

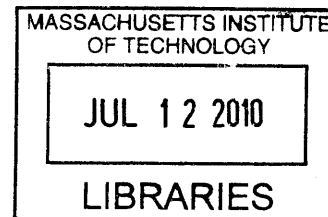
Toward autonomous harbor surveillance

by

Hordur Johannsson

B.S. University of Iceland (2002)

M.S. University of Iceland (2006)



Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Science


at the

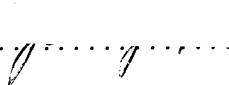
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

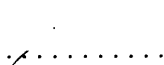
June 2010

ARCHIVES

© Massachusetts Institute of Technology 2010. All rights reserved.

Author 
Department of Electrical Engineering and Computer Science
May 21, 2010

Certified by 
John Leonard
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by 
Terry P. Orlando
Professor of Electrical Engineering and Computer Science
Chairman, Department Committee on Graduate Theses

Toward autonomous harbor surveillance

by

Hordur Johannsson

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In this thesis we address the problem of drift-free navigation for underwater vehicles performing harbor surveillance and ship hull inspection. Maintaining accurate localization for the duration of a mission is important for a variety of tasks, such as planning the vehicle trajectory and ensuring coverage of the area to be inspected. Our approach uses only onboard sensors in a simultaneous localization and mapping setting and removes the need for any external infrastructure like acoustic beacons. We extract dense features from a forward-looking imaging sonar and apply pair-wise registration between sonar frames. The registrations are combined with onboard velocity, attitude and acceleration sensors to obtain an improved estimate of the vehicle trajectory. In addition, an architecture for a persistent mapping is proposed. With the intention of handling long term operations and repetitive surveillance tasks. The proposed architecture is flexible and supports different types of vehicles and mapping methods. The design of the system is demonstrated with an implementation of some of the key features of the system. In addition, methods for re-localization are considered. Finally, results from several experiments that demonstrate drift-free navigation in various underwater environments are presented.

Thesis Supervisor: John Leonard

Title: Professor of Mechanical Engineering

Acknowledgments

How quickly time passes. It feels like only yesterday, that rainy August day, when we arrived at our new home here in Boston two years ago. This thesis is the result of the work I have pursued during this time. There are many people, that have made MIT and Cambridge such a wonderful place, and many, that without, this work would never have been possible.

First, I would like to thank my advisor, John Leonard, who has been so helpful and understanding, going out of his way to make my stay here as pleasant as it has been. His insight and extensive background knowledge have been an invaluable resource. I thank him for his many suggestions on this work, and his encouragement to always reach further.

To my friends and collaborators on the HAUV project, Michael Kaess and Brendan Englot, I owe a great gratitude. Sitting out for trials in our lab, trudging down to the river on numerous occasions, carrying the HAUV around like it was our baby, and sitting out a storm in Norfolk. Without them this work would not have been possible. Especially I would like to thank, Michael, for listening and answering my endless questions, much of what I have learned in the last couple of years I owe to him. In addition, I am really grateful to Franz Hover, who welcomed me to participate with his group, and for his sound advice on my research. I have also had numerous fruitful discussions with Ryan Eustice, who is one of our collaborators on the ship hull inspection project.

When operating an autonomous vehicle, the difference between success and failure can be a sharp edge, indeed, so it is really nice to have professionals on your side. It has been a pleasure working with the people at Bluefin Robotics, especially Jerome Vaganay, Kim Shurn and Mike Elkins; their preparedness and quick problem solving skills have made our operations together successful.

I am also grateful to the sponsors of this work, which is supported by the Office of Naval Research under Grant N00014-06-10043, monitored by Dr. T.F. Swean. I would also like to thank Peter Biber and Tom Duckett for providing us access to their

long term dataset.

My time here at MIT has been truly wonderful, that is in big part to all the great people in the Marine Robotics group, Aisha, Been, Georgios, Jacques, John, Maurice, Mike and Hunter. I am also really grateful to Marcia Munger, for keeping us fed at our weekly meetings, helping with reimbursements and numerous other tasks.

Last, but not least, I would like to thank my family. My mother for always supporting me and encouraging me on any endeavor I have pursued. My dear daughter, little inventive Bjarney, for inspiring me and putting a smile on my face every day. Especially I would like to thank my loving wife, Gyda, for encouraging me to start on this journey. I do not know how I can start to thank her for being willing to leave everything behind, cross the ocean and start a new home here in Cambridge.

Thank you,

Hordur

Contents

1	Introduction	17
1.1	Challenges in underwater navigation	17
1.2	Underwater harbor surveillance	20
1.2.1	Piers, seabed and other structures	20
1.2.2	Ship hull inspection	22
1.3	Overview of navigation methods	22
1.3.1	Dead reckoning	23
1.3.2	Navigation using a prior map	24
1.3.3	Navigation with an unknown map	26
1.4	Simultaneous localization and mapping	27
1.5	Overview	29
2	Acoustic imaging	31
2.1	Sonar overview	31
2.2	Vehicle state	36
2.3	Imaging sonar geometry	37
2.4	Summary	42
3	Sonar-aided navigation	43
3.1	Feature extraction	45
3.2	Registration	47
3.3	Map estimate	48
3.4	Implementation	50

3.5	Results	52
3.5.1	Tank	53
3.5.2	Charles River	55
3.5.3	King Triton	58
3.6	Summary	61
4	Architecture for persistent mapping	65
4.1	The map database	66
4.1.1	Assumptions and rationalization	69
4.2	Requirements	70
4.2.1	Core features	70
4.2.2	Services	72
4.3	Design	73
4.3.1	Data model	73
4.3.2	Execution model	77
4.4	Implementation and applications	78
4.4.1	LCM adapter	79
4.4.2	Map estimate	79
4.4.3	Laser processing	80
4.4.4	Examples	80
4.5	Summary	87
5	Re-localization	89
5.1	Features	89
5.2	Comparison of different places	90
5.3	Fixed radius nearest neighbor search	92
5.3.1	KD tree	92
5.3.2	BBD tree	92
5.3.3	Results	93
5.4	Customized search in database systems	97
5.5	Summary	97

6 Conclusion	99
6.1 Summary of contribution	99
6.2 Future work	100
6.2.1 Sonar-aided navigation	100
6.2.2 Long term navigation	102

List of Figures

1-1	Bluefin-MIT Hovering Autonomous Underwater Vehicle (HAUV) . . .	18
1-2	Objects lying on the riverbed in viewed with a DIDSON sonar	21
1-3	A side-scan sonar image inside a harbor	21
1-4	Different sensing modalities.	22
1-5	Viewing a ship hull with a camera.	23
2-1	900Khz side-scan sonar image	32
2-2	How sonar images are captured	33
2-3	Images from the MIT Sailing Pavilion	33
2-4	<i>R/V Oceanus</i> images	34
2-5	<i>King Triton</i> profile image	35
2-6	<i>King Triton</i> ship hull 3D reconstruction	35
2-7	Coordinate system	36
2-8	Imaging sonar geometry	37
2-9	Translation error	40
3-1	Sonar feature extraction	46
3-2	Sonar registration	47
3-3	Bayes net formulation of the estimation problem	49
3-4	Overview of the sonar-aided navigation system.	51
3-5	The HAUV operating in the tank.	53
3-6	Evaluation of the tank experiment	54
3-7	Evaluation of the Charles River experiment	56
3-8	Sonar map of the MIT Sailing Pavilion	57

3-9	Operations at the MIT Sailing Pavilion in the Charles River	59
3-10	Evaluation of the experiment on the <i>King Triton</i> vessel	60
3-11	Difference between heading estimates	61
3-12	Operations on the <i>King Triton</i> vessel	62
4-1	Effects of marginalization	66
4-2	System overview of the Map DB.	74
4-3	Overview of the Map DB data model	75
4-4	Overview of the Map DB execution model	77
4-5	Sonar map of the MIT Sailing Pavilion.	83
4-6	Comparing sonar maps from dead-reckoning and smoothing.	84
4-7	Map database laser scan examples	85
5-1	Example of laser scans and extracted features	91
5-2	Comparison of partitioning using KD-Tree vs BBD-Tree	93
5-3	Killian court dataset	94
5-4	Comparison of search times	95
5-5	Comparison of search times by dimension	96
5-6	Looking at execution time for different dimensions and denser data	96
6-1	A liquefied natural gas carrier	101

List of Tables

3.1	Overview of the results from our experiments.	52
4.1	Overview of SLAM algorithms.	68

List of Acronyms

GPS global positioning system

DVL Doppler velocity log

DIDSON dual frequency identification sonar

LBL long baseline acoustic navigation

RLG ring laser gyro

FOG fiber optic gyro

AUV autonomous underwater vehicle

HAUV hovering autonomous underwater vehicle

ESEIF exactly sparse extended information filter

SLAM simultaneous localization and mapping

EKF extended Kalman filter

EIF extended information filter

SEIF sparse extended information filter

ESDF exactly sparse delayed-state filter

SAM smoothing and mapping

iSAM incremental smoothing and mapping

LORAN-C long range navigation

USBL ultra short base line

UKF unscented Kalman filter

NDT normal distribution transform

ROV remotely operated vehicle

MRF Markov random field

DBMS database management system

BBD balanced box-decomposition

SQL standard query language

GiST generalized search tree

Chapter 1

Introduction

Many underwater structures and areas need to be routinely inspected for maintenance and security purposes. Including pier pilings, harbor seafloor, pipelines, oil platforms and ships. To address this need, Bluefin Robotics and MIT built a ship hull inspection vehicle (see Fig. 1-1), called the Hovering Autonomous Underwater Vehicle (HAUV) [99]. The HAUV is equipped with a Doppler velocity log (DVL) to measure velocity relative to a surface, a ring laser gyro (RLG) for attitude measurements and a dual frequency identification sonar (DIDSON) [5] for imaging the structures being inspected.

1.1 Challenges in underwater navigation

Accurately keeping track of the vehicle position is crucially important, but difficult in harbor environments in particular. Full coverage of the area being inspected is required, as is avoiding obstacles, restricted areas, and reporting accurate location estimate for detected targets. It is difficult, however, to obtain a global position estimate underwater from an external source. The global positioning system (GPS) is only available at the surface, so for accurate positioning, acoustic beacons would need to be deployed. Moreover, a magnetic compass works poorly near large metal structures. Integrating a rate gyro and odometry, will cause sensor errors to accumulate overtime and the position estimate will drift. Using time of flight measurements

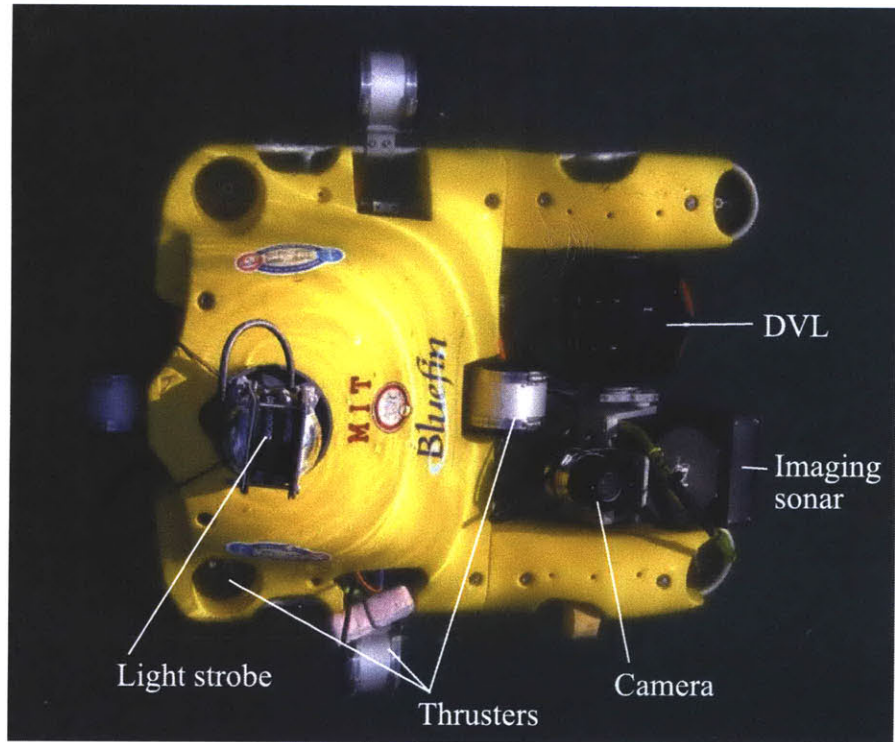


Figure 1-1: Top view of the Bluefin-MIT Hovering Autonomous Underwater Vehicle (HAUV). The vehicle is equipped with a Doppler velocity log (DVL), an imaging sonar, an optical camera and a light strobe. The sonar and DVL can be actuated independently to optimally align the sensors to the surface being inspected.

with acoustic beacons has been commonly used in underwater navigation [105–107] to obtain a global position estimate. This has also proved successful in various applications like underwater archeology [72] and ship hull inspection [42]. Given the above it is of interest to provide drift-free navigation, by identifying and aligning previously visited areas using the on-board imaging sonar, combined with dead reckoning from the vehicle’s other sensors.

Surveillance tasks tend to be repetitive, e.g. each time a cruise ship comes to berth. This fact shows that it is beneficial to: i) store information about previous missions, and ii) localize relative to this prior information. In this thesis we look at both of these issues, considering an architecture for persistent mapping to store and utilize data from previous missions. This system can then be used by a re-localization algorithm, so that the vehicle can reuse the information while executing a subsequent

mission. By staying localized relative to previous missions it is possible to re-use data from those missions, e.g. go back to previously visited locations and verify if objects have been cleared. In the harbor environment it would be possible to initialize the location using a GPS, the vision based localization could then provide a more precise relative location to the dataset. Another benefit of vision aided navigation would be when performing ship-hull inspection because the ship might not be at the same location each time.

Augmenting vehicle localization using sonars has been undertaken in a number of prior works. Walter et al. [102] used manually extracted landmarks, and later automatic feature detection [103] with the exactly sparse extended information filter (ESEIF) to produce a map of an underwater ship hull. An automatic feature detector and a landmark formulation using an extended Kalman filter (EKF) filter was used in [22]. Sekkati et al. used extracted corner features from dual frequency identification sonar (DIDSON) frames to estimate vehicle motion over several frames [87]. In related work Negahdaripour et al. combined the DIDSON with an optical camera for 3-D target reconstruction using opti-acoustic stereo [76]. Eustice et al. [24] used constraints from overlapping camera frames within a simultaneous localization and mapping (SLAM) information filter to estimate vehicle pose. In work by Folkesson et al. [32], a forward-looking sonar was used with a prior map to track features in sonar images and use them to localize the vehicle. A full 360-degree sonar scanner has been used in partially structured underwater environments [84] for localization, by tracking line features in the environment using an EKF for the estimation process. Mallios et al. recently showed promising results in [68] using a mechanical scanning sonar and scan matching in an EKF framework.

Here, a pose graph formulation is used to combine onboard navigation information with sonar registration. Pose graphs [63, 81] represent a map of the environment as a graph, where the nodes are variables representing the vehicle states along its trajectory and edges are constraints between those variables. Efficient on-line methods have been developed to optimize this pose graph, such as incremental smoothing and mapping (iSAM) [53]. This is the approach used for the present work.

Our main contributions are:

1. An automated dense feature extraction technique that allows for effective registration and loop closures.
2. A real-time implementation of the sonar-aided navigation system.
3. Experiments in various underwater environments with the HAUV vehicle.
4. An infrastructure to support long term operations and repetitive mapping.

In this thesis, the focus is on using imaged areas that are roughly flat and horizontal, like the large open areas of ship hulls, and the seafloor. Most importantly, our system allows for drift-free navigation without depending on any external infrastructure. The dense feature extraction allows us to detect and use a wide range of objects for this purpose, and is particularly useful to disambiguate between different places when searching for loop closures. Large areas having few features can be dealt with, because the sensor drift of the vehicle used is fairly small. These scenarios are brought out in our analysis of the testing performed to date.

1.2 Underwater harbor surveillance

Autonomous underwater harbor surveillance is the prime motivation for this work. There are many tasks to consider: inspection of harbor structures, monitoring the harbor bottom and inspecting vessels operating in the harbor. The purpose of those tasks is to improve the security and safety of the harbor area.

1.2.1 Piers, seabed and other structures

Most harbors have to be dredged so the water depth is sufficient to allow ships safe access to the harbor. It would be beneficial if a vehicle could routinely navigate around the harbor area and collect bathymetry data that could then be compared with the intended depth limits inside the harbor. Over time these maps could be compared to identify troublesome areas. The depth limits could be violated because

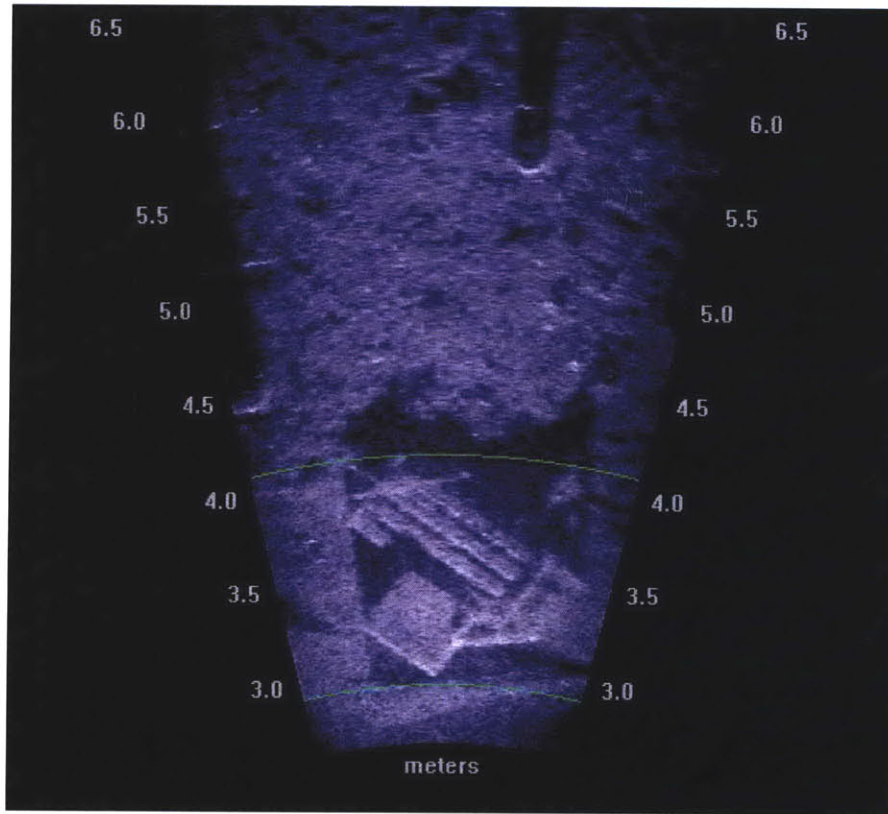


Figure 1-2: Objects lying on the riverbed beneath the pier at the MIT Sailing Pavilion in the Charles River. The image is taken from a DIDSON imaging sonar.

