity of the detected targets over time.

In conclusion, the mapping system presented in this thesis forms a useful integrated mapping system that should form a solid foundation on which to build the desired increases in a UUV's autonomy.

7.2 Future Research and Improvements/Extensions

Many issues have not been addressed in this thesis but should be considered for a practical system. There are several key areas for future research.

- **INTEGRATING MAPPING SYSTEM WITH APPLICATIONS SUCH AS PATH PLANNING**
  While care has been taken in this thesis not to design the mapping system to be too specific for any one application, there are advantages to partially integrating the mapping system with the applications which use it. For example, integration with a path-planner may allow the path-planner to evaluate the traversability of entire trajectories instead of single locations. In this case, a measure of traversability would require that the mapping system be coupled to the path planner in some manner, where that the map provides the individual measures and the path planner calculates the partial paths and the corresponding traversability measures.

- **INCORPORATING UNCERTAINTY IN UUV'S OWN LOCATION**
  This thesis assumes that the UUV is fairly certain of its location during the mapping process. However, it is not always possible during certain types of missions for the UUV to surface to receive a true measurement of its position from GPS. The incorporation of

uncertainty in the estimate of the UUV's own position may provide more accuracy in the mapping system.

- **DEALING WITH TIDES AND OTHER TEMPORAL EFFECTS ON THE STATIC OBSTACLES**
  Since the UUV is operating in very shallow waters close to shore, the tidal variations can cause a critical threat to the accuracy of the map over time. The "stationary" obstacles may actually change position with the tide, and this will cause the data in the stationary obstacle map to be blurred.

- **EXTENDING DYNAMIC OBSTACLE MAP TO TRACK AGGRESSIVE CONTACTS**
  Since the UUV simulation models targets as moving linearly and independently of the UUV, the dynamic obstacle map has not been tested in more realistic scenarios, such as targets that are avoiding the UUV in a complicated manner or are chasing the UUV. The ability to track aggressive contacts is important in practical applications of the mapping system.

[This Page Intentionally Left Blank]

Appendix A

Derivation for the distribution for the sample mean and the distribution for the sample variance

Let $(X_1,\ldots,X_n)$ be a random sample of *X*, where *X* has pdf $f_X(x)$. The X_i's are independent and identically distributed with mean μ and variance σ^2. We want to find the distribution for the mean and the distribution for the variance of the random sample of *X*.

A.1 Distribution for the Mean

Let *M* be the random variable that describes the distribution for the mean of the random sample of *X*.

$$M=\tfrac{1}{n}(X_1+X_2+\cdots+X_n)=\tfrac{1}{n}\sum_{i=1}^{n}X_i=\overline{X} \tag{A.1}$$

M is also called the sample mean, since it is an unbiased estimator of the mean of *X*. The mean of *M* is

$$E[M]=E[\tfrac{1}{n}\sum_{i=1}^{n}X_i]=\tfrac{1}{n}\sum_{i=1}^{n}E[X_i]=\tfrac{1}{n}\sum_{i=1}^{n}\mu=\tfrac{1}{n}(n\mu)=\mu \tag{A.2}$$

The variance of *M* is

$$\begin{aligned}\mathrm{var}[M]&=\mathrm{var}[\tfrac{1}{n}\sum_{i=1}^{n}X_i]=\mathrm{var}[\sum_{i=1}^{n}\tfrac{1}{n}X_i]=\sum_{i=1}^{n}\tfrac{1}{n^2}\mathrm{var}[X_i]\\&=\sum_{i=1}^{n}\tfrac{1}{n^2}\sigma^2=n(\tfrac{1}{n^2}\sigma^2)=\tfrac{1}{n}\sigma^2\end{aligned} \tag{A.3}$$

Thus, we have shown that the distribution of the mean has mean μ and variance $\frac{1}{n}\sigma^2$. The derivation above makes no assumptions about the type of distribution for *X*. In the special case where X_i is a Gaussian random variable, then M is also a Gaussian random variable that is completely specified by its mean μ and variance $\frac{1}{n}\sigma^2$.