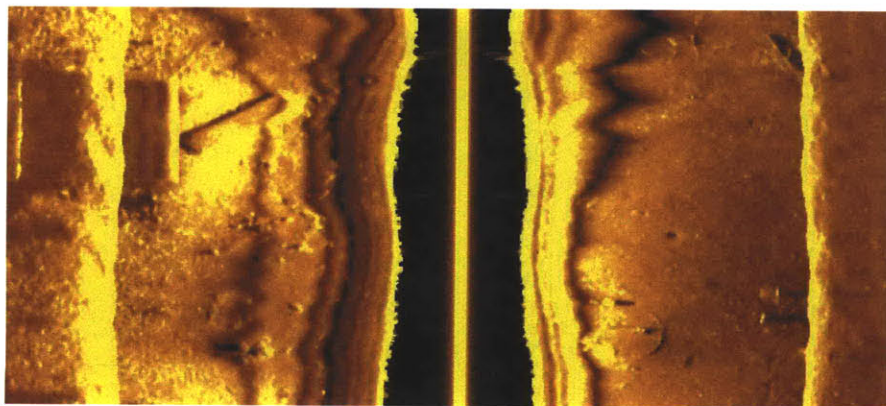
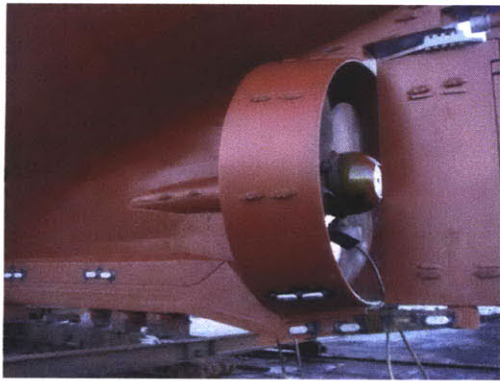


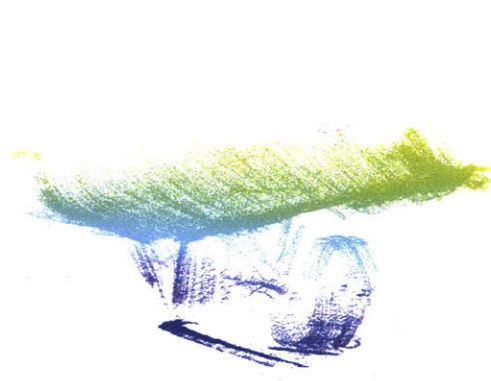
Figure 1-3: A side-scan sonar image inside a harbor. The vehicle is located in the center of the image while traveling vertically and viewing the harbor on both sides. On the port side a pier pillar and some rubble are visible.

of mistakes in the dredging operation, objects being dropped in the shipping lanes or other similar reasons. Inspection might also be needed for security reasons, verifying that there are no unexpected objects lying on the harbor bottom. An example of things one might look for on the bottom can be seen in Figures 1-2 and 1-3.

1.2.2 Ship hull inspection



(a) Camera image of a ship's stern while in drydock



(b) Sonar profile of a ship's stern

Figure 1-4: Different sensing modalities.

Another task that might be considered as a part of the harbor surveillance mission is inspection of the ships. This can include a visual inspection of the ship using different types of sensors like a sonar and a camera, with the purpose of identifying if any foreign objects are on the hull, checking if there are any damages and assessing the general condition of the hull. Figure 1-4(b) shows 3D reconstruction of the stern of *R/V Oceanus*, the data was collected with the DIDSON sonar in profiling mode. In Figure 1-4(a) is a picture of a similar stern, while in dry dock. Cameras can also be used, two camera images of the *R/V Oceanus* are shown in Figure 1-5. This data was collected with the HAUV.

1.3 Overview of navigation methods

This section gives an overview of the different navigation methods that are commonly used and in particular methods for underwater navigation. As mentioned before, when

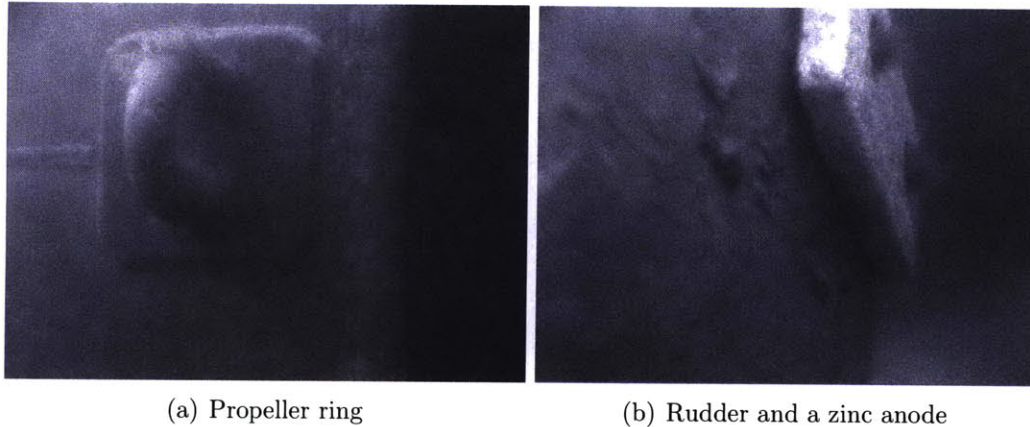


Figure 1-5: Viewing a ship hull with a camera. (a) Shows the side of a propeller ring. (b) Shows zinc anode attached to the ship's rudder.

localizing underwater, access to an external absolute positioning system is not readily available. Because of this, the methods commonly used in underwater navigation are: dead-reckoning, deployment of a positioning system in the form of acoustic beacons, and/or using the vehicle's external sensors to aid the navigation. This last option is what is considered in more detail in this thesis. Given a prior map of the world it is possible to localize relative to that map. On the other hand, if the map is unknown SLAM can be used to estimate both the map and the vehicle pose. There has been extensive work in SLAM over the last couple of decades [18, 24, 31, 62, 63, 73, 89, 96, 97, 102], and in recent years it has become feasible to use SLAM in real-time for relatively large scale problems [9, 10, 17, 53, 58, 79, 81, 82, 88].

1.3.1 Dead reckoning

Dead reckoning has been used by mariners for centuries [16] to keep track of their position. Using the last well known position, the course, speed and time a new estimate of the vessel's position can be estimated. Early mariners relied often on landmark sightings to determine their position and heading, meaning that many did not venture to far from shore. Though there were some that used the stars, including the Sun as aids for finding their heading [101]. The invention of the magnetic compass [44] made it easier to navigate across the world's oceans without sight to land.

The magnetic compass is still used today. In addition to the magnetic compass there are gyro compasses. The most advanced gyros are north-seeking RLG and fiber optic gyro (FOG), which can give a very accurate bearing measurement to true north. They both use the same principle of measuring the time a light travels a known distance, that way they can accurately measure angular velocities. The RLG uses mirrors to guide the light from the transmitter to the receiver while the FOG user fiber optic cable. The RLG has difficulties for slow velocities, and must be vibrated at a known frequency, this is overcome in the FOG [13].

Early methods to measure speed were based on throwing a piece of wood in the water and seeing how far it drifted away in a fixed period of time [41]. This was later improved with the chip log [12] or simply the log, where the float was attached to a string with series of knots. Then the speed was determined by the number of knots passed out of the reel. Even today, ship's speed is measured in knots which is one nautical mile per hour. Modern methods have improved on the chip log and other types of logs have been introduced. Like the pit log [92], which translates an engine RPM into speed through water. The DVL is yet another sensor to estimate a vessel's velocity. The velocity is measured by transmitting an acoustic signal and then measuring the Doppler shift of the incoming signal. Using that it is possible to calculate the relative velocity between the target and the transducers [46]. A DVL can give very accurate velocity measurements ($\pm 0.2\%$ cm/s precision) [104]. If the water is shallow then the DVL can measure velocity relative to ground compared to velocity through water that the other methods measured. The actual range of the DVL will vary with frequency. By measuring velocity over ground it is possible to compensate for errors caused by water currents. The DVL can also be used to measure velocity through water, but not as accurately. The water velocity can be measured because particles in the water reflect some of the acoustic signal.

1.3.2 Navigation using a prior map

When using dead reckoning, inaccuracies in the measurements will inevitably cause the position estimate to drift over time. As mentioned before, landmarks can be used

to navigate along the coastline: a fisherman sails out the fjord until two of the peaks on the mountain range align up, knowing that he has reached good fishing grounds [16]. Celestial navigation has been used by mariners through the ages. It relies on using the Sun, the Moon and other stars in the sky to determine your position. With a known model of the celestial objects, the earth and time, it is possible to measure the elevation of the objects from the horizon to acquire a position fix. The elevation can be measured with a sextant. Even today this is a required skill for any offshore navigators, and nautical almanacs with tables for celestial navigation are published every year. Using a sextant it is possible to estimate a vessels position with around one mile accuracy [29].

Using similar principles it is possible to not only use maps of natural landmarks to navigate, but introduced artificial landmarks into the environment. One such system is the long range navigation (LORAN-C) system [38], which consists of low frequency radio transmitters. Today the LORAN-C system has been obsoleted by the global positioning system (GPS). The GPS system consists of 24 satellites in Earth's orbit, that continuously transmit timed radio signals. Receivers listen for signals from multiple satellites and measure the time of flight. The time of flight and the satellites trajectory can then be used to estimate the ranges to the satellites. This information is then used to acquire a position estimate for the receiver.

When navigating underwater it is no longer possible to use any of the natural or artificial beacons mentioned earlier. This is because electromagnetic radiation attenuates quickly in water, whether it is visible light, low frequency radio signals, or the high frequency of the GPS. So, underwater a common method for navigation is to use acoustic beacons and is referred to as long baseline acoustic navigation (LBL). Using a LBL is done by deploying two or more acoustic buoys. Then the location of the buoys is surveyed from a surface vessel. A vehicle equipped with an acoustic transducer can then actively query the buoys, and measure the time of flight. Given a sound velocity profile this information can be transformed into a range to the beacon. The ranges can then be used to triangulate a position fix for the vehicle. Some systems are also setup so that the buoys and the vehicle are time synchronized, as

was reported by Eustice et al. [25]. In this case each buoy continuously broadcasts a timed signal. This is similar to the GPS system. The benefit of this system is that it allows multiple vehicles to share the beacons, because each vehicle only needs to listen to the signal and not transmit. There are other variants of this system, running on higher frequencies which can give a better position accuracy but limits the range of the system to 100-200 meters compared to the several miles of the lower frequency systems [106]. Another related method is ultra short base line (USBL) where the transducers are placed very close together in a single sensor, the benefit is that there is no need to deploy a network of buoys. The drawback is that the position accuracy degrades as the distance to the target increases [65].

There has also been some work in using terrain aided navigation underwater [47]. If the operation area has already been surveyed by a surface vehicle to create a high resolution bathymetry map, then this map can be used by the underwater vehicle to aid its navigation and bound the position drift by looking at the correlation of its depth measurements with the prior map.

1.3.3 Navigation with an unknown map

Having no a priori map available is a problem commonly addressed by simultaneous localization and mapping (SLAM). If the vehicle trajectory is known then detected landmarks can be placed in the map at a certain locations. However, the correct trajectory of the vehicle is not known, thus the need for simultaneously estimating the trajectory and the map. The map can be represented in different ways, one is to use landmarks or beacons as in LBL navigation, but this time the map is no longer considered static and the landmark locations become variables in the estimation framework. The buoys can be initialized with the first few measurements. Then, as the vehicle moves, the positions would be updated along with the vehicle trajectory. Thus, giving a more accurate estimate of their location over time [47, 77].

1.4 Simultaneous localization and mapping

There have been many approaches to the SLAM problem, in terms of optimization algorithms, models, data association techniques and sensors. The key concept is that uncertain measurements are collected, and an estimate of the state is computed. There are many texts that give good overview of the many different approaches to SLAM, e.g. Thrun et al. [95], Durrant-Whyte and Bailey [4,21].

Some of the early work to consider uncertainty in measurements can be found in [20,89]. Prior to this it was common to try to eliminate uncertainty by accurate positioning of the devices. When considering mobile robotics, where you do not have full control of the environment, this is usually not feasible. Smith et al. introduced the “stochastic map”, which estimates relative spatial relationship between objects [89]. They use the EKF to estimate the means and second order moments of the spatial relationships between the objects. The state variables are the object poses relative to a global frame. Leonard et al. [62] represented the map as geometric beacons, like corners, lines and cylinders, they also used an EKF to estimate the states.

There are several well known issues associated with the EKF. One is that the complexity grows quadratically with the number of landmarks, because the covariance matrix over all landmarks and the current robot pose needs to be maintained. Another issue is the risk of the filter diverging because of linearization errors. The unscented Kalman filter (UKF) tries to address the issue of linearization by sampling so called sigma points for a better approximation of the underlying distribution [50]. When the function is non-linear, the UKF improves over the EKF, but in many practical applications there is little difference between the two [95].

The extended information filter (EIF) is the dual of the EKF, where the information matrix and information vector are maintained. The information matrix is the inverse of the covariance matrix. In the information form it is easy to incorporate new measurements. However, the benefit is not directly gained with the extended information filter since for each update the mean needs to be computed to be used as a linearization point. The information matrix can be viewed as a graph, where the non-

zero entries correspond to links between nodes. The value in the matrix corresponds to the strength of the link between nodes. Because of the way the information matrix is created in SLAM there are strong links between nodes that are close together and weak links between nodes that are far apart. Because of the robot movement the information matrix will be dense, but most of the entries will be close to zero. The sparse extended information filter (SEIF) uses this fact to provide a constant time algorithm for SLAM [96]. The SEIF algorithm continuously sparsifies the matrix as the robot moves around, allowing for efficient updates of the filter. This sparsification is only an approximation and can result in the filter becoming inconsistent. If the vehicle trajectory is kept as a part of the state then the information matrix will be naturally sparse [23, 26], and this leads us to the exactly sparse delayed-state filter (ESDF). One of the issue with the delayed-state filter is that the size of the state vector grows with time, compared to the size of the map, Walter proposed the ESEIF to address this issue [103]. In ESEIF the vehicle is regularly marginalized out and re-localized, this ensures that the information matrix remains sparse.

One of the problems with previous methods is linearization errors. When measurements were applied, they were applied about the current mean and there was no way to reapply those measurements, so the linearization errors got incorporated into the covariance matrix. This problem can be addressed by considering the full optimization problem, i.e. try to find an optimal solution that considers simultaneously all the measurements. This is commonly referred to as smoothing [36]. In the photogrammetry community this has been referred to as bundle adjustment [98], where the problem is formulated as a large non-linear least squares problem, and using the inherent sparsity for efficient implementation. This approach was introduced as a solution to SLAM by Dellaert et al. as square root smoothing and mapping (SAM) [18, 19]. Square Root SAM factorizes the measurement matrix into the factor R , which is the square root of the information matrix. When the information matrix is sparse, the least squares normal equations can be solved efficiently using the R factor. To support efficient incremental updates it is possible to use Givens rotations at each step to maintain a QR factorization of the system, Kaess et al. incorporated this into incremental SAM

(iSAM) [53]. To support robust data association, access to the covariance is needed. The iSAM algorithm uses dynamic programming to recover parts of the covariance matrix [51], and avoids computing the full covariance matrix when possible.

Particle filters have also been used for SLAM, most common are FastSLAM and FastSLAM 2.0 [73]. The particle filters can track multimodal distributions, and are good at handling non-linear sensor and motion models. One of the key problems with the particle filters is particle depletion, computation resources are limited so it is only possible to maintain a fixed number of particles. What makes it hard to predict how large of an environment we can operate in is that it is difficult to predict how many particles will eventually be needed to estimate the probability distribution. Still, the particle filters have been successfully used. The museum guide Minerva used a particle filter to localize [94] in a static map, an AUV with several sonars used a particle filter to navigate into underwater holes [27], estimating both a map of the hole and the vehicle position..

1.5 Overview

The remainder of the thesis is organized into the following chapters:

Chapter 2 gives an overview of the imaging sonar and our state formulation.

Chapter 3 describes a method for using the imaging sonar to improve the navigation accuracy. A dense feature extraction algorithm is introduced and a scan matching method, the normal distribution transform (NDT), is used to calculate constraints between vehicle poses at different times. These measurements are used with dead reckoning to maintain a maximum likelihood estimate of the vehicle trajectory.

Chapter 4 develops an architecture for repeated operations. The data and execution model of the system are described. Then a prototype implementation of the system is used to demonstrate some of the key concepts. The examples use data from a land robot, equipped with a laser scanner.

Chapter 5 describes a method for re-localization. A feature descriptor is described that can be used with 2D point clouds. Then different data structures to index k -dimensional place descriptors are compared. These data structures can be used to efficiently search in large amount of data.

Chapter 6 presents conclusions from this work and explores directions for future research.

Chapter 2

Acoustic imaging

Sonar sensors are commonly used for underwater imaging, because visible light attenuates rapidly in water. This limits the range of typical vision sensors like the camera. Turbid waters are also a challenge for underwater cameras, while having little or no effect on sonars. In this chapter we will look at few different types of sonars and the principles behind their operation before focusing in detail on the imaging sonar that is used on the HAUV vehicle.

2.1 Sonar overview

One of the earliest sonars used to collect underwater imagery is the so called side-scan sonar. The first side-scan sonar images were taken in 1963 [30] by Dr. Harold Edgerton. He used a sub bottom profiler (SBP) looking sideways instead of straight down. This enabled him to take images of a wider swath. Since then there have been many advances in the equipment used for side-scan sonar imaging.

The side-scan sonar is composed of an acoustic transducer connected to a receiver and a transmitter. The acoustic transducer sends out an acoustic signal at regular intervals. When the acoustic signal hits an obstacle there will be a reflection or a backscatter of the signal that can be measured by the receiver [69]. The side-scan sonar can be deployed on a tow-fish, a remotely operated vehicle (ROV) or an autonomous underwater vehicle (AUV), typically transducers are fitted on both sides

to get images on both port and starboard. Figure 2-1 shows a side-scan sonar record, the vehicle is located in the center where the bright yellow line shows up. Time is spread vertically across the image while the horizontal distance is the slant range. Each horizontal line corresponds to a single ping of the side-scan sonar.

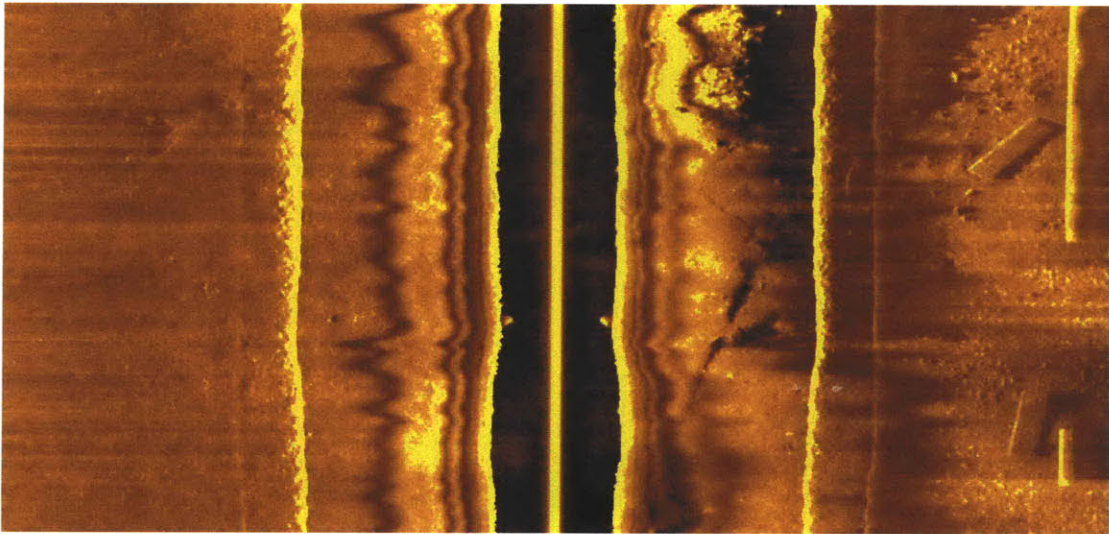


Figure 2-1: An image from a 900Khz Marine Sonic side-scan sonar (courtesy of Gavia). The vehicle is located in the center of the image and is traveling vertically. This is a harbor bottom. On the right, parts of the dock are visible and also some debris lying on the bottom.

Multibeam sonars are another type of sonars. One of the limitations of the side-scan sonar is that it only receives the range to the targets being imaged, as compared to the multibeam sonar that can actually give us range and bearing to the target. Thus, allowing calculation of an accurate position of the returned signal in 3D space. Using these multibeam sonars it is possible to create high quality bathymetric maps [56].

The sonar used for the work in this thesis is the DIDSON imaging sonar [5]. It is a multibeam sonar that uses an acoustic lens. It has an array of 96 beams, a single image is formed from 96 pings, compared to the single ping for the side-scan. As with the side-scan sonar there is uncertainty in the elevation of the returned signal. In the DIDSON this is 14 degrees in the standard setup. Figure 2-2 shows how the image is formed, the sonar is located close to the target at a shallow angle. If there

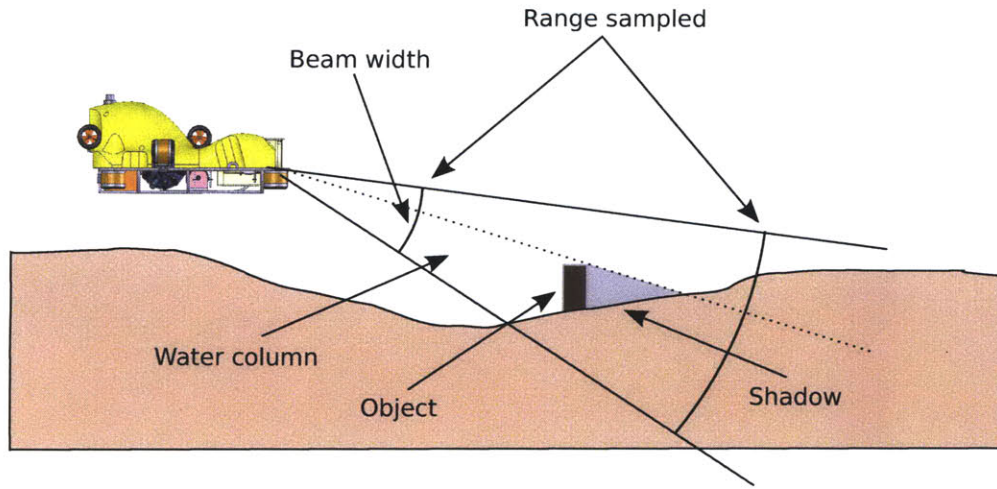


Figure 2-2: How the sonar image is formed: the sonar transmits a sound wave and then listens for returns over a given range. The range returned is the slant range to the reflecting surface. The transducer is restricted to a certain vertical and horizontal beam width. Objects that protrude above the surface will often give a strong returns, followed by a shadow. The bottom usually gives a much weaker return.

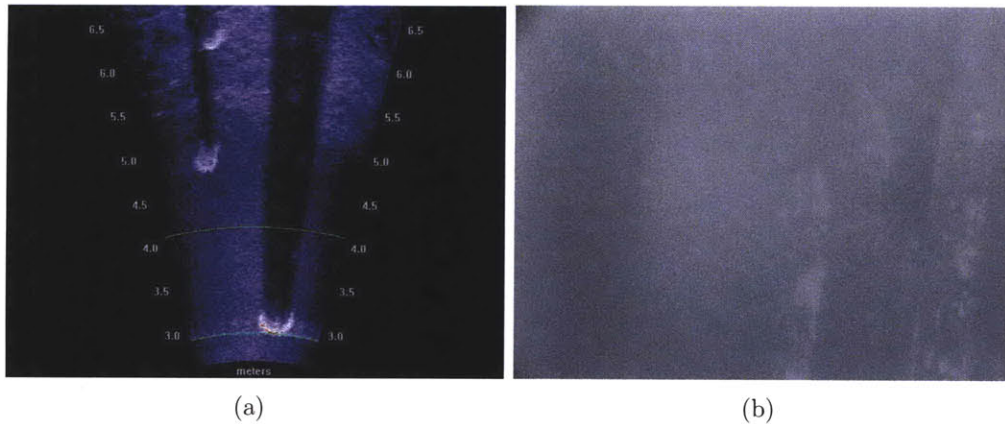


Figure 2-3: Images from the MIT Sailing Pavilion showing pier pilings. (a) A DIDSON image using a 28 degree spreader lens, the bright returns followed by a long shadow are pier pilings, the pilings are between 3 and 6 meters away from the sonar. (b) A camera image of a pier piling taken close to the surface about half a meter from the piling. The water is very murky which results in low visibility .

are any objects in the sonar's field of view they will return a strong return, and cause a shadow on the image. This shadow can be clearly seen in Figure 2-3, the bright objects are pier pilings at the MIT Sailing Pavilion in the Charles River. Because the pier pilings fill out the vertical field of view of the sonar, the shadow reaches to the end of the image. On the right there is a camera image of a pier piling, taken at the surface and only around 50cm away. This clearly demonstrates why, in some situations, acoustic imaging is the only feasible option for underwater imaging.

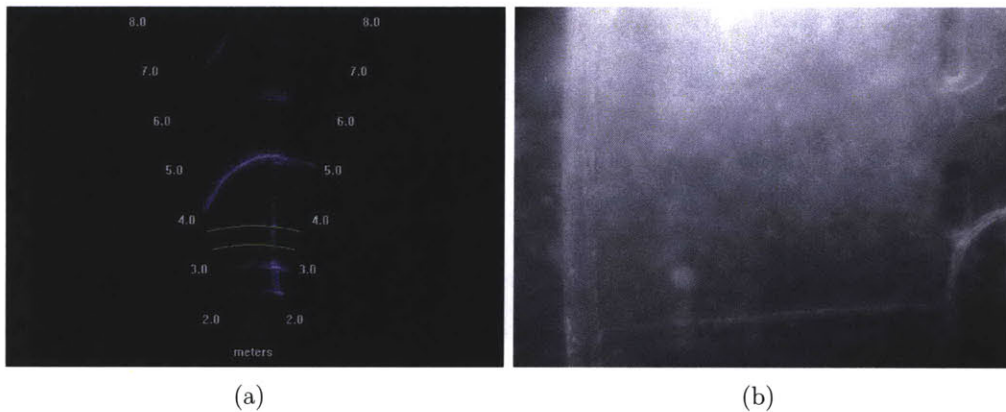


Figure 2-4: Images of the propeller ring on the *R/V Oceanus* vessel. (a) A DIDSON profiling image of the propeller ring using a 1 degree concentrator lens. (b) A camera image of the side of the propeller ring, the visibility was around 1 meters. This shows how much detail camera images have when there is decent visibility in the water.

The DIDSON uses acoustic lenses to manipulate the incoming beams, and it is possible to add a lens on the sensor's front to either spread or concentrate the effective beam width of the sonar. With a 1 degree concentrator lens the DIDSON can be used as a micro-bathymetry sonar. Figure 2-4(a) shows an image from the DIDSON when using the profiling lens: now the ship's hull return is just a narrow line. It is relatively simple to extract a 3D profile from this image by simply choosing the point with the peak return and use that as a range. The reason it is not possible to do this reconstruction without the concentrator lens is because the return can be anywhere along an arc that subtends 14 or 28 degrees. So the accuracy would not be sufficient for 3D reconstruction. Figure 2-4(b) shows a camera image of the same feature that is shown in the profile image. In this image the water clarity is much

better and more detail is available using the camera. In many situations use of both the sonar and camera can complement one another.

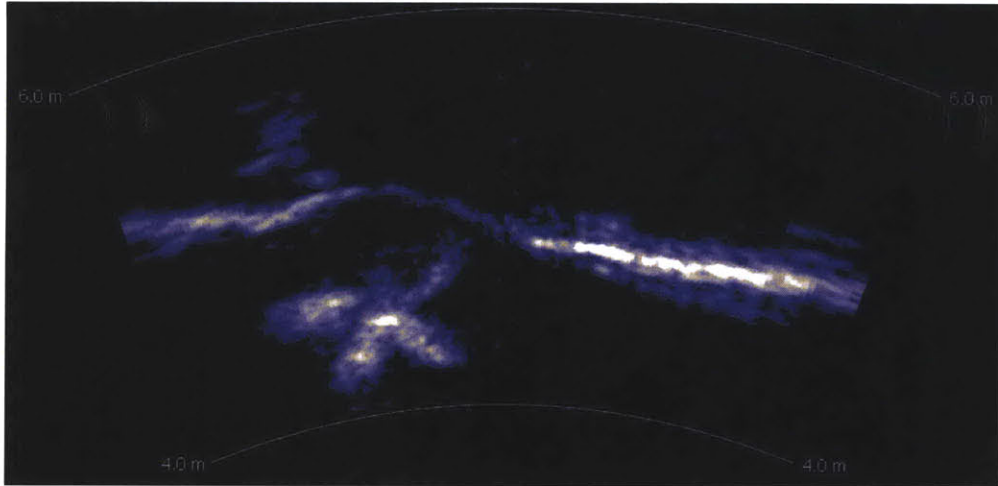


Figure 2-5: *King Triton* profile image showing a part of the hull and the propeller. This image is from the BlueView MB2250 micro-bathymetry sonar. A range profile can be created from this image by extracting points with the highest intensity along each beam.

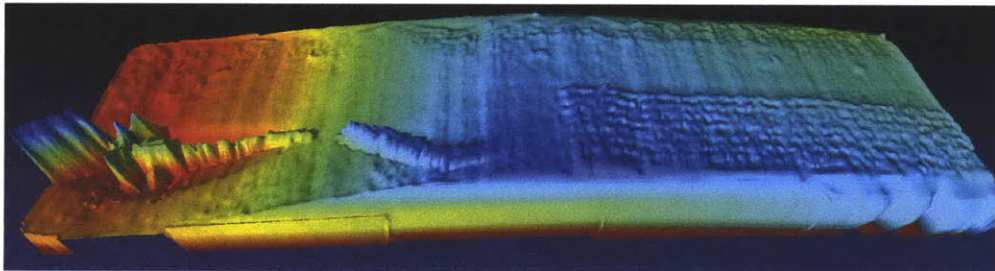


Figure 2-6: *King Triton* ship hull 3D reconstruction. It was created using sonar scans from the BlueView MB2250 mounted on the HAUV. Range points were extracted from the sonar image by choosing, for each beam, the point with the highest intensity. The points were projected into Cartesian coordinate frame using the vehicle's dead-reckoning. The mesh was then created by placing the points into a grid and using average height in each cell.

Another type of sonar that falls in the same category as the DIDSON is the BlueView MB2250. It is a micro-bathymetry sonar with 256 beams and a 1 degree beam width in the elevation. It does not use the same technology as the DIDSON but provides similar data. One of the benefits of the MB2250 is that it has a greater field of view than the DIDSON, 45 degrees compared to 28 degrees. Figure 2-5 shows a

single frame from the MB2250 taken of the vessel *King Triton*, it shows the propeller and the ship hull. Using the HAUV navigation sensors it is possible to project all the range measurements into a global frame and perform a 3D surface reconstruction. Figure 2-6 shows 3D reconstruction of the *King Triton* using the BlueView MB2250.

2.2 Vehicle state

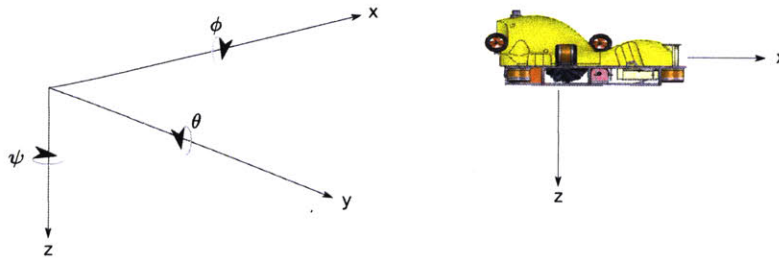


Figure 2-7: The Coordinate system used has the x -axis pointing forward (surge), y -axis to the right (sway) and z -axis is down (heave). The rotations used are yaw (ψ), pitch (θ) and roll (ϕ).

The vehicle state being estimated is the vehicle pose, which consists of position and attitude. The vehicle position in 3D is specified by Cartesian coordinates x, y, z with respect to some arbitrary reference frame, such as the starting point of the mission or a GPS frame acquired before diving. The attitude of the vehicle is specified by the standard Euler angles ϕ, θ, ψ that refer to roll, pitch, and heading respectively. This follows the convention given in Fossen [33] and is illustrated in Figure 2-7.

While the vehicle state has six degrees of freedom, it is only necessary to estimate three of them. Absolute depth measurement is obtained by measuring water pressure and absolute measurements of pitch (θ) and roll (ϕ) are based on gravity. On the HAUV the accuracy for pitch and roll is $\pm 0.1^\circ$ [103]. The estimate for the remaining three degrees of freedom will drift over time when gyro and DVL measurements are available.

The RLG is used for estimating heading because using a magnetic compass in close vicinity to a ship hull or other steel structure is often not a viable option. The vehicle estimates its heading by integrating the angular rotation as measured by the

gyro. The x, y position is then estimated by dead reckoning using the velocities from the DVL and the heading estimate. The goal of our work is to bound drift in the position estimate by using the imaging sonar.

2.3 Imaging sonar geometry

The DIDSON samples each beam 512 times when recording an image, so each image is 96×512 8bit intensity values. The intensity values are a function of the beam pattern, transducer gains, orientation of the surface relative to the sonar, and the surface material [48]. From this data consistent features are extracted and used for registration when going over previously visited areas.

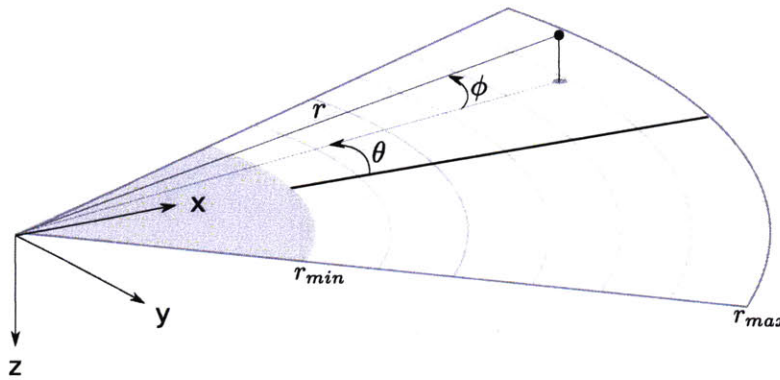


Figure 2-8: Imaging sonar geometry. The sonar images surfaces in a volume spanned by its field of view in the azimuth and elevation. The coordinate system is set up such that x -axis points outwards from the sonar, y -axis to the right and z -axis down. The returned acoustic energy is recorded over a fixed time interval r_{min} and r_{max} . Considering the return from a single point, then θ is azimuth angle to the point and ϕ is the elevation. The elevation is not recorded in the image, so all points along an arc at a fixed range end up contributing the same value in the image.

Using the imaging sonar to correct for the drift accumulated by dead reckoning requires an understanding how the sensor functions. Following the formulation by Negahdaripour et al. [75, 76, 87], the geometry of the imaging sonar is defined and a model, that describes how the image is formed, is derived. To generate an image, the sonar emits a sound wave and then listens with an array of receivers to the returning sound wave, sampling the acoustic energy returned from different directions. The

sonar samples the transducers at fixed intervals, providing time of flight, azimuth angle and intensity for each sample. Combining the returns from all the elements provides an image of the reflective surfaces in front of the sonar. For the imaging sonar under consideration, the vertical beam width is greater than the horizontal beam width. A given point in the image can lie anywhere on an arc at a fixed range, spanning the vertical beam width.

Mathematically the imaging process can be described as follows. The coordinate system for the sonar is as shown in Figure 2-8. For simplicity let us assume that the sonar coordinate system coincides with the vehicle coordinate system. The x -axis is perpendicular to the sonar array, the y -axis is to the right and z -axis points down. Let us consider a point $\mathbf{p} = [x \ y \ z]^T$ in the sensor coordinate frame, where x, y, z are the Cartesian coordinates of the point. Now let $\mathbf{s} = [r \ \theta \ \phi]^T$ be the same point in spherical coordinates, where r is the range, θ is the azimuth and ϕ is the elevation of the point. The spherical and Cartesian coordinates can be related with the following equations

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r \cos \phi \cos \theta \\ r \cos \phi \sin \theta \\ r \sin \phi \end{bmatrix} \quad (2.1)$$

$$\mathbf{s} = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \tan^{-1} \left(\frac{y}{x} \right) \\ \tan^{-1} \left(\frac{z}{\sqrt{x^2 + y^2}} \right) \end{bmatrix} \quad (2.2)$$

In practice the measured image of point \mathbf{p} is $I(\mathbf{p}) = [r \ \theta]^T$. The Cartesian projection of the point \mathbf{p} is then

$$\hat{I}(\mathbf{p}) = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \quad (2.3)$$

This projection can be viewed as an approximation to an orthographic projection.