A.2 Distribution for the Variance

Let S_1^2 be the random variable that describes the distribution for the variance of the random sample of X.

$$\begin{aligned} S_1^2 &= \tfrac{1}{n-1}((X_1-\overline{X})^2+(X_2-\overline{X})^2+\ldots+(X_n-\overline{X})^2) \\ &= \tfrac{1}{n-1}\sum_{i=1}^{n}(X_i-\overline{X})^2 \end{aligned} \tag{A.4}$$

S_1^2 is also called the sample variance, since it is an unbiased estimator for the variance of the random sample of X. To simplify the calculations of the mean and the variance of S_1^2, we introduce the biased estimator S^2.

$$\begin{aligned} S^2 &= \tfrac{1}{n}((X_1-\overline{X})^2+(X_2-\overline{X})^2+\ldots+(X_n-\overline{X})^2) = \tfrac{1}{n}\sum_{i=1}^{n}(X_i-\overline{X})^2 \\ &= \tfrac{1}{n}\sum_{i=1}^{n}(X_i^2-2X_i\overline{X}+\overline{X}^2) = \tfrac{1}{n}\sum_{i=1}^{n}X_i^2-\overline{X}^2 = \tfrac{1}{n}\sum_{i=1}^{n}X_i^2-(\tfrac{1}{n}\sum_{i=1}^{n}X_i)^2 \end{aligned} \tag{A.5}$$

We note that S_1^2 can be expressed in terms of S^2 as

$$S_1^2 = \tfrac{n}{n-1}S^2 \tag{A.6}$$

We will find the mean and the variance of S^2, and we will use these values to find the mean and variance of S_1^2. The mean of S^2 can be calculated as

$$E[S^2] = E[\tfrac{1}{n}\sum_{i=1}^{n} X_i^2 - (\tfrac{1}{n}\sum_{i=1}^{n} X_i)^2] = E[\tfrac{n}{n^2}\sum_{i=1}^{n} X_i^2 - \tfrac{1}{n^2}(\sum_{i=1}^{n} X_i^2 + \sum_{i\neq j} X_i X_j)]$$

$$= E[\tfrac{n-1}{n^2}\sum_{i=1}^{n} X_i^2 - \tfrac{1}{n^2}\sum_{i\neq j} X_i X_j] = \tfrac{n}{n^2}\sum_{i=1}^{n} E[X_i^2] - \tfrac{1}{n^2}\sum_{i\neq j} E[X_i X_j] \qquad \text{(A.7)}$$

$$= \tfrac{n(n-1)}{n^2} E[X_i^2] - \tfrac{n(n-1)}{n^2} E[X_i X_j] = \tfrac{n-1}{n}\{E[X_i^2] - E[X_i X_j]\}$$

Since X_i and X_j are independent and identically distributed,

$$E[X_i X_j] = (E[X_i])^2 \qquad \text{(A.8)}$$

Using equation (A.8), the expression for the mean of S^2 in equation (A.7) simplifies to

$$E[S^2] = \tfrac{n-1}{n}\{E[X_i^2] - (E[X_i])^2\} = \tfrac{n-1}{n}\mu_2 \qquad \text{(A.9)}$$

where μ_2 is the second-order central moment[10], defined as

$$\mu_2 = E[X_i^2] - (E[X_i])^2 \qquad \text{(A.10)}$$

The variance of S^2 is given by

$$\text{var}[S^2] = E[S^4] - (E[S^2])^2 \qquad \text{(A.11)}$$

The second term in equation (A.11) is known from equation (A.9), and we will calculate the first term.

[10] The second-order central moment, μ_2, is also known as the variance, and this notation should not cause confusion with the population mean, μ, for X_i.

$$E[S^4] = E[(S^2)^2] = E[(\tfrac{1}{n}\sum_{i=1}^{n} X_i^2 - (\tfrac{1}{n}\sum_{i=1}^{n} X_i)^2)^2]$$

$$= E[(\tfrac{1}{n}\sum_{i=1}^{n} X_i^2)^2 - 2(\tfrac{1}{n}\sum_{i=1}^{n} X_i^2)(\tfrac{1}{n}\sum_{i=1}^{n} X_i)^2 + (\tfrac{1}{n}\sum_{i=1}^{n} X_i)^4] \qquad \text{(A.12)}$$

$$= \tfrac{1}{n^2}E[(\sum_{i=1}^{n} X_i^2)^2] - \tfrac{2}{n^3}E[(\sum_{i=1}^{n} X_i^2)(\sum_{i=1}^{n} X_i)^2] + \tfrac{1}{n^4}E[(\sum_{i=1}^{n} X_i)^4]$$