What is the effect of observing the same structure from two different poses? Let

us first consider the effects of the approximation on translation. Starting with point $p = [x \ y \ z]^T$, which yields the spherical coordinates $s = [r \ \theta \ \phi]^T$. Now the sonar moves by $[\delta x \ \delta y \ 0]^T$. The point in the new coordinate frame is

$$\mathbf{p}' = \begin{bmatrix} x - \delta x \\ y - \delta y \\ z \end{bmatrix} \quad (2.4)$$

Using the projection we obtain

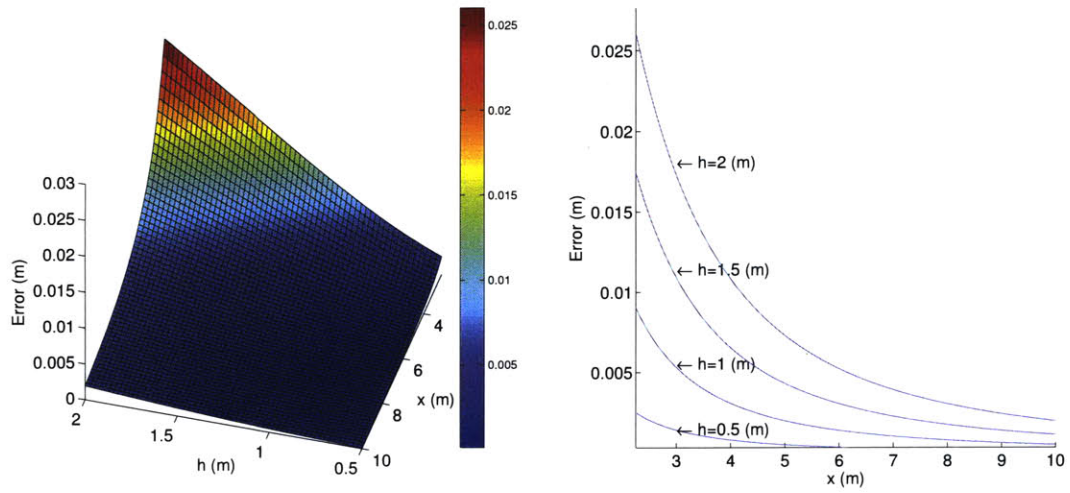
$$\hat{I}(\mathbf{p}) = \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \quad \hat{I}(\mathbf{p}') = \begin{bmatrix} r' \cos \theta' \\ r' \sin \theta' \end{bmatrix} \quad (2.5)$$

From the projection the estimated translation is $\mathbf{t} = \hat{I}(\mathbf{p}') - \hat{I}(\mathbf{p})$. Then the translation error becomes $\epsilon = \mathbf{t} - [\delta x \ \delta y]^T$

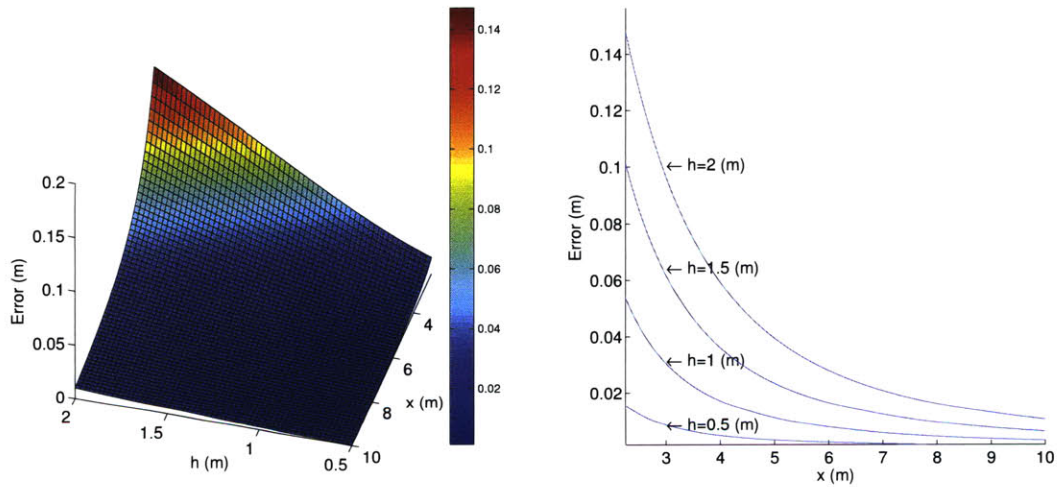
$$\epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \end{bmatrix} = \begin{bmatrix} u(1 - \cos \phi) - u'(1 - \cos \phi') \\ v(1 - \cos \phi) - v'(1 - \cos \phi') \end{bmatrix} \quad (2.6)$$

Assuming, a flat surface that is level with the world's x - y plane, at a distance h from the vehicle, then from equation (2.2) we can see that the elevation depends on the distance in the x - y plane and the vertical distance, z , to the point. Let us consider a situation where the targets are at 2 meter distance below us, then for 0.5 meter translation the error is 0.14 meters when the target is close (around 2.25 meters), and decreases as the targets gets farther away (less than 3cm at 7m distance). In Figures 2-9(a) and 2-9(b) we evaluate the error for 0.1m and 0.5m step size respectively and varying distances to the object.

Next we analyze how the projection affects rotation around the vertical axes of the sonar. Let \mathbf{p} and \mathbf{p}' be the point before and after rotation respectively. Rotating



(a) Translation error for 0.1m steps



(b) Translation error for 0.5m steps

Figure 2-9: Translation error in the x direction. On the right is a surface of the translation error when an object at different x and h positions is used for calculating the translation. Where x is the distance in meters along the x axis of the sonar and h is the distance in meters from the z axis, or the height we are from the object. On the right are error plots for certain values of h .

\mathbf{p} counter-clockwise by angle α around the z -axis

$$\mathbf{p}' = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p} = \begin{bmatrix} r \cos(\theta + \alpha) \cos \phi \\ r \sin(\theta + \alpha) \cos \phi \\ r \sin \phi \end{bmatrix} \quad (2.7)$$

and

$$\mathbf{s}' = \begin{bmatrix} r \\ \theta + \alpha \\ \phi \end{bmatrix} \quad (2.8)$$

Assuming only rotation of the sonar, let $\hat{\alpha}$ be the estimate of the rotation of the sonar. The estimate is obtained by calculating the angle between the images of the two points.

$$\|\mathbf{p}\| \|\mathbf{p}'\| \cos(\hat{\alpha}) = \hat{I}(\mathbf{p}) \cdot \hat{I}(\mathbf{p}') \quad (2.9)$$

$$= \begin{bmatrix} r \cos \theta \\ r \sin \theta \end{bmatrix} \cdot \begin{bmatrix} r \cos(\theta + \alpha) \\ r \sin(\theta + \alpha) \end{bmatrix} \quad (2.10)$$

$$= r^2 \cos \alpha \quad (2.11)$$

$$\Rightarrow \hat{\alpha} = \alpha \quad (2.12)$$

In conclusion, the projection preserves the change in azimuth angles. This holds for rotation around the sonar's z -axis, but normally the sonar is tilted relative to the vehicle's coordinate frame and it is the heading change in that frame that is to be measured.

Rotating the sonar around its y -axis (pitch) only changes ϕ and has no effect on the projection of the points. The rotation only affects the intensity of the returned values, and eventually the points will go out of the sonar's field of view. Roll of the sonar will tend to cause compression of the points along the y -axis. In our case the error is minimal as the roll angle is typically small. However, if the imaged surface deviates

significantly from our horizontal surface assumption, then the deviation would have to be modeled. One crude approximation is to assume a plane that is aligned with the sonar and positioned at the sonar origin. A better approximation is to use range measurements from the DVL to estimate the plane. This is feasible when the DVL is used to track the surface being imaged, i.e. when carrying out ship hull oriented tracking or bottom tracking for the bottom surveys.

2.4 Summary

In this chapter we have looked at several different types of sonars: side-scan sonar, imaging sonar and profiling sonar. Each of those sonars serves a different purpose for the proposed application. Further detail has been presented for the imaging sonar and the implication the image formation has on the accuracy when registering pairs of frames. In the next chapter this information will be used to build a sonar-aided navigation system, that can improve upon simple dead-reckoning navigation, and provide drift free navigation over an extended period of time.

Chapter 3

Sonar-aided navigation

There has been much interest in augmenting the navigation capabilities of underwater vehicles by using environmental measurements. This is primarily due to the fact that other methods for bounding the navigation drift require us to deploy acoustic beacons. Hence, it is of interest to consider different methods that allow us to avoid the cost and effort of deploying a separate positioning system. This chapter describes a method that registers pairs of acoustic images and illustrates how it can be used to estimate the vehicle trajectory and to bound the navigation accuracy. Before continuing, we first provide a brief overview of prior work on utilizing environmental measurements for underwater vehicle localization.

Many different methods have been developed for underwater localization, using various sensors and estimation methods. A forward looking multibeam sonar was used by Carpenter [14]. He identified objects from the sonar image by clustering strong backscatters. Furthermore, he extracted features describing the objects to aid with data association. The map and vehicle position was then estimated using an EKF filter. Others have worked with forward looking sonars, Walter et al. [102] used manually extracted landmarks, and later automatic feature detection [103] with the Exactly Sparse Extended Information Filter (ESEIF) to produce a map of the environment. An automatic feature detector and a landmark formulation using an EKF filter was used by Englot et al. [22]. In work by Folkesson et al. [32], a forward-looking sonar was used with a prior map to track beacons in sonar images and use

them to localize the vehicle. This map would then get augmented as new features appeared. They also introduced an innovative way of tracking features to improve initialization, then a global optimization was performed on multiple submaps.

A full 360-degree sonar scanner has been used in partially structured underwater environments [84, 85] for localization, by tracking line features in the environment using an EKF for the estimation process. Mallios et al. recently showed promising results in [68] using a mechanical scanning sonar and scan matching in an EKF framework. A particle filter with a 3D evidence grid was used by Fairfield et al. [27] for the Deep Phreatic Thermal Explorer (DEPTHX), which had 52 pencil beam sonars. This system was later used with a multibeam sonar [28], to localize from a known bathymetric map. Using side-scan sonar images for navigation was reported by Ruiz et al. [93]. Newman et al. developed an efficient estimation method using several maps [78], this method was used with data from a synthetic aperture sonar (SAS).

Vision has also been used for underwater navigation. Visual navigation using mosaics was developed by Gracias [37]. Eustice et al. introduced the Visually Augmented Navigation (VAN) framework, which used registration of camera frames to improve navigation accuracy. The system was used to visually navigate the Titanic [24]. This has been followed up in more recent work by Kim et al. [54]. The use of Cholesky factorization, for efficient state recovery, was introduced to the VAN framework by Mahon et al. [67]. Negahdaripour et al. have combined an optical camera and an acoustic camera [76], forming opti-acoustic stereo, which is used for 3-D target reconstruction.

In this thesis we are interested in using the scan matching concept that has been so successful with laser range finders [7, 8, 64]. One of the reasons to consider dense feature extraction is that even though it is easy to pick training targets as features for navigation, we would also like to be able to localize using the more natural features in the environment, like: the sea chests, welding lines, cooling pipes on a ship hull, tiers, planks and other debris on the harbor floor. Using complete scans each time aids with the data association, because, by registering two frames, multiple objects

are naturally considered each time. To estimate the map, an incremental nonlinear least square solver is used, and it addresses the inconsistency [49] issues that can occur using an EKF.

3.1 Feature extraction

Unlike a laser scanner, which gives the range to the closest obstacle, the imaging sonar returns multiple intensity values along the beam. An example sonar image is shown in Figure 3-1(a). A strong return usually represents an object standing above the sea floor or the surface being imaged, and it is usually followed by a shadow. Also a hole or a depression on the surface can show up as a shadow but then there is no associated bright return. Variations in the returned signal are also caused by difference in material properties, strength of the transmitted signal, receiver sensitivity, distance to target, and the grazing angle, among other things.

We propose to extract reasonably stable features based on sharp transitions. We identify sharp transitions in the measurement, and check for a possible return from an object followed by low signal caused by the object shadowing the background. The main steps of the algorithm are:

1. Smooth image
2. Calculate gradient
3. Threshold a top fraction as features
4. Cluster points and throw out small clusters

First the image is smoothed using a median filter, significantly reducing noise, while still preserving edges, as shown in Figure 3-1(b). Next, the gradient is calculated by computing the difference between the current value and the mean of the last few values (Figure 3-1(c)). The number of previous values used to calculate the mean around the current values affect the type of objects that are detected. Then points with gradient exceeding a given threshold are marked as features (Figure 3-1(d)). The

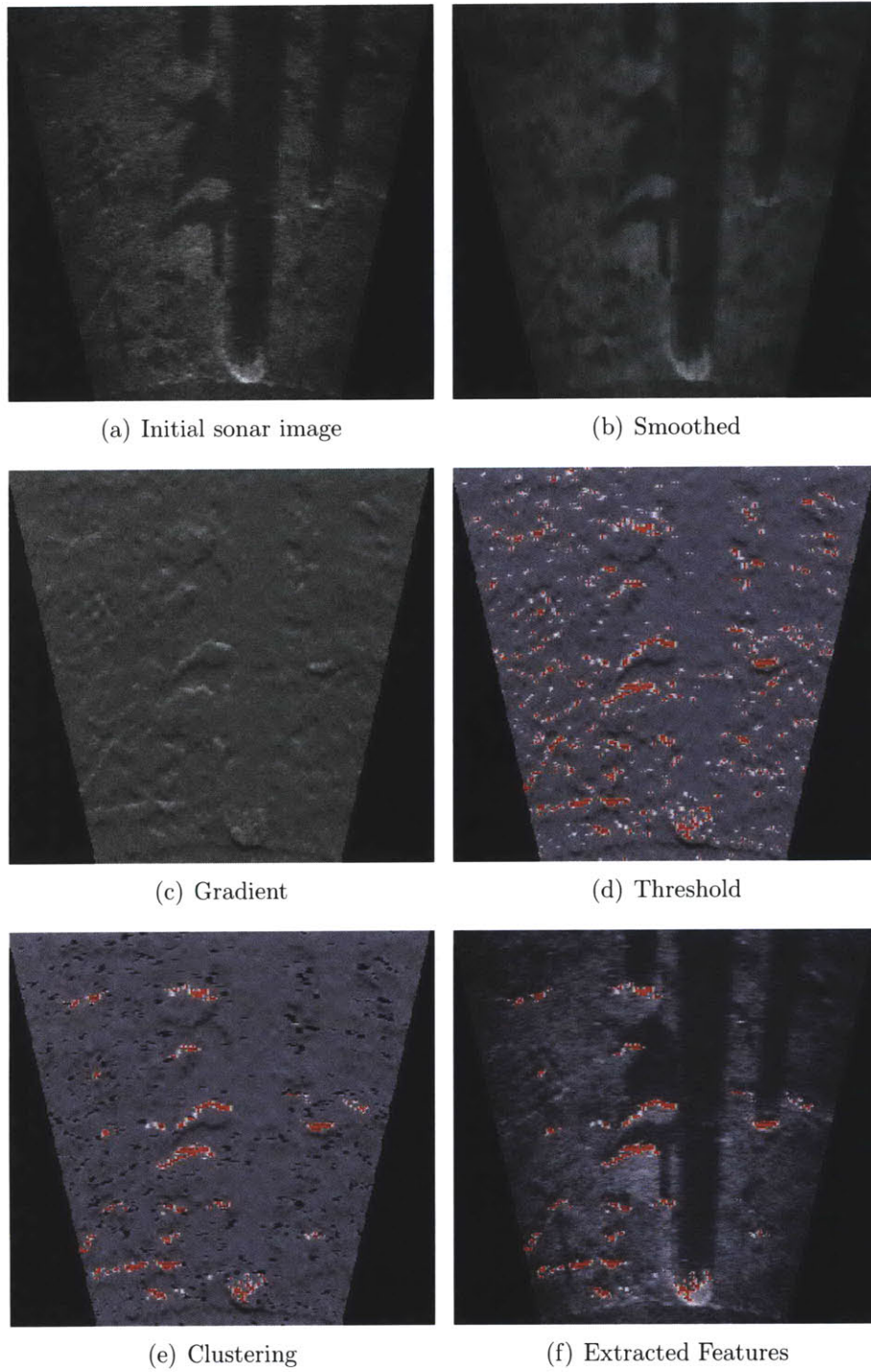


Figure 3-1: Intermediate steps of the feature extraction process. The extracted features are shown in red.

threshold is adaptively selected, such that a fixed fraction of the features is retained. Next, spurious features are eliminated by clustering the points and eliminating small clusters (Figure 3-1(e)). The extracted features are shown in Figure 3-1(f), which typically consist of 1000 to 2000 points.

How do the extracted features translate into 3D Cartesian coordinates? Because the elevation of a measured point is unknown, there is an ambiguity in the projection into Cartesian coordinates. The points are assumed to lie in a plane that is level with the vehicle. This approximation gives a reasonable result when the surface being imaged is roughly level. For the best imaging results it is good to stay around 1-2 meters away from the surface, so the objects being imaged are around 0.5-2 meters away from the projection plane. Registration is never performed on frames that are far away from each other. As a consequence, the error in the registration caused by the approximation can be up to 10-15cm, while typically being below 5cm. It is possible to improve on this approximation by using the range measurements from the DVL to estimate the ground plane.

3.2 Registration

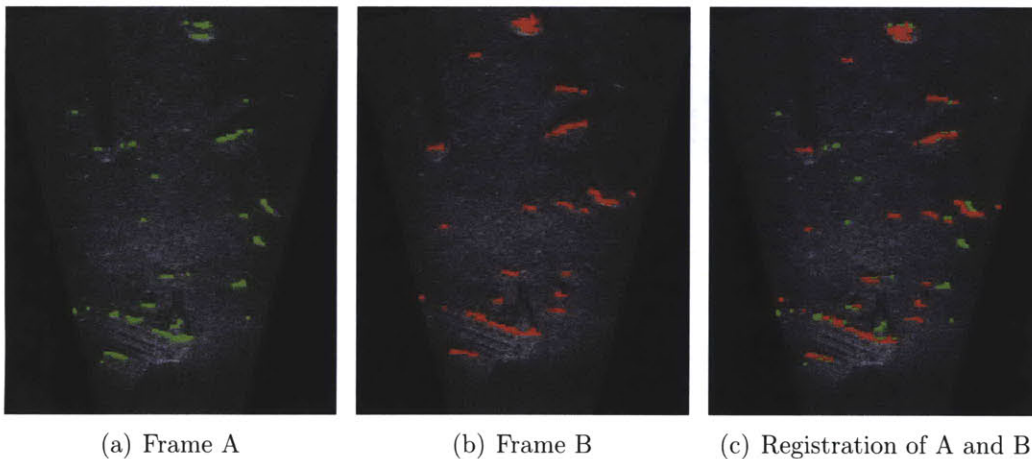


Figure 3-2: Two scans before and after registration. Red points show the model scan, green features the current scan after registration.

Figure 3-2 shows the alignment of two sonar images using the Normal Distribution

Transform (NDT) algorithm [7]. The NDT algorithm works by assigning points to cells of a grid spanning the area. For each cell the mean and variance of the points that end up in the cell are calculated. This is done for four overlapping grids, where each grid is shifted by half a cell width along each axis. Using multiple shifted grids alleviates the effect of discontinuities resulting from the discretization of space. Two of the benefits using the NDT are that it gives a compact representation of the scan and exact correspondences between points is not needed. This is useful because the movement of the vehicle causes variation in the insonification of the surfaces, which causes some points to drop in and out of the extracted feature set.

The NDT serves as the model for registration. From the current scan, a score is calculated for each point by evaluating Gaussians parameterized using the values in the corresponding cells that contain the points. This gives a measure of how likely it is that the point is measured given the data. A cost function is defined as the sum of the negative score of all the points in the current view. Minimizing the cost function with respect to the parameters (x, y, ψ) gives the transformation between the two scans.

Because the main goal of the registration method is to localize the vehicle, an initial estimate of the vehicle location when searching for the match is not needed. Instead, the optimization is repeated with several initial values to try to find the global minimum. To avoid accepting a wrong match, the match has to have a normalized score over a given threshold, and the current scan has to include a minimum number of points.

3.3 Map estimate

The geometric measurements resulting from sonar registrations are used to correct the vehicle drift over time. For that purpose, a pose graph formulation of the simultaneous localization and mapping (SLAM) problem is adopted, to obtain a least-squares estimate based on all measurements.

To achieve a drift-free estimate of the vehicle position, an estimate of the vehicle's

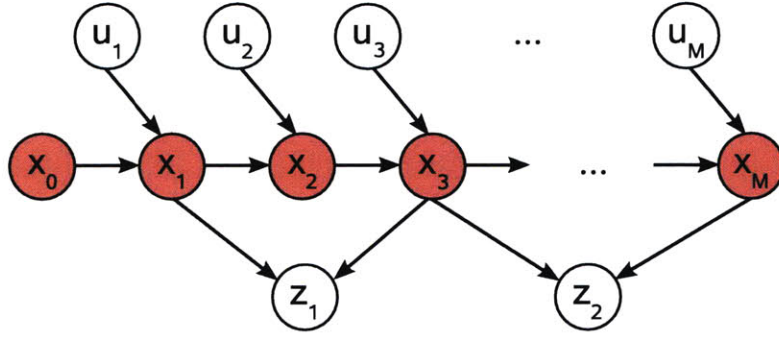


Figure 3-3: Bayes net formulation of the estimation problem. x_i is the vehicle state at time i , u_i the control input and z_k a registration between arbitrary poses along the trajectory.

complete trajectory is maintained. As the vehicle moves around the environment it collects sensor information along the trajectory. This information is then used to detect when the vehicle reaches a place that it has seen before. This is commonly referred to as loop closure in the SLAM literature.

Following the formulation in [53], we model the problem as the joint probability distribution $P(X, Z, U)$ where $X = [x_1 \ x_2 \ \dots \ x_N]$ is the vehicle trajectory, $Z = [z_1 \ z_2 \ \dots \ z_M]$ are measurements between two poses and $U = [u_1 \ u_2 \ \dots \ u_N]$ are the controls between two consecutive poses. The Bayes net corresponding to this model is shown in Fig. 3-3. From the model we can directly write the joint probability distribution as

$$P(X, Z, U) = P(x_0) \prod_{i=1}^N P(x_i | x_{i-1}, u_i) \prod_{k=1}^M P(z_k | x_{a_k}, x_{b_k}) \quad (3.1)$$

We assume Gaussian process and measurement models. The motion model is defined as a function f that takes the previous state and the control input to predict our current state

$$x_i = f(x_{i-1}, u_i) + w_i \quad w_i \sim N(0, \Sigma_i) \quad (3.2)$$

and the measurement model is defined by a function h that relates two poses

$$z_k = h(x_{a_k}, x_{b_k}) + v_k \quad v_k \sim N(0, \Lambda_k) \quad (3.3)$$

Given the measurements Z and U we obtain an estimate of our latent variables X . One such estimate is the *maximum a posteriori* (MAP) estimator. Let \hat{X} be the MAP estimator, then

$$\hat{X} = \arg \max_X P(Z, U|X)P(X) \quad (3.4)$$

$$= \arg \max_X P(Z, U, X) \quad (3.5)$$

$$= \arg \min_X -\log P(Z, U, X) \quad (3.6)$$

Under the assumption that the measurement noise is Gaussian, we plug equation (3.1) into (3.6) and arrive at the following *nonlinear least-squares* problem

$$\hat{X} = \arg \min_X \sum_{i=1}^N \|f(x_{i-1}, u_i) - x_i\|_{\Sigma_i}^2 \quad (3.7)$$

$$+ \sum_{i=k}^M \|h(x_{a_k}, x_{b_k}) - z_k\|_{\Lambda_k}^2 \quad (3.8)$$

where $\|x\|_{\Sigma}^2 := x^T \Sigma x$.

By exploiting the sparseness of the problem it is possible to obtain an efficient solution to this problem with Gauss-Newton methods. Incremental smoothing and mapping (iSAM) [53] provides an efficient on-line solution to the problem that updates an existing solution rather than recalculating from scratch in each step.

3.4 Implementation

The navigation system we built operates as a payload module for the HAUV. The system receives sensor measurements from the HAUV payload driver. The two main components in the payload are the map module and the pilot module. An overview

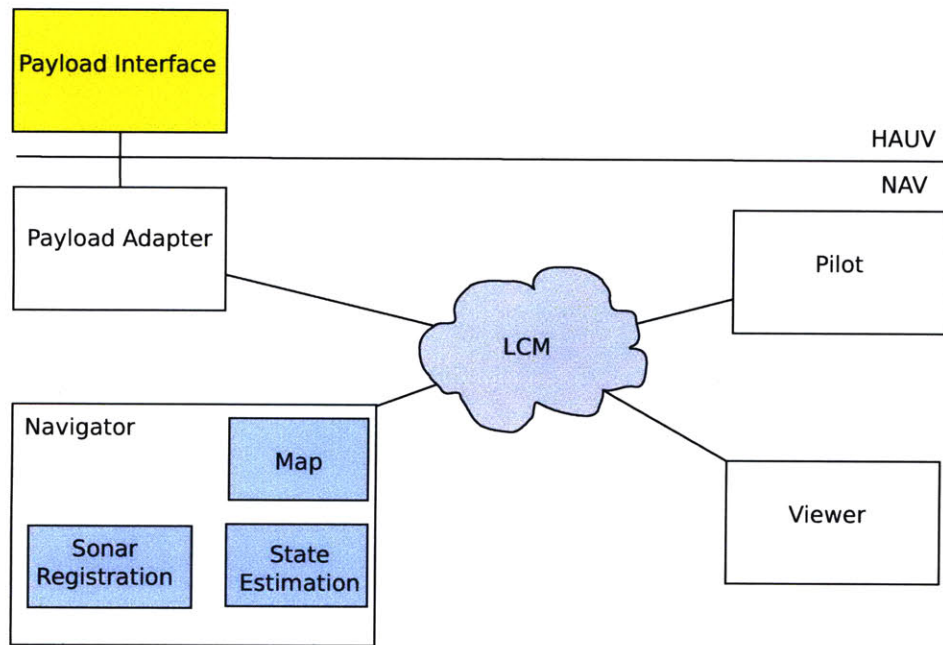


Figure 3-4: Overview of the sonar-aided navigation system.

of the system is shown in Figure 3-4.

The map module integrates the DIDSON imagery and the vehicle position data from the HAUV to maintain an estimate of the vehicle trajectory and a map of the environment. New DIDSON frames are being sent while building the map. For each frame a dense feature set is extracted. Frames with a low number of detected features are ignored. A lookup in the map is performed to see if there are any other frames close to the vehicle's current position. If there are frames nearby the system tries to register them to one another. Successful registrations are then used for the trajectory estimation. If there are no frames nearby then the current frame is added to the map. As mentioned in section 3.2 only registrations with sufficiently high scores are accepted.

The pilot module tracks the coordinate transformation between the latest state estimate from the map system and the vehicle state. The vehicle estimates its position by using the DVL and gyro for dead reckoning. The pilot is sent mission plans as a set of waypoints that it then sequentially executes. First the pilot transforms the waypoint from the map coordinate frame into the vehicle coordinates, and then it

issues the transformed waypoint to the HAUV vehicle using the payload driver.

3.5 Results

Table 3.1: Overview of the results from our experiments.

Data set	Length (m)	Time (min)	Sonar			DVL		
			Err μ (m)	Err σ	Err max	Err μ	Err σ	Err max
Tank Box	173	43	0.2	0.17	0.82	2.4	1.48	4.3
Charles River	970	83	0.7	0.57	2.3	1.15	0.82	3.63
Charles River (post processed)	1459	133	0.31	0.23	1.2	1.6	1.18	4.88
<i>King Triton</i>	324	44	0.14	0.07	0.3	0.27	0.18	0.66

We performed several experiments in different settings. Our initial trials were tank experiments to verify that we could run the system online and stay localized over an extended period of time. Then we ran the vehicle in a more realistic setting: An inspection of the river bottom of the Charles River at the MIT Sailing Pavilion. Another experiment was performed on the *King Triton* vessel in Boston Harbor. The main difference between the harbor surveillance and ship hull inspection missions is the type of features that the vehicle encounters. The results allow us to show that the system works in a wide range of situations.

To evaluate the results we manually labeled several obvious objects in multiple frames over the duration of the mission. The purpose of this was for us to systematically track localization errors of those objects using different localization methods. It is important to note that these labels were never used in the navigation solution; all features used for navigation were automatically extracted during the missions. This metric tells us if a method gives a consistent estimate of an object position over time. It is not useful to detect systematic errors like scale factor error in velocity measurements with this metric, but it is still useful for comparing different settings and algorithms. In Table 3.1 the main results from the experiments are summarized. Note that when the navigation accuracy is drifting the mean error might not be the best measure of accuracy, so maximum error encountered is also included. The heading is also drifting, so the actual position error varies depending on the vehicle location. For example, given a rotation around the origin, the position error increases as the vehicle moves farther away from the origin.



Figure 3-5: The HAUV operating in the tank.

The experiments demonstrate that the sonar registration can make use of a wide range of features for navigation. The features used for registration ranged from cooling pipes and zinc anodes on the ship, to pier pilings, planks and depressions in the Charles River. It is important not to focus on a specific type of feature, as the kind of features encountered in new underwater harbor environments are difficult to predict.

3.5.1 Tank

The first experiments that we report were performed in a testing tank at MIT. The tank is approximately 9 meters long, 3 meters wide, and 1 meter deep. The vehicle ran for approximately one hour in the tank, keeping stationary while observing several cinder blocks as targets. Then we had the vehicle move in a box pattern so that it occasionally lost sight of the features but was able to reacquire them and re-localize. The vehicle had multiple features in sight for most of the frames.

One interesting aspect of this experiment is that the tank is a poor environment for the DVL, both because the water is shallow and the vehicle is surrounded by walls. The poor performance of the DVL meant that the drift using only dead-reckoning

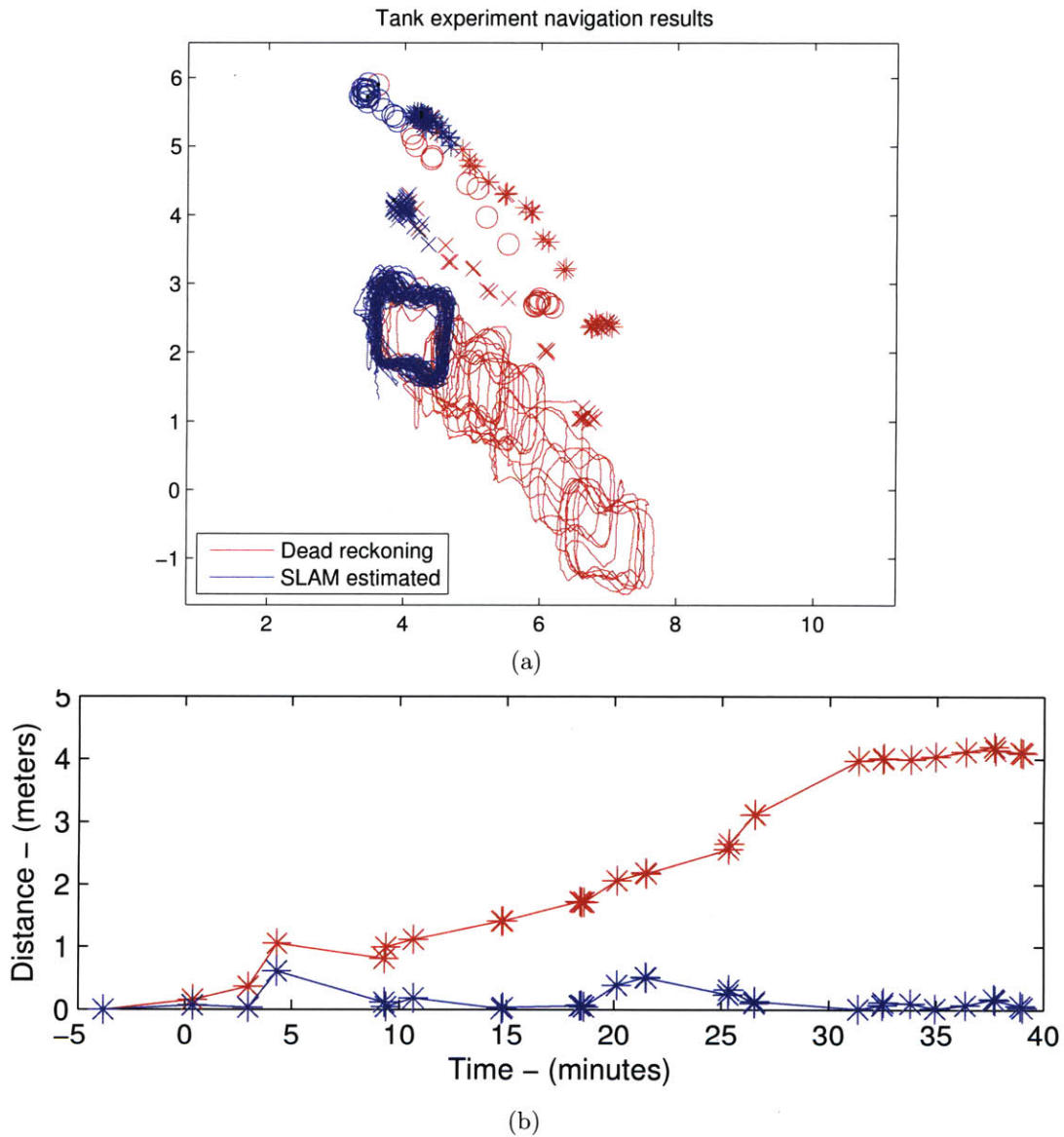


Figure 3-6: Evaluation of the tank experiment, comparing the dead-reckoned position estimate (red) and the smoothed trajectory (blue). (a) Comparison of the DVL and smoothed trajectories. The vehicle executed a box pattern for just under an hour. In the tank the water is very shallow and the DVL does not perform well, so that we see a considerable drift in the dead-reckoned trajectory. (b) Estimation error of a single target that was manually selected in several sonar frames. The dead reckoning errors grows with time, while the error in the SLAM estimated trajectory is bounded.

was significant, but the navigation system was able to correct for that by periodically acquiring registration with the map. Note that a similar problem occurs when the DVL is used in deep water near the limit of its range. It would not have been possible to operate using only the DVL, because the drift was large enough that the vehicle would have hit the walls of the tank. So to collect this data set it was actually necessary to run the navigation online.

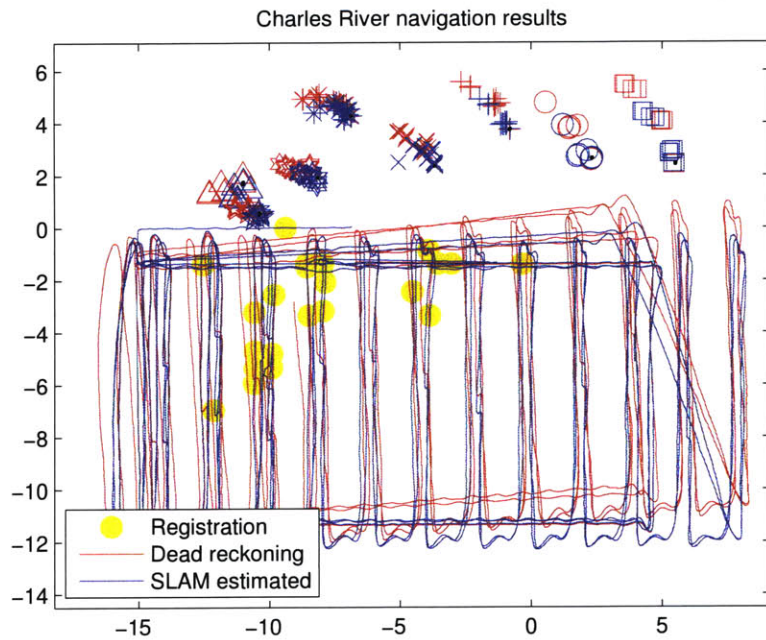
Fig. 3-6(a) provides a comparison of the dead-reckoning and the smoothed trajectory. To evaluate the performance of the system we manually labeled several objects and compared their estimated position, at each time, to the position of the object when it was first seen. Fig. 3-6(b) shows a plot of the position errors for one of the objects.

3.5.2 Charles River

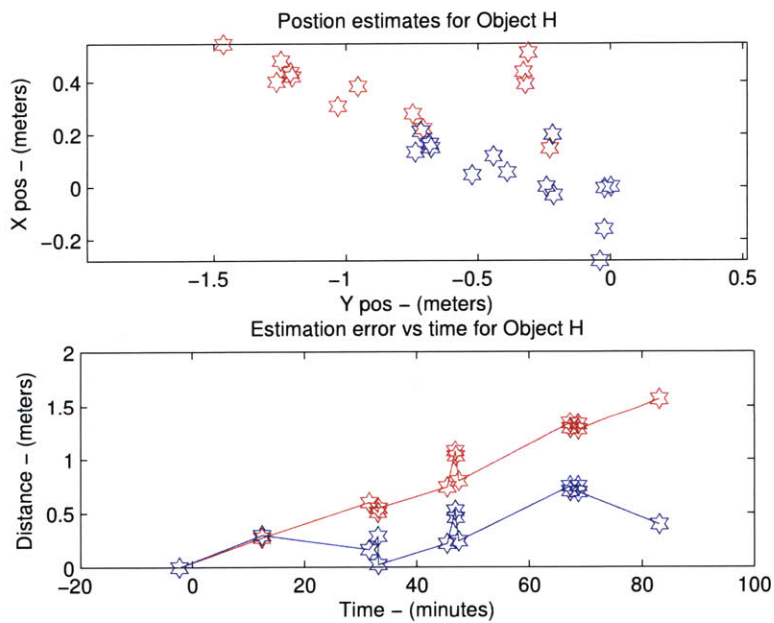
The purpose of this experiment was to emulate a harbor bottom inspection. One reason for performing this type of inspection is, when a foreign ship comes into harbor it is often required to inspect the area where it will dock. For this mission the vehicle traversed in a lawn mower pattern, with the sonar looking under the pier and collecting imagery of various objects lying on the riverbed.