We can rewrite the summations in equation (A.12) as

$$(\sum_{i=1}^{n} X_i^2)^2 = \sum_{i=1}^{n} X_i^4 + \sum_{i\neq j} X_i^2 X_j^2 \qquad \text{(A.13)}$$

$$(\sum_{i=1}^{n} X_i^2)(\sum_{i=1}^{n} X_i)^2 = \sum_{i=1}^{n} X_i^4 + \sum_{i\neq j} X_i^2 X_j^2 + 2\sum_{i\neq j} X_i^3 X_j + \sum_{i\neq j\neq k} X_i^2 X_j X_k \qquad \text{(A.14)}$$

$$(\sum_{i=1}^{n} X_i)^4 = \sum_{i=1}^{n} X_i^4 + 3\sum_{i\neq j} X_i^2 X_j^2 + 4\sum_{i\neq j} X_i^3 X_j + 6\sum_{i\neq j\neq k} X_i^2 X_j X_k$$
$$+ \sum_{i\neq j\neq k\neq l} X_i X_j X_k X_l \qquad \text{(A.15)}$$

Substituting equations (A.13), (A.14), and (A.15) for the summations in equation (A.12) and simplifying, we have

$$E[S^4] = \tfrac{1}{n^2}E[\sum X_i^4 + \sum X_i^2 X_j^2] - \tfrac{2}{n^3}E[\sum_{i=1}^{n} X_i^4 + 2\sum_{i\neq j} X_i^3 X_j + \sum_{i\neq j} X_i^2 X_j^2]$$
$$+ \tfrac{1}{n^4}E[\sum_{i=1}^{n} X_i^4 + 4\sum_{i\neq j} X_i^3 X_j + 3\sum_{i\neq j} X_i^2 X_j^2 + 6\sum_{i\neq j\neq k} X_i^2 X_j X_k$$
$$+ \sum_{i\neq j\neq k\neq l} X_i X_j X_k X_l]$$

$$= \tfrac{1}{n^2}\{nE[X_i^4] + n(n-1)E[X_i^2 X_j^2]\}$$
$$- \tfrac{2}{n^3}\{nE[X_i^4] + 2n(n-1)E[X_i^3 X_j] + n(n-1)E[X_i^2 X_j^2]$$
$$+ n(n-1)(n-2)E[X_i^2 X_j X_k]\}$$
$$+ \tfrac{1}{n^4}\{nE[X_i^4] + 4n(n-1)E[X_i^3 X_j] + 3n(n-1)E[X_i^2 X_j^2]$$
$$+ 6n(n-1)(n-2)E[X_i^2 X_j X_k]$$
$$+ n(n-1)(n-2)(n-3)E[X_i X_j X_k X_l]\} \tag{A.16}$$

Again, we note that X_i and X_j are independent and identically distributed. As a result, X_i^2 and X_j^2 are independent (and X_i^3 and X_j are independent, etc.), so we can write

$$E[S^4] = \tfrac{1}{n^2}\{nE[X_i^4] + n(n-1)(E[X_i^2])^2\}$$
$$- \tfrac{2}{n^3}\{nE[X_i^4] + 2n(n-1)E[X_i^3]E[X_i] + n(n-1)(E[X_i^2])^2$$
$$+ n(n-1)(n-2)E[X_i^2](E[X_i])^2\} \tag{A.17}$$
$$+ \tfrac{1}{n^4}\{nE[X_i^4] + 4n(n-1)E[X_i^3]E[X_i] + 3n(n-1)(E[X_i^2])^2$$
$$+ 6n(n-1)(n-2)E[X_i^2](E[X_i])^2 + n(n-1)(n-2)(n-3)(E[X_i])^4\}$$