The mission length was around 1 km, see Fig. 3-7 for results. One difficulty of the environment is the sparsity of the features. Good features were only visible from one side of the rectangle covered by the vehicle. Nonetheless, for around an hour and a half the vehicle was able to get registrations to earlier sonar images and improve the localization accuracy. During that time the heading estimate from the vehicle was around 4 degrees off and object prediction errors up to 3.6 meters. Using the SLAM estimate the maximum error was 2.3 meters and the mean was 0.7 meters compared to 1.15 meters for the dead reckoning.

After one and a half hour the navigation system accepted a wrong registration, which caused the estimated position to veer off from the true position. Building on the experience from these experiments we implemented a few improvements to the navigation system, that were subsequently tested on the recorded data. The key

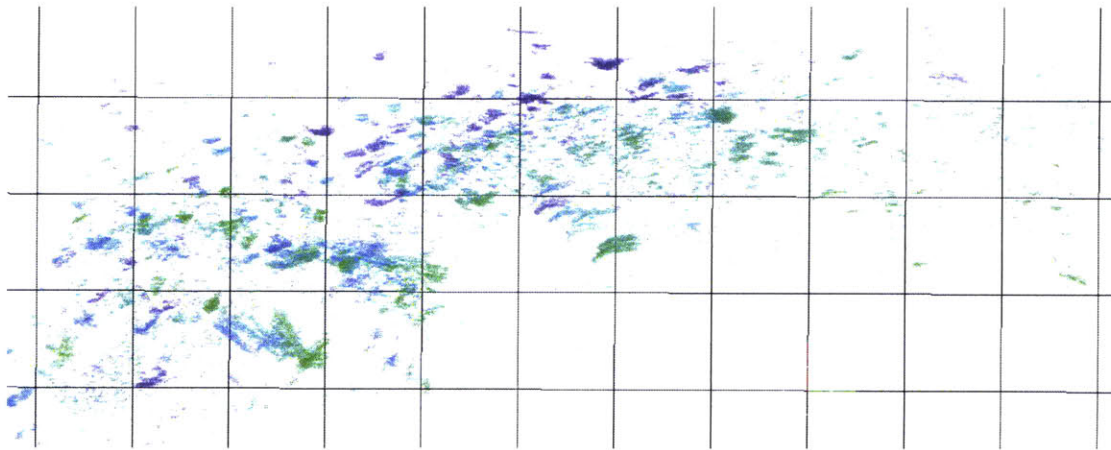


(a)

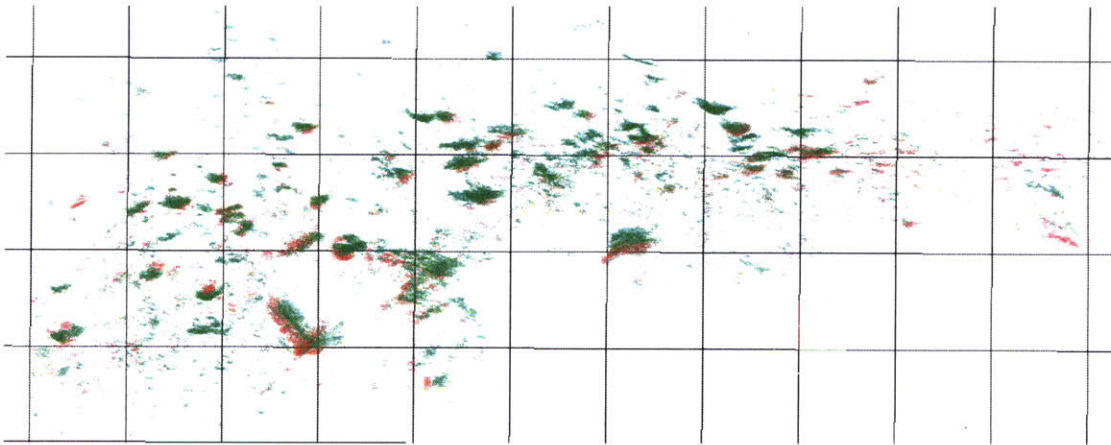


(b)

Figure 3-7: Evaluation of the Charles River experiment. (a) Trajectories with manually labeled landmark positions shown for reference. Results based on dead-reckoning alone are shown in red, our corrected estimates are shown in blue. (b) Enlarged plot of position error for one of the objects. The effects of the gyro drift are clearly visible.



(a)



(b)

Figure 3-8: Sonar map of the MIT Sailing Pavilion. Showing clearly the improved quality of the map created using the smoothing. The color represent the time the scans were recorded. For the smoothed map brown is the oldest scan and green new. In the dead-reckoning, green is the oldest and blue the most recent images.

components that were improved upon are:

- Improved feature extraction
- More selective registration process
- Feature decimation

By improving the feature extraction we were able to use a stricter acceptance criterion to determine if a given registration was a good match. The DIDSON frames arrive at 5 to 10 Hz and it is not possible to try to register every single incoming frame to all the potential frames at a given location. In the initial version of the registration, we always picked the most recent DIDSON frame and then tried to register it to all frames within a given range. However, by exploiting the fact that the vehicle has very high navigation accuracy over short distances, our computational resources can be used more efficiently by not trying to register to frames that are close in time, but instead choosing to match against frames that were recorded much earlier in the mission. This strategy improved the chances of finding large loop closures. Finally, feature decimation was incorporated, which improved the execution speed of the registration.

After implementing these improvements the data was reprocessed. For the complete mission, the distance traveled was 1459 meters and the duration was 133 minutes. The mean error went down to 0.31 meters and the maximum error was 1.2 meters compared to a mean error of 1.6 meters and maximum error of 4.9 meters for the dead reckoning. The improvement was possible because more registrations were accepted while wrong registrations were avoided. The development of SLAM algorithms that are robust to incorrect registrations is an important topic for future research.

3.5.3 King Triton

This experiment was targeted towards ship hull inspection, which is one of the main goals of the HAUV project. The acoustic imagery from a ship hull has different



(a) MIT Sailing Pavilion



(b) HAUV sitting idle and preparing to enter the murky river



(c) Charles River mission control. Brendan and Hordur monitoring the mission progress



(d) HAUV breaching the surface

Figure 3-9: Operations at the MIT Sailing Pavilion in the Charles River

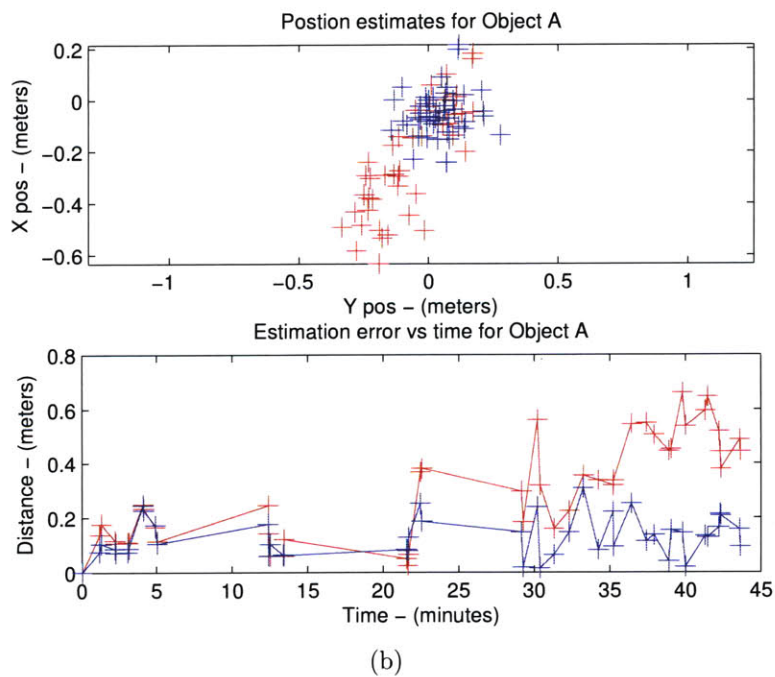
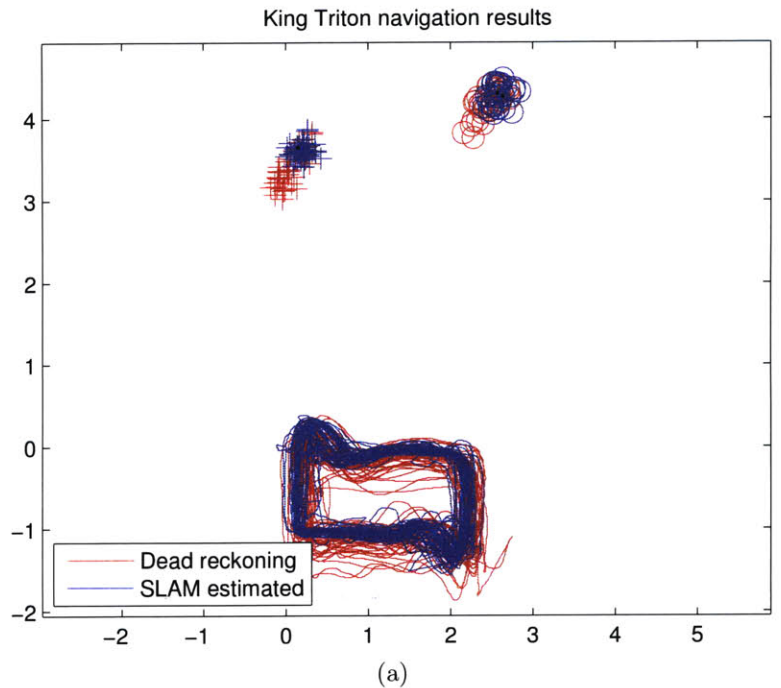


Figure 3-10: Evaluation of the experiment on the *King Triton* vessel. (a) Trajectories with manually labeled landmark positions shown for reference. Results based on dead-reckoning alone are shown in red, the corrected estimates are shown in blue. (b) Enlarged plot of position error for one of the objects. The effects of the gyro drift are clearly visible.

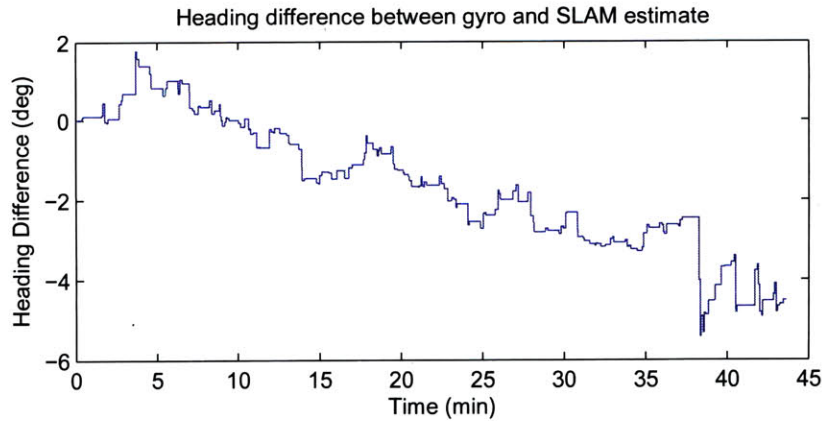


Figure 3-11: Difference between heading estimated during the *King Triton* experiments. The gyro has drifted by 4 degrees from the smoothed trajectory in 40 minutes.

characteristics than the seafloor, in that it is often more repetitive and generally has a more complex shape. Therefore it is important to test the system on an actual ship hull. The *King Triton* is 20 meters long with a flat bottom, it has several features on the hull that the system was able to use for localization, this included cooling pipes and zinc anodes. Note that the cooling pipes provide only limited constraints. Because of the small size of the boat the DVL was pointed to the seafloor instead of toward the hull, as would be done for larger vessels. An overview of one of the runs can be seen in Fig. 3-10 together with the position error for one the verification objects. The HAUV navigated in a box pattern for about 45 minutes, and repeatedly localized using features on the hull to correct the drift in the dead reckoning, keeping the position error within 0.3 meters compared to 0.66 meters for the dead reckoning.

3.6 Summary

This chapter has presented an algorithm for dense feature extraction from imaging sonar frames. The extracted features were used to compute constraints between different parts of the vehicle trajectory. When this information is combined with the vehicle's dead-reckoning, bounded navigation accuracy is achievable. To demonstrate the versatility of the system we gave results from several experiments. The next



(a) Preparing for operation on the *King Triton*



(b) HAUV deployed of the *King Triton*



(c) The HAUV ready to start its mission



(d) The *King Triton* vessel

Figure 3-12: Operations on the *King Triton* vessel

chapter will look into how this navigation system can be used on a larger scale and how it can support repeated operations over a long time.

Chapter 4

Architecture for persistent mapping

For long term operations persistent storage is needed for the map data. In the previous chapter we described a sonar-aided navigation method that provides drift-free navigation for a single mission. In this chapter we will develop a system that provides support for repeated missions. There has been much progress made in scaling mapping systems to support ever larger environments, using: sub-mapping [9, 61, 79], hierarchical organization [39] and biologically inspired methods [70, 71]. A relative formulation has also allowed for efficient implementations [82, 88] of the map estimation process.

Many of these recent methods use a pose-graph formulation, where the full trajectory is included in the state estimation – this addresses the “correlation problem” in SLAM [62]. This problem can be better understood by considering the Markov Random Field (MRF) for the problem. If the trajectory is kept around the graph will be sparsely connected, and inference on the graph can be carried out efficiently. On the other hand, if previous poses are marginalized out at each time step, the graph over the landmarks will end up fully connected, causing the storage and time complexity to be at least quadratic in the Gaussian case. This is demonstrated by the MRF in Figure 4-1.

Keeping the full trajectory around, poses a new challenge – the size of the estima-

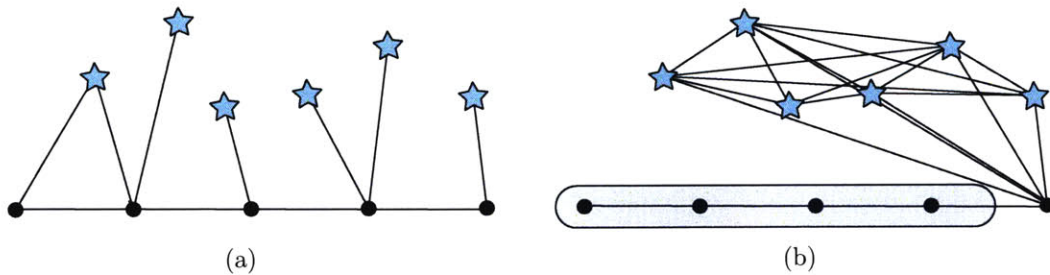


Figure 4-1: These figures show the effects of marginalizing out previous poses, and the effect that has on the number of links in the estimation problem. When past poses are marginalized out it introduces links between the landmarks, resulting in a fully connected graph.

tion problem will grow over time. Another issue is that as areas are repeatedly revisited dense clusters will form, which will be quadratic in the number of passes. Even though currently there are no solutions to this problem, there are several promising approaches, such as sub-mapping [79], hierarchical [39] and tree [52] representations. Normally any robot would be interested in looking at a map that corresponds to the most recent knowledge of the world. Marginalizing out all the poses that are not a part of the current map will yield a dense structure. Another alternative would be to simply throw away old poses. The reason that this might be feasible is that some constant number of passes may be sufficient to represent the map with sufficient accuracy.

4.1 The map database

To achieve perpetual autonomous operation a system is needed that can store and organize the knowledge acquired by the agents the system supports. We envision that the system should not restrict itself to support only live-long operations, but rather serve as a repository of knowledge for future generations of agents. The distinction we want to make here is that the mapping system should outlive any individual robot. Of course one might argue that eventually live-long will mean eternity. In the rest of this chapter the requirements of a system that can support multi-session / multi-agent mapping will be defined. Then a design for this system is proposed and

finally an experimental implementation that realizes some aspects of the system will be described.

The key component in the system is the map. It is possible to represent the map in different ways. On a high level there are metric or topological maps. Topological maps are usually sufficient when being used for navigation, i.e. finding paths from one place to another. Metric maps are useful when accurate measurements of the world are needed, e.g. ensure full coverage of a structure being inspected. They can also be useful if there is a need to reason about unexplored areas. The drawback is, that it is normally more expensive to create and maintain metric maps. Furthermore, the actual structure and entities contained in the maps can vary. Some of the different map representations include:

Landmarks are objects or places that can be recognized by the robot's sensors. This can include some point features in the environment, such as lines, corners, cylinders, and so on. They are also referred to as geometric beacons. In different formulations of the SLAM problem these landmarks are relative to some coordinate frame, such as the global frame or some local frame where the landmark was measured from.

Pose graphs are a representation where landmarks are not explicitly maintained in the state estimation, instead only the vehicle poses are estimated and constraints between these poses. The actual objects in the map would then be associated to those poses. The best known examples for pose graphs are the LIDAR maps where constraints are derived from pairwise registration of laser scans [63, 64], and the pose of each scan is estimated. A global map can be rendered by drawing each scan transformed by its pose.

Gridmaps or occupancy grids are discretized representation of the world. These are commonly used with particle filters. The gridmaps are useful for path planning and localizing can be done efficiently. The biggest challenges for gridmaps are changing environments, and large updates, such as when loop closures occur.

Topological graphs represent a connectivity of distinct places, where connectivity between places usually means that it is possible to traverse from one place to another. These maps are easier to maintain, compared to their metric counterparts, and work well for path planning, where a path between two places is simply the path in the graph.

These methods do not exclude each other and all have their strengths and weaknesses. The pose graphs can include landmarks and the topological maps could include pose graphs locally while globally manage a topology over these local maps. Gridmaps are useful for path planning and obstacle avoidance and they can provide efficient localization. The gridmap can be created from a posegraph. This shows that these different methods can actually be combined in different ways to address different problems.

Table 4.1: Overview of SLAM algorithms.

Name	References
Extended Kalman Filter (EKF)	Self et al. [89]
Sparse Information Filter (SEIF)	Thrun et al. [96]
Exactly Sparse Information Filter (ESEIF)	Walter et al. [102]
Smoothing and Mapping (SAM)	Dellaert [18]
Incremental Smoothing and Mapping (iSAM)	Kaess et al. [53]
Conjugate Gradient	Konolige et al. [57]
Stochastic Gradient Decent	Olson et al. [81], Grisetti et al. [40]
Bayes Tree	Kaess et al. [52]
Thin Junction Tree filter	Frese et al. [83]
TreeMap	Frese et al. [35]
Rao-Blackwellized particle filters (FastSLAM)	Montemerlo et al. [73]

Just as there are many ways to represent the maps there are also many ways to estimate the state. Several of the algorithms that have been used are given in Table 4.1. This is not an exhaustive list of all the different approaches. These methods have their different strengths and weaknesses. Even though there has been lot of activity in this area there has been less attention paid to how to represent the maps for persistence and sharing in online operations. Examples of long term operation include the Mobot museum robots which operated for 5 years [80], using static maps but still in the presence of dynamic objects, Biber et al. operated for 25

days maintaining a map history consisting of long term and short term maps [6]. Our goal for the map database is not to replace the optimization machinery that already exists; instead it should support those different map representation and estimation methods. In our implementation a full pose-graph formulation is used and the graph is optimized using iSAM, which is an incremental non-linear least squares solver.