Further simplification and combining of terms gives

$$\begin{aligned}E[S^4] = \tfrac{1}{n^3}\{&E[X_i^4](n^2-2n+1)+E[X_i^2](E[X_i])^2((6n^2-12n+6)\\&+(-2n^3+6n^2-10n+6))+E[X_i^3]E[X_i](-4n^2+8n-4)\\&+(E[X_i^2])^2(n^3-3n^2+5n-3)+(E[X_i])^4((-3n^2+6n-3)\\&+(n^3-3n^2+5n-3))\}\end{aligned}$$

$$\begin{aligned}= \tfrac{1}{n^3}\{&E[X_i^4](n-1)^2+6E[X_i^2](E[X_i])^2(n-1)^2\\&-2E[X_i^2](E[X_i])^2(n-1)(n^2-2n+3)\\&-4E[X_i^3]E[X_i](n-1)^2+(E[X_i^2])^2(n-1)(n^2-2n+3)\\&-3(E[X_i])^4(n-1)^2+(E[X_i])^4(n-1)(n^2-2n+3)\}\end{aligned}$$

$$\begin{aligned}&= \tfrac{(n-1)^2}{n^3}\{E[X_i^4]-4E[X_i^3]E[X_i]+6E[X_i^2](E[X_i])^2\\&\quad-3(E[X_i])^4\}\\&\quad+\tfrac{(n-1)(n^2-2n+3)}{n^3}\{(E[X_i^2])^2-2E[X_i^2](E[X_i])^2+(E[X_i])^4\}\\&= \tfrac{(n-1)^2}{n^3}\mu_4+\tfrac{(n-1)(n^2-2n+3)}{n^3}\mu_2^2\end{aligned}\tag{A.18}$$

where μ_2 is the second-order central moment defined in equation (A.10), and μ_4 is the fourth-order central moment, defined as

$$\mu_4 = E[X_i^4]-4E[X_i^3]E[X_i]+6E[X_i^2](E[X_i])^2-3(E[X_i])^4 \tag{A.19}$$

Substituting equations (A.9) and (A.18) into equation (A.11), we have

$$\begin{aligned}\mathrm{var}[S^2] &= E[S^4]-(E[S^2])^2 = (\tfrac{(n-1)^2}{n^3}\mu_4+\tfrac{(n-1)(n^2-2n+3)}{n^3}\mu_2^2)-(\tfrac{n-1}{n}\mu_2)^2\\&= \tfrac{(n-1)^2}{n^3}\mu_4+\tfrac{(n-1)(n^2-2n+3)}{n^3}\mu_2^2-\tfrac{n(n-1)^2}{n^3}\mu_2^{\,2} = \tfrac{(n-1)^2}{n^3}\mu_4+\tfrac{(-n^2+4n-3)}{n^3}\mu_2^2\\&= \tfrac{(n-1)^2}{n^3}\mu_4-\tfrac{(n-1)(n-3)}{n^3}\mu_2^2\end{aligned}\tag{A.20}$$

Equations (A.9) and (A.20) are the general expressions for the mean and the variance of S^2. The derivations did not rely on assumptions for the type of distribution for *X*. In the special case where X_i is a Gaussian random variable, we can calculate μ_2 and μ_4, and equations (A.9) and

(A.20) can be simplified. To find μ_2 and μ_4, we need to know the higher-order moments of X_i: $E[X_i]$, $E[X_i^2]$, $E[X_i^3]$ and $E[X_i^4]$. Consider the moment-generating function for a Gaussian random variable X_i with mean μ and variance σ^2.

$$M_{X_i}(s) = \exp(\tfrac{1}{2}\sigma^2 s^2 + \mu s) \tag{A.21}$$

We already know $E[X_i]$, but we will calculate it to be complete. The expectation of X_i is defined as the first moment of $M_{X_i}(s)$ evaluated at *s = 0.*

$$\frac{dM_{X_i}(s)}{ds} = M_{X_i}(s)\{\mu + s\sigma^2\} \tag{A.22}$$

$$E[X_i] = \frac{dM_{X_i}(s)}{ds}\Big|_{s=0} = \mu \tag{A.23}$$

This result is indeed consistent with our knowledge that X_i has mean μ. The expectation of X_i^2 is defined as the second moment of $M_{X_i}(s)$ evaluated at *s = 0.*

$$\frac{d^2M_{X_i}(s)}{ds} = M_{X_i}(s)\{(\sigma^2 + \mu^2) + s(2\sigma^2\mu) + s^2(\sigma^4)\} \tag{A.24}$$

$$E[X_i^2] = \frac{d^2M_{X_i}(s)}{ds}\Big|_{s=0} = \sigma^2 + \mu^2 \tag{A.25}$$

The expectation of X_i^3 is defined as the third moment of $M_{X_i}(s)$ evaluated at *s = 0.*

$$\begin{aligned}\frac{d^3M_{X_i}(s)}{ds} = M_{X_i}(s)\{&(3\sigma^2\mu + \mu^3) + s(3\sigma^4 + 3\sigma^2\mu^2) \\ &+ s^2(3\sigma^4\mu) + s^3(\sigma^6)\}\end{aligned} \tag{A.26}$$

$$E[X_i^3] = \frac{d^3 M_{X_i}(s)}{ds}\Big|_{s=0} = 3\sigma^2\mu + \mu^3 \tag{A.27}$$