4.1.1 Assumptions and rationalization

When specifying the requirements for the system, one of the key decisions we made was that there should be a central database that could serve multiple robots. We can ask the following question: why does a centralized robotic map database make sense? First, the map might not easily fit on a single robot, i.e. a visual map of Boston. Secondly, it might be difficult to share the state between all the robots. The drawback is that now the robot is reliant on a central server. To address this the robot is required to be self sufficient in case of loosing contact with the server, so it will still operate, even though it might not perform as well. Konolige et al. used one team leader in the multi-robot mapping to distribute the map data, and as a fallback each robot could also act independently [59]. Another concern might be scalability; looking at the amazing scalability in services such as Google search, once can see that with the right design, scalability for robot mapping should be possible to achieve.

Another decision we made was that the database should collect all the data acquired by the robots. Storage is assumed to be cheap, so it is possible to store majority of the sensor data. However, for high bandwidth sensor like cameras, it might not be feasible to store the data at full rate at the moment. This is an important area of research: how do we decimate the data while keeping as much of the history available?

It is assumed that the vehicle has wireless access most of the time. In most situations, land robots can be expected to have almost continuous high speed wireless connection, for underwater vehicle that is not the case. The underwater vehicles will normally only have wireless access when on the surface, though there has been work done on optical communication underwater [100], it is still limited in speed and range compared to the land alternatives. This means that for underwater application, the

vehicle would start with downloading a part of database the would be needed for the operation, and then when the mission finished, it would transmit any new data to the server.

4.2 Requirements

When considering the requirements for the system it should be noted that there are two major applications of the system. The first requirement is to support map making, localization and navigation for operational robots. The second requirement is to support research, development, and evaluation of mapping systems. For us as researchers it is important that the system assists us with these tasks, by being flexible in supporting different algorithms and by allowing for comparison of different algorithms and analysis of the mapping data. This makes it easier to build better mapping systems. Next the requirements for the robotic map database will be described in more detail.

4.2.1 Core features

Multiple sessions are supported by the system. A single session corresponds to a continuous operation of a single robot. By supporting multiple sessions maps can be built by distinct passes through the environment. This is also a key component for enabling long term operations. Data from each session should eventually be integrated into the global map.

Multiple robots can access that database at any time, either to request services from the system or to feed new information the database.

Spatial separation is required to allow robots to request subsets of the global map, assuming that it would not be feasible for a single robot to store the complete database locally. For example, an AUV inspecting in Rowes Wharf would not need the map of East Boston harbor.

On-line access is required to support multiple robots.

Durable storage should be provided to assure minimal data loss.

Change history should be available. This requires that the database should collect all the data it receives.

Support for development needs to be considered. This means that the system should be designed in such a way that access to intermediate results should be provided. Support for multiple optimization methods, and comparison of those methods, should also be provided. Functions that help with evaluating quality of the map at any given time should also be provided, as should tools for subscribing to certain types of data for visualization or other analysis.

Flexible design. The system should be able to support different types of map estimates. For a large systems it can be expected that different types of robots operate at the same time. It should be possible to support the various types of robot middle-ware that is available. The robots might also have different sensor modalities, and the system should be able to collect this data and connect into a shared map.

Robust in the sense that it should be able to identify wrong information received from the robots being supported, whether it is intentional error or not.

There should be a central repository that robots can fetch a subset of the world from. Robots should be able to push their sensor information to the map database. The robots can also forward derived information like visual odometry constraints, or constraints derived from scan matching of laser-scans. Visual odometry constraints are a good example of decimating high bandwidth data. But periodic camera frames might be pushed to the server, to facilitate appearance based localization and rich reconstruction of the environment.

The database should not rely on any single robot and each robot should be able to function without continuous connection to the central database. This can be done by running a local instance of the database on the robot.

This system can be considered as an unified approach to persistent mapping, it enables long duration operation. The system should also be supportive of development tasks. That is, allow different algorithms to be run, and provide visualization facilities of the inner workings of the system.

4.2.2 Services

For any robot there are many tasks that need to be performed, from low level control, sensor processing and up to decision making [60]. It should be possible to use the map database to assist with completing some of these tasks. Even though some of the tasks might not be in the scope of the actual database, we still include a list of few relevant tasks with the intent of guiding the design of the system by demonstrating possible uses of the system. In addition we will also list some of the basic services the database needs to provide.

Path planning is a crucial service for most robots, it resolves what actions the robot needs to take to get from one place to another. Gridmaps have been successfully used for path planning, and it has also been shown that topological map can also serve as good representation when navigating between places.

Obstacle avoidance is related to the path planning, but it typically considers only the short-term view of the world to avoid immediate obstacles, like dynamic objects that normally would not be in the map. If the database is running centrally on a server it would not be feasible to involve the database in obstacle avoidance. On the other hand, a local copy of the server might be able to provide a low latency view of the world.

Semantic information should be available from the database, this can include information like traversability, doors, elevators, etc. This should be supported in an extensible way, such that a later time new algorithm can be added to the system to enrich the mapping information that is available. To connect with the previous items, then obstacles can actually be seen as one type of semantic information,

Localization is another key component in any robotic system and is obviously a necessary feature for the mapping system.

Re-localization refers to localizing a robot without including any prior knowledge about its location. This is needed to provide robust operations. In the case a robot gets lost it will need to re-localize, it is also a useful feature in many other situations, e.g. when a robot first starts, and when there are undetectable discontinuities in the vehicle motion, like happens when an elevator is used.

Sub-map queries for parts of the map. It should be possible to request only a subset of the complete map, both spatially and temporarily.

Map updates should be supported. The robots using the database should be able to feed their sensor input to the central server. This also facilitates sharing of world information between multiple robots.

4.3 Design

In this section we describe a design of the system that addresses the requirements detailed in the previous section. A systems overview is shown in figure 4-2. It manages basic mapping concepts like sessions, maps, zones and views. For persistence it relies on a data storage system, this could be a simple file structure or a more complex storage like a relational database management system. The system defines two types of interfaces, one is for the storage layer the other is for communication with robots and various services like path planning and change detection. The system collects data from the robots and runs a map estimation internally to maintain a map of the world. The robots and other services can then query the system for this data.

4.3.1 Data model

Figure 4-3 shows an overview of the data model for the map database. The key entity is the map node which corresponds to a single point in time for a single robot. The

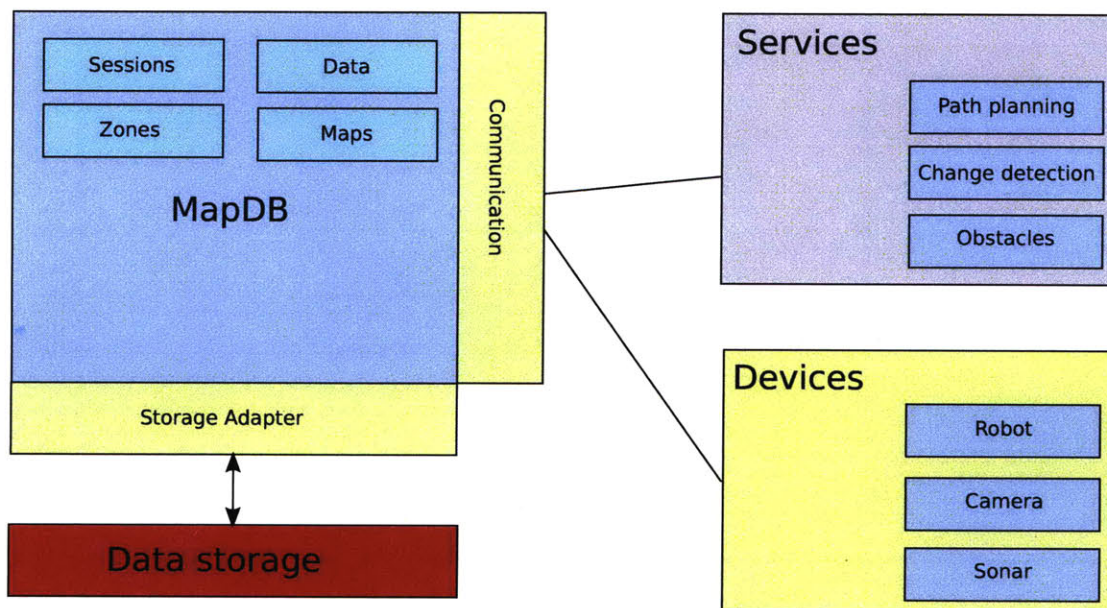


Figure 4-2: System overview of the Map DB. The core components are the sessions, zones, maps and sensor data. The system defines a set of interfaces that can be implemented to support different types of robots and data storages. Camera and sonars are example of devices that can send data to the database. Path planning is a service that can query the database to aid with generating navigation plans through the environment.

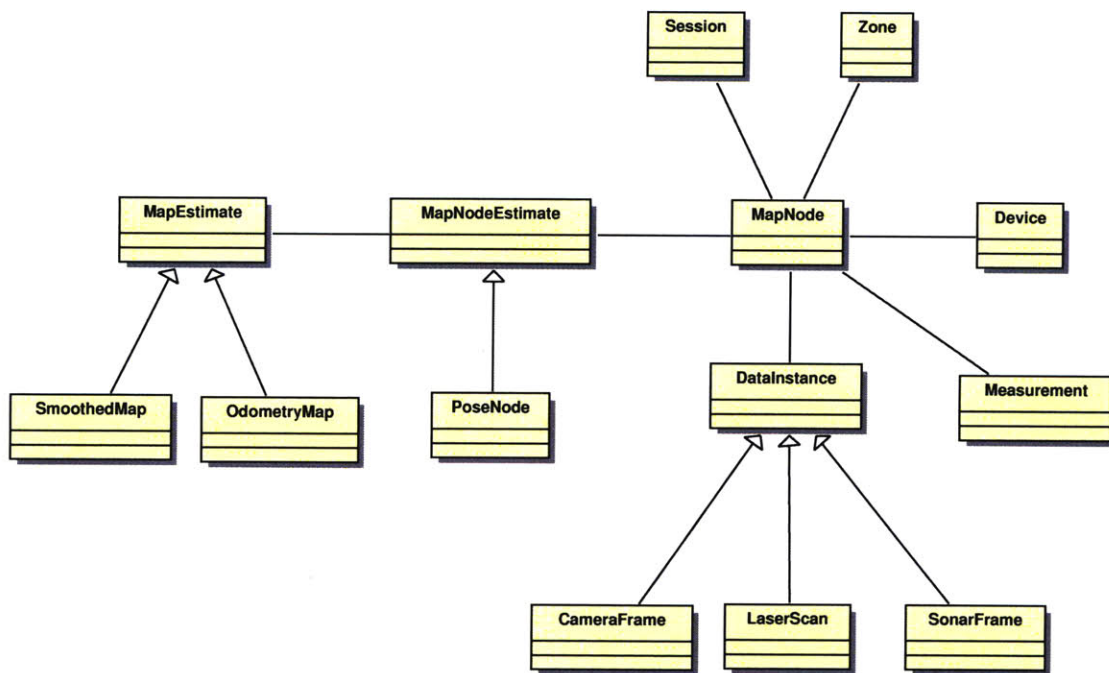


Figure 4-3: Overview of the Map DB data model. The central entity in the data model is the *MapNode*, which links together sensor data, measurements, state estimates and more. There can be many different types of data instances and map estimation methods.

map node then has some associated variables and then one or more estimates of those variables. The database stores raw sensor data in addition to any derived data that various algorithms running on the system might create, e.g. extracted features to describe a given laser scan or a camera frame.

The core entities in the database are:

Device

A robot is represented by a hierarchy of devices. A device has a pose that is relative to a parent device. There is one root device that defines the coordinate frame of the robot.

Map

A map is represented by an estimate of robot poses and sensor measurements at those poses. In addition the map will have an associated estimation method, and depending

on the estimation method some related data. By allowing more than one estimate it is possible to support systems that might like to maintain different estimates for different types of robot. This can also be useful for system that would like to perform multiple hypothesis tracking. From the development perspective this is also useful to experiment with different estimation algorithms.

Example: A posegraph estimation using iSAM. We store the estimated means of the poses and all the constraints between the poses.

Session

Each device data will have an associated session. A session corresponds to some continuous period of time for the robot. A session does not need to be restricted to a single run, a robot might decide to restart a session if, for instance, the robot gets lost and decides to start a new session. That would start a new trajectory that would be in a new reference frame that does not have any transformation to the global frame until the robot successfully re-localizes.

Zone

The map data can be organized into zones. The idea is that a zone is a reasonable sized chunk of data that should fit on a single robot. For small environments, e.g. one or two floors of a building, or a whole building might be a reasonable chunk.

DataInstance

The system can store data from different type of sensors. Each sensor measurement is stored as a *DataInstance* associated with some *MapNode*. It is also connected to the *Device* that created the data, and the transformation to the base device at the time the data was collected.

DerivedData

The derived data is information that is created during run-time. From an implementation perspective this is not much different from the *DataInstance*, but there is a

conceptual difference that we want to emphasize in the design. Derived data can be something like the NDT transform used in Chapter 3, feature descriptor describing the main view of the node or some classification of the data points. In general any computationally intensive operation might be worth storing in the database to share with other users.

4.3.2 Execution model

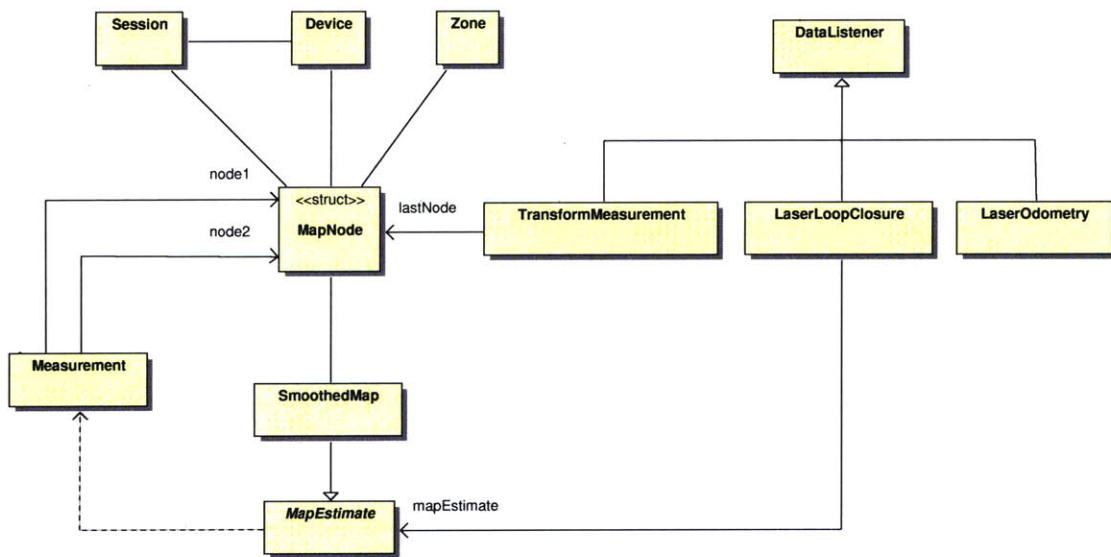


Figure 4-4: Overview of the Map DB execution model. The execution model is a collection of classes that aid with managing the map during runtime. There are classes that handle data processing and map estimation.

Now we turn our attention to the execution model of the map database. In addition to the data entities the execution model includes transient components that provide services while the system is up and running. First, let us look at the core components.

MapDB is the main interface to the map database. It provides methods to starting sessions, sending in data, registering listeners, map estimates and channeling data into corresponding maps. At runtime a set of estimation algorithms is maintained.

DataListener is an interface for any component that is interested to process incoming data, this can include components like a scan matching algorithm that registers a pair of laser scans.

DataManager is an interface for a database implementation. It declares methods for creating, retrieving, updating and deleting the various data types in the map database.

Inputs are used to direct measurements created by the various *DataListener* components. It is most common that inputs are defined on *MapEstimate* components.

MapEstimate is a component that receives measurements through defined inputs. The *MapEstimate* is responsible for maintaining an estimate of the pose for each *MapNode*. In addition it can publish viewer collections to be used for visualization of the state of the estimation process. The viewer collection can include information like accepted and rejected registrations, key frames, and etc.

ViewerCollection is used for visualizing the mapping process. The MapDB can send mapping information like map nodes, registrations, extracted sonar features to a component that implements this interface. That component is then responsible to transmit the data to the actual viewer.

4.4 Implementation and applications

To test the design of the system we created a prototype map database that implements some features described in previous sections. The development has been carried out using recorded data from actual robots. The core system is implemented as a collection of C++ classes. All the data storage services were in-memory. The communication method used is the Lightweight Communication and Marshalling (LCM) library [45]. LCM was initially developed to be used by the MIT DARPA Urban Challenge Team. We tested the system both using sonar data from the HAUV and then a laser scan dataset.

4.4.1 LCM adapter

The LCM Adapter supports several message types: laser scan, sonar images, odometry measurements and HAUV specific messages. The adapter is responsible for listening to the incoming LCM messages and converting the data to the correct format and then send it to the map database. A part of the LCM software used is a graphical viewer that listens to various LCM messages and renders that information on the screen, so the operator can monitor the operation of the robot. One feature in the viewer that we make extensive use of is the so called "Collections" viewer. The collections consists of objects and links, any program can transmit several collections of objects and links to the viewer. The operator using the viewer can then easily choose which collection to show. We extended this collection component by adding a point cloud data type. This allowed us to render complete maps of laser scan data and a mosaic of extracted features from the sonar images. Figure 4-5 demonstrates the use of those point clouds, to better explain the sonar map, actual sonar images have been overlay-ed on the map and connected to the corresponding features.

4.4.2 Map estimate

For the map estimation the *incremental Smoothing And Mapping* (iSAM) library was used. As was showed in Chapter 3, iSAM maintains an ML estimate of the full vehicle trajectory. The constraints used were either from vehicle odometry, sonar frame registration, and in the laser case a scan-matching library was used that can compute a transformation between pair of scans. Using iSAM it is possible to maintain an estimate of multiple vehicle trajectories, this is done by introducing so called anchor nodes, as described by Kim et al. [55]. The scan-matching library that was used implemented by Bachrach et al. [2, 3]. The smoothed map manages a collection of accepted and rejected constraints, using that the mapdb instance can forward this information to the viewer.

```

// Create an instance of the map database runtime
MapDB* mapdb = new MapDB( viewer , datamanager );

// Create a map estimator
SmoothedMap* smoothedMap =
    new SmoothedMap( mapdb, 0, "Smoothed" );

LaserOdometry* laserOdo =
    new LaserOdometry( mapdb );
mapdb->addEstimate( smoothedMap );
mapdb->addInput( smoothedMap, laserOdo );
mapdb->addListener( "LASER", laserOdo );

```

Listing 4.1: Initializing the MapDB

4.4.3 Laser processing

For the laser processing a scan-matching library to computes constraints between pair of laser scans. This was used to create two *DataListener*'s implementation, one only considered consecutive poses, and those poses were used as odometry constraints. The other implementation searched through all nearby laser scans, where nearby was decided based on the overlap of two scans. The overlap was computed by considering a bounding box of the two scans, and the area of the intersection of those two boxes was used as a criteria to decide if a registration should be tested.