The expectation of X_i^4 is defined as the fourth moment of $M_{X_i}(s)$ evaluated at $s = 0$.

$$\begin{aligned}\frac{d^4 M_{X_i}(s)}{ds} &= M_{X_i}(s)\{(3\sigma^4 + 6\sigma^2\mu^2 + \mu^4) + s(12\sigma^4\mu + 4\sigma^2\mu^3) \\ &\quad + s^2(6\sigma^6 + 6\sigma^4\mu^2) + s^3(4\sigma^6\mu) + s^4(\sigma^8)\}\end{aligned} \tag{A.28}$$

$$E[X_i^4] = \frac{d^4 M_{X_i}(s)}{ds}\Big|_{s=0} = 3\sigma^4 + 6\sigma^2\mu^2 + \mu^4 \tag{A.29}$$

Substituting equations (A.23), (A.25), (A.27), and (A.29) into equations (A.10) and (A.19), we have

$$\mu_2 = (\sigma^2 + \mu^2) - (\mu)^2 = \sigma^2 \tag{A.30}$$

$$\begin{aligned}\mu_4 &= (3\sigma^4 + 6\sigma^2\mu^2) - 4(\sigma^2\mu + \mu^3) + 6(\sigma^2 + \mu^2)(\mu)^2 - 3(\mu)^4 \\ &= 3\sigma^4\end{aligned} \tag{A.31}$$

Using equations (A.30) and (A.31), equation (A.9) simplifies to

$$E[S^2] = \tfrac{n-1}{n}\sigma^2 \tag{A.32}$$

and equation (A.20) becomes

$$\begin{aligned}\mathrm{var}[S^2] &= \tfrac{(n-1)^2}{n^3}\mu_4 - \tfrac{(n-1)(n-3)}{n^3}\mu_2^2 = \tfrac{(n-1)^2}{n^3}(3\sigma^4) - \tfrac{(n-1)(n-3)}{n^3}(\sigma^2)^2 \\ &= \tfrac{1}{n^3}\{(n^2 - 2n + 1)(3\sigma^4) - (n^2 - 4n + 3)(\sigma^4)\} = \tfrac{1}{n^3}\sigma^4(2n^2 - 2n) \\ &= \tfrac{2(n-1)}{n^2}\sigma^4\end{aligned} \tag{A.33}$$

Now, we can finally find the mean and the variance of S_1^2, assuming that X is Gaussian. Using equation (A.6), the mean of S_1^2 is

$$E[S_1^2] = E[\tfrac{n}{n-1} S^2] = \tfrac{n}{n-1} E[S^2] = \tfrac{n}{n-1}(\tfrac{n-1}{n}\sigma^2) = \sigma^2 \qquad \text{(A.34)}$$

and the variance of S_1^2 is

$$\text{var}[S_1^2] = \text{var}[\tfrac{n}{n-1} S^2] = (\tfrac{n}{n-1})^2 \text{ var}[S^2] = \tfrac{n^2}{(n-1)^2}(\tfrac{2(n-1)}{n^2}\sigma^4) = \tfrac{2}{(n-1)}\sigma^4 \qquad \text{(A.35)}$$

Thus, assuming that X_i is a Gaussian random variable, the distribution of the variance is a Gaussian random variable with mean σ^2 and variance $\frac{2}{(n-1)}\sigma^4$.

[This Page Intentionally Left Blank]

References

[1] Abramson, M., M. Adams, H. Leung, D. Moe, and R. Powers. “Obstacle Avoidance Planner Development System.” Technical Report CSDL-R-2346. Charles Stark Draper Laboratory, Inc., July 1991.

[2] Boult, T.E. “Updating Distance Maps When Objects Move.” SPIE Vol. 852, Mobile Robots II, 1987.

[3] Burgard, W., D. Fox, D. Hennig, and T. Schmidt. “Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids.” In Proc. Fourteenth National Conference on Artificial Intelligence (AAAI-96), 1996.

[4] Chen, D.Z., R.J. Szczerba, J.J. Uhran, Jr. “Using Framed-Quadtrees to Find Conditional Shortest Paths in an Unknown 2-D Environment.” Technical Report 95-2. Department of Computer Science and Engineering, University of Notre Dame, January 1995.

[5] Chen, D.Z., R.J. Szczerba, J.J. Uhran, Jr. “Using Framed-Octrees to Find Conditional Shortest Paths in an Unknown 3-D Environment.” Technical Report 95-9. Department of Computer Science and Engineering, University of Notre Dame, 1995.

[6] Dunn, P. “Navy UUV Master Plan.” Proc. of the International Unmanned Undersea Vehicles Symposium, Newport, RI, April 2000.

[7] Elfes, A. “Sonar-Based Real-World Mapping and Navigation.” IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, pp. 249-265, June 1987.

[8] Etter, P. *Underwater Acoustic Modeling*, Second Edition, University Press, Cambridge, UK, 1996.

[9] Feder, H.J.S., J.J. Leonard, and C.M. Smith. “Adaptive Mobile Robot Navigation.” Int. Journal of Robotics Research, Vol. 18, No. 7, pp. 650-668, July 1999.

[10] Fenwick, J.W. “Collaborative Concurrent Mapping and Localization.” Master of Science Thesis. Massachusetts Institute of Technology, May 2001.

[11] Fox, D., W. Burgard, and S. Thrun. “Markov Localization for Mobile Robots in Dynamic Environments.” Journal of Artificial Intelligence Research, Vol. 11, pp. 391-427, 1999.

[12] Fraichard, T. “Trajectory Planning in Dynamic Workspace: a ‘State-Time Space’ Approach.” Technical Report 3545. Institut National de Recherche en Informatique et en Automatique, October 1998, Submitted in January 1998 to RSJ Advanced Robotics. Available at: http://citeseer.nj.nec.com/fraichard98trajectory.html.

[13] Fujimura, K. and H. Samet. “A Hierarchical Strategy for Path Planning among Moving Obstacles.” IEEE Transactions on Robotics and Automation, Vol. 5, No. 1, pp. 61-69, February 1989.

[14] Garland, M. and P.S. Heckbert. "Fast Polygonal Approximation of Terrains and Height Fields." Technical Report CMU-CS-95-181. Carnegie Mellon University, Pittsburgh, PA, 1995. Available at: http://www.cs.cmu.edu/~garland/scape.

[15] Kambhampati, S. and L.S. Davis. "Multiresolution Path Planning for Mobile Robots." IEEE Robotics and Automation, Vol. RA-2, No. 3, September 1986.

[16] Kendall, M. G. and A. Stuart. *The Advanced Theory of Statistics, Vol. 1: Distribution Theory*, Third Edition, Charles Griffin & Company, Ltd., London, 1969.

[17] Kunz, C., T. Willeke, and I. Nourbakhsh. "Automatic Mapping of Dynamic Office Environments." In Proc. IEEE International Conference on Robotics and Automation, Vol. 2, pp. 1681-1687, 1997. Available at: http://www.cs.cmu.edu/~illah/PAPERS/inductobeast.pdf.

[18] Leonard, J.J and H.F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*, Kluwer Academic Publishers, Norwell, MA, 1992.

[19] Levy, L. J. "The Kalman Filter: Navigation's Integration Workhorse." Available at: http://www.cs.unc.edu/~welch/kalman/Levy1997/index.html.

[20] Lopez-Sanchez, M., G. Sukhatme, and G.A. Bekey. "Mapping an Outdoor Environment for Path Planning." Research Report 98-683, University of Southern California, June 1998. Available at: http://www.usc.edu/dept/cs/tech.html.

[21] Martin, M.C. and H.P. Moravec. "Robot Evidence Grids." Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, 1996.

[22] Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Vol. 1, Academic Press, 1979. Available at: http://www.cs.unc.edu/~welch/kalman/maybeck.html.

[23] McKeever, S. D. "Path Planning for an Autonomous Vehicle." Master of Science Thesis, Massachusetts Institute of Technology, June 2000.

[24] Moravec, H. P. "Sensor Fusion in Certainty Grids for Mobile Robots." In A. Casals, editor, Sensor Devices and Systems for Robotics, NATO ASI Series, Vol. 52, pp. 253-276, 1989.

[25] Moravec, H.P. and A. Elfes. "High Resolution Maps from Wide Angle Sonar." In Proc. IEEE International Conference on Robotics and Automation, pp. 116-121, 1985.

[26] Paglia, Jr., J. and W. Wyman. "DARPA's Autonomous Minehunting and Mapping Technologies (AMMT) Program: An Overview." Technical Report CSDL-97-010, Charles Stark Draper Laboratory, Inc., 1996.

[27] Ricard, M.J. and M. Keegan. "Intelligent Autonomy for the Manta Test Vehicle." In Proc. of the IEEE Oceanic Engineering Society, Oceans 2000 Conference, Providence, RI, September 2000.

[28] Ricard, M.J. "Mission Planning for an Autonomous Undersea Vehicle: Design and Results." Proceedings of the Technology and the Mine Problem Symposium, Monterey, CA, November 1996.

[29] Samet, H. "An Algorithm for Converting Rasters to Quadtrees." IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 1, January, 1981.

[30] Samet, H. "Distance Transform of Images Represented by Quadtrees." IEEE Transactions Pattern Analysis and Machine Intelligence, Vol. 4, pp. 298-303, 1982.

[31] Samet, H. "An Overview of Quadtrees, Octrees, and Related Hierarchical Structures." In Earnshaw, R. A. Theoretical Foundations of Computer Graphics and CAD, NATO ASI Series, Vol. F40, pp. 51-67, 1988.

[32] Samet, H. "Spatial Data Structures." In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability, and Beyond*, pp. 361-385, Addison Wesley/ACM Press, Reading, MA, 1995. Available at: http://www.cs.umd.edu/~hjs/pubs/kim.pdf

[33] Sammon, R. "A Mapping System for an Autonomous Helicopter." Master of Science Thesis, Massachusetts Institute of Technology, June 1999.

[34] Simon, D., "Kalman Filtering." Embedded Systems Programming, pp. 72-79, June 2001.

[35] Steiner, S. "Mapping and Sensor Fusion for an Autonomous Vehicle." Master of Science Thesis, Massachusetts Institute of Technology, June 1996.

[36] Stentz, A. and Hebert, M. "A Complete Navigation System for Goal Acquisition in Unknown Environments." In Autonomous Robots, Vol. 2, No. 2, August, 1995.

[37] Stentz, A. "Map-Based Strategies for Robot Navigation in Unknown Environments." In Proc. AAAI 96 Spring Symposium on Planning with Incomplete Information for Robot Problems, 1996.

[38] Stewart, W. K., "Three-Dimensional Stochastic Modeling Using Sonar Sensing for Undersea Robotics." Autonomous Robots, Vol. 3, pp. 121-143, Kluwer Academic Publishers, 1996.

[39] Thrun, S. and A. Buecken. "Integrating Grid-based and Topological Maps for Mobile Robot Navigation." Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence, 1996. Available at: http://www.ri.cmu.edu/pubs/pub_647.html.

[40] Urick, R. J. *Principles of Underwater Sound*, McGraw-Hill, New York, 1983.

[41] Welch, G. and G. Bishop. "An Introduction to the Kalman Filter." Technical Report TR 95-041. Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, March 2002. Available at: http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.

[42] Yahja, A., A. Stentz, S. Singh, and B.L Brumitt. "Framed-Quadtree Path Planning for Mobile Robots Operating in Sparse Environments." In Proc. IEEE Conference on Robotics and Automation, (ICRA), Leuven, Belgium, May 1998.

[43] Yahja, A., S. Singh, and A. Stentz. "An Efficient On-line Path Planner for Outdoor Mobile Robots." Robotics and Autonomous Systems, Vol. 32, pp. 129-143, 2000.

[44] Zelinsky, A. "A Mobile Robot Exploration Algorithm." IEEE Transactions on Robotics and Automation, Vol. 8, No. 6, December 1992.

[45] http://www.reson.com/glossidx.htm

[46] http://www.chartmaker.ncd.noaa.gov/hsd/specs/CHAPTER5.pdf

4470 - 58