4.4.4 Examples

To better demonstrate the map database we will look at some usage scenarios. The code examples are based on the map database implementation but have been simplified to better convey the main idea.

Initializing

First, let us look at how the system can be initialized. Listing 4.1 shows an initialization code that instantiates a single map estimate, the *SmoothedMap*. Then there is one *DataListener* that listens the the "LASER" channel for incoming laser scans. The laser odometry object will then send out odometry measurements that are de-

```

void onTransform2dData(Transform2D msg,
                        string channel)
{
    // Find the device associated with the channel
    DeviceSessionPtr ds = getSession(channel);

    // Convert LCM transform to MapDB transform
    Transform* t = new Transform(
        msg.x, msg.y, 0,
        msg.heading, 0, 0);

    // Send the transformation to the database
    mapdb->onData(msg.utime, ds, t);
}

```

Listing 4.2: LCM 2D odometry handler

rived from consecutive scans. This is added as an input to the smoothed map. The viewer is a previously constructed viewer object.

Transformation handler

A common task is sending odometry constraints to the map database. This is shown in Listing 4.2. It is assumed that the messages come in on some specific channel. Using the channel the currently active session is fetched, and the associated device. The map database defines a transform type, which is a 6D transformation between two poses. The handler converts the incoming message into the the map database representation and notifies it about the new data.

Loop closure algorithm

One of the crucial operations in a mapping system is to detect loop closure. Listing 4.3 shows a loop closures that uses point clouds, which is one of the standard map database data types, tries to register to the closest point cloud in the database. The result from all registrations is sent to the map database as a measurement, even though a registration failed. The standard measurements contains a transformation between two poses, the covariance of that transformation, and references to

```

void findLoopClosure(PointCloudPtr & data)
{
    // Search database for nearby scans
    PointCloudPtr pointCloud = findClosest(data);

    // pointCloud contains a nearby scan

    // Test registration
    Transform t = doRegistration(data, pointCloud, score);

    // Send a measurement
    Measurement m;
    m.transform = t;
    m.node1 = pointCloud.node;
    m.node2 = data.node;

    // All attempted measurements are forwarded
    if (score > threshold) {
        m.setAccepted(true);
    } else {
        m.setAccepted(false);
    }

    mapdb->addMeasurement(m, this);
}

```

Listing 4.3: Loop closure service

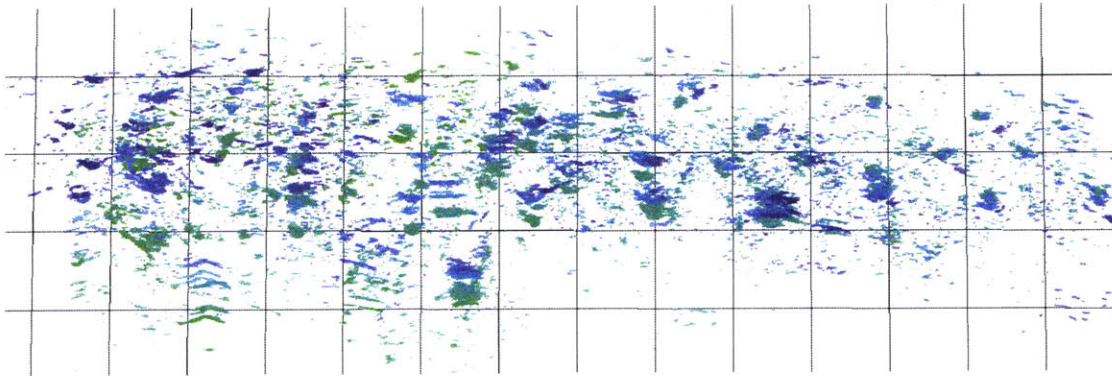
the corresponding map nodes. This is an example of a feature that is useful during development, it allows the user to visualize, not only successful registrations, but also the failed registrations (see Figure 4-7(d)). The failed registrations give insight into how well the pre-filtering, of which views to use, works.

Comparing results

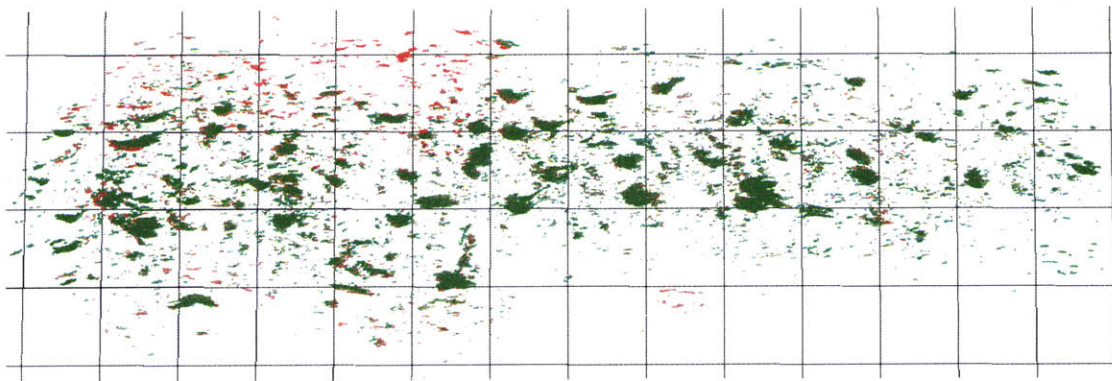
Figure 4-5 shows a complete sonar map from the Charles River. The HAUV imaged the river bottom, under the Sailing Pavilion dock. In Figures 4-5 and 4-6(b), the first map uses only dead-reckoning and the latter uses sonar constraints. In the smoothed map the features stick tight together while they are drifting away in the dead-reckoning, as would be expected.



Figure 4-5: Sonar map of the MIT Sailing Pavilion. The map consists of all the features in the key frames used. The vehicle trajectory was smoothed using sonar frame registrations. The purple link show the loop closures that were used. The raw sonar images have been overlaid to better illustrate what the objects added to the map look like.



(a) Sonar map of the MIT Sailing Pavilion using dead-reckoning.



(b) Sonar map of the MIT Sailing Pavilion using smoothing and mapping.

Figure 4-6: Comparing sonar map from dead-reckoning and smoothing. The images show composite map of all the extracted features. The color represent the time when a particular key frame was added. The key frames are rendered from the oldest to the most recent. (a) is using only dead-reckoning while (b) includes loop closures from sonar frame registrations. It is apparent that there is no drift in the smoothed version while the features can be seen drifting off in the dead-reckoning map.

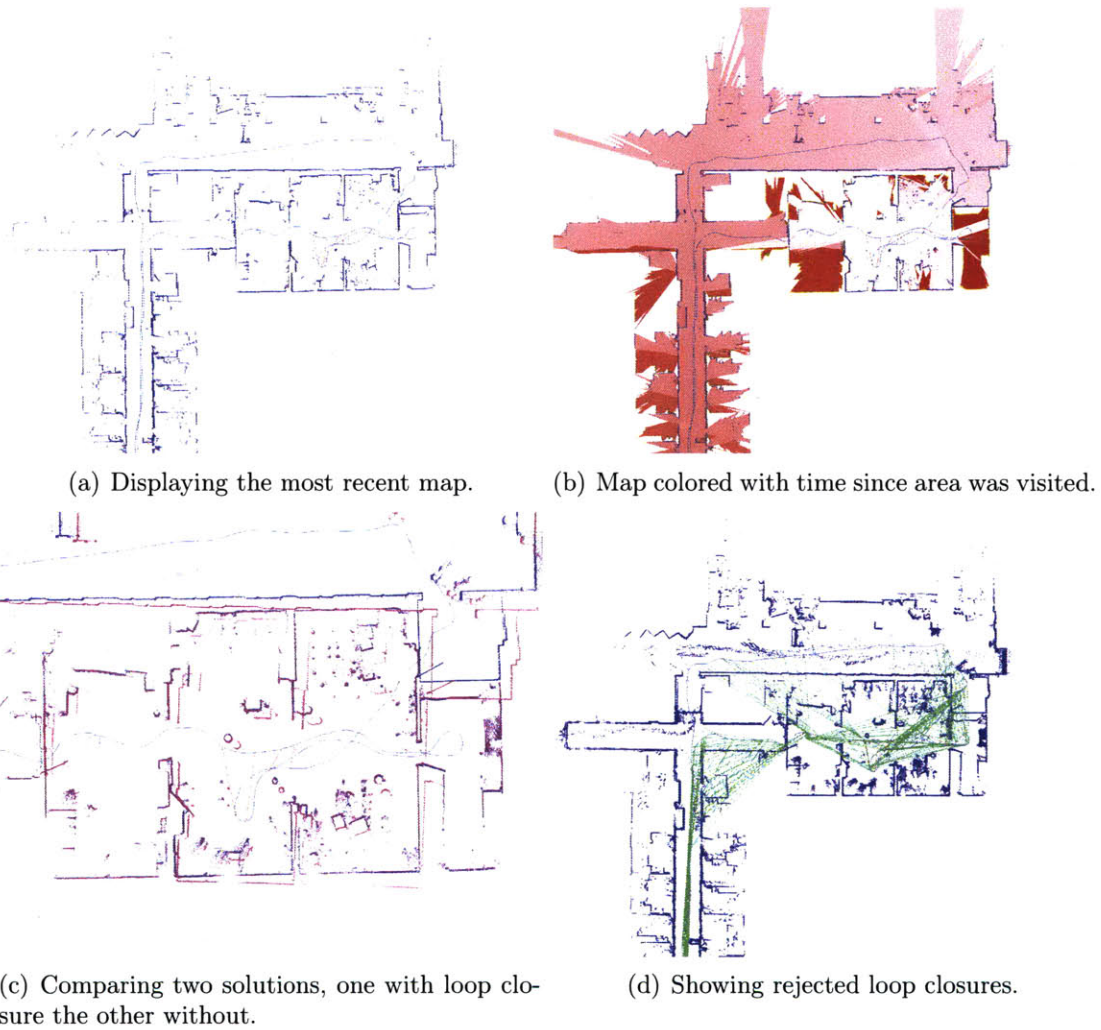


Figure 4-7: Map database laser scan examples. Here are shown results after processing laser scans from the Biber and Ducket datasets. (a) included the most recent version of the map. (b) shows a age colored map, red are old poses and white are new. (c) Two estimates overlaid on top of each other. The blue includes loop closures while the purple does not. (d) A map showing the rejected loop closures.

Gridmaps and path planning

Gridmaps can be used for path planning and general visualization of the environment. If the map consists of laser scans then it is possible to create the most recent map by simply rendering filled laser scans with the painter's algorithm. That is we start rendering the oldest scan and finish at the most recent. The result map, will be a gridmap of the most recent view of the world. An example of this method, implemented in the viewer is shown in Figure 4-7(a).

Surveillance

A variant of the gridmap, that is useful for surveillance tasks, is the age colored gridmap. It works the same way except each scan is colored based on the time the scan was recorded. An example of this is shown in Figure 4-7(b). The red spaces are those that have not been recently visited. A surveillance planner could use this service from the map database to aid with planning frequent surveillance of some area.

Visualization

One of the feature that was included in the map database were viewer collections. The collection can contain nodes, links and point clouds. The point clouds can be used to render laser scans, extracted sonar features, 3D point clouds from stereo cameras among other things. The *SmoothMap* component publishes its nodes and links. The links include both the odometry links and loop closures. Another benefit of the visualization system is that it easy to visualize multiple map estimates. Figure 4-7(c) shows two estimates overlaid on top of each other. The blue map is an estimate that includes loop closures, while the purple only contains odometry links. In the same figure, the loop closures are visible as links across the vehicle path. In Figure 4-7(d) all the rejected loop closures are shown in green.

4.5 Summary

In this chapter we have described an architecture for persistent, long term operations. The main requirements for such a system were identified and a design presented that realized those requirements. The prototype that was built demonstrates some of the concepts behind our design. It shows the flexibility of the design, resulting in a system that can work with different types of robots and sensor modalities. It will be possible to serve robots running on different platforms, by simply adding a new adapter implementation for each platform. Furthermore, the mapping system has been a useful tool to explore and compare different estimation algorithms.

There are still many steps to go, from the prototype to the next generation of the system. All the experiments performed with the system were done in post-processing, so analyzing the performance for real-time operations is one of the steps. Running a local and a central version of the database and then synchronization between them is another thing we have yet to look into. Even though there might be a long way to go, we believe this work is a valuable contribution to the concept of robotic databases.

In the next Chapter we will look at algorithms that can be used to create indexes that can later be used by any vehicle that needs to localize relative to some map.

Chapter 5

Re-localization

This chapter considers methods to efficiently compare the robot's current view to other places that have been visited before. This is a key service for any map database system. It is not feasible to query the database directly with the raw sensor data. So the view needs to be encoded in a way that allows for efficient indexing and a search for similar places.

5.1 Features

We propose a feature descriptor for 2D point clouds that is inspired by the work by Magnusson et al. for 3D point clouds. The raw sensor measurements are range measurements in the vehicle's reference frame. If the vehicle rotates or moves, then the ranges can change drastically. The features must be invariant to these changes, so that two views being compared do not have to be at the exact same place.

The normal distribution transform (NDT) algorithm from Chapter 3 is used to form the feature descriptor. As was mentioned earlier, first the ranges are transformed into Cartesian coordinates. Then the points are placed into corresponding grid cells, the mean and covariance is calculated for all the points that end up in the same grid cell.

In our case we are not interested in the mean, because the mean will change if the vehicle moves slightly. On the other hand, the covariance tells something about the

shape of the points. Let $\lambda_1 > \lambda_2$ be the two eigenvalues of the covariance matrix. Then if the ratio $\frac{\lambda_2}{\lambda_1}$ is small then the points are arranged linearly otherwise otherwise they are arranged in a more circular shape. In an indoor environment this can correspond to walls and corners.

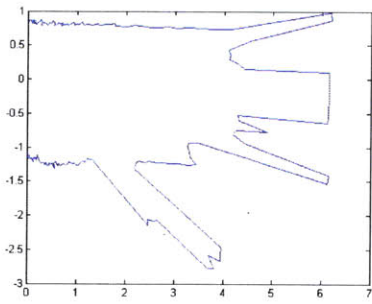
The feature vector for a collection of points is then

$$f = \{n_0, n_1, n_c, l_1, l_2, l_3, l_4\}$$

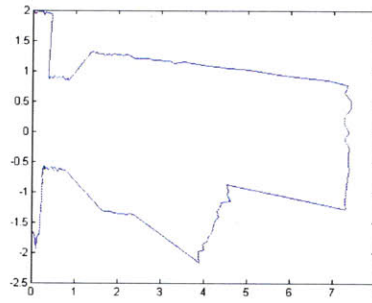
Where n_0 and n_1 are the number of cells that have fewer than 10 points and 10 or more points respectively, n_c is the number of cells classified as circles. For the cells that are classified as lines we have l_1, l_2, l_3, l_4 which correspond to different angles of the eigenvector for the largest eigenvalue, i.e. the principle component of the data. The angles are discretized into four different sectors at 45 degree intervals, notice that lines rotated by 180 degrees are at the same angle. Because all measurements are relative to the vehicle's coordinate frame, the feature vector is not invariant to rotation. To address this issue we ordered the line counts so that the largest value, or the most dominant heading, was picked as zero heading. This is similar to the method used in [66].

5.2 Comparison of different places

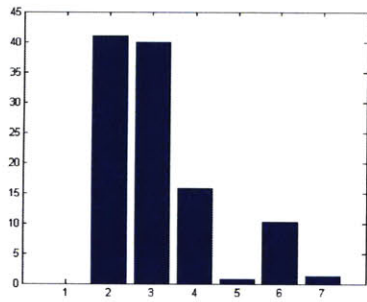
Magnusson et al. [66] looked at the absolute difference of two feature vectors to classify if two point clouds were from the same place, and looking for the place that had the smallest difference to make the final choice. Using the feature vectors described in previous section we tested classifying on the Euclidean distance between two feature vectors. Next section will look at methods to quickly locate either the place with the least difference or all places that are within a given distance, in feature space, from the current view. In a related work we also experimented with using more advanced machine learning techniques to compare different features vectors, but that is out of the scope of this work.



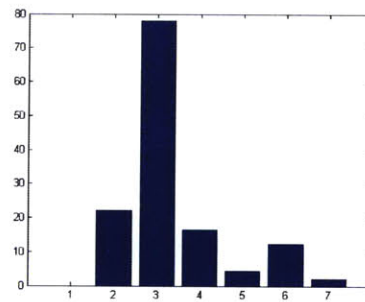
(a) Laser range scan A



(b) Laser range scan B



(c) Features extracted from range scan A



(d) Features extracted from range scan B

Figure 5-1: Example of laser scans and extracted features

5.3 Fixed radius nearest neighbor search

In this section we look at two algorithms one is kd-tree [34] and the other is balanced box-decomposition (BBD) tree [1]. Searching for high dimensional data is a complicated task. Even if the data is sorted by some column then for a distance query it is still possible that all the data points need to be considered. For instance, if the features are binary vectors, then sorting on any single column will not help with finding every vector that differs from the query string in two places. It is possible to look at one of the principle components of the data and project on that vector, that could help narrow the search a bit, but that really depends on the data being used. For these reasons it is interesting to look at some data structure that allow these queries to be answered efficiently.

5.3.1 KD tree

The kd-tree is similar to a binary search tree, in the sense that it organizes the data into a tree structure. At each level of the tree, it splits the data in two parts along a single dimension. It sequentially goes through the dimensions and splits until the number of points are below some threshold. The terminal nodes are called buckets. This algorithm guarantees that the cardinality of points at each level goes down exponentially. If N is the number of points and k the number of dimensions then the tree can be built in $O(N)$ space and $O(kN \log N)$ time. The expected time for queries is $O(\log N)$, but as mentioned in [1] the asymptotic running time for nearest neighbor hides constants that grow as 2^d . In the data set we tested we did not see exponential growth with d , so the data might also have a big impact on the complexity, also in our case only the fixed radius search was used.

5.3.2 BBD tree

The bbd-tree tries to improve on the kd-tree by trying to maintain more even partition of the data, and also allowing to zoom in quickly on clusters. It has two operations, a fair split and shrink. The split is just like in the kd-tree the data is split by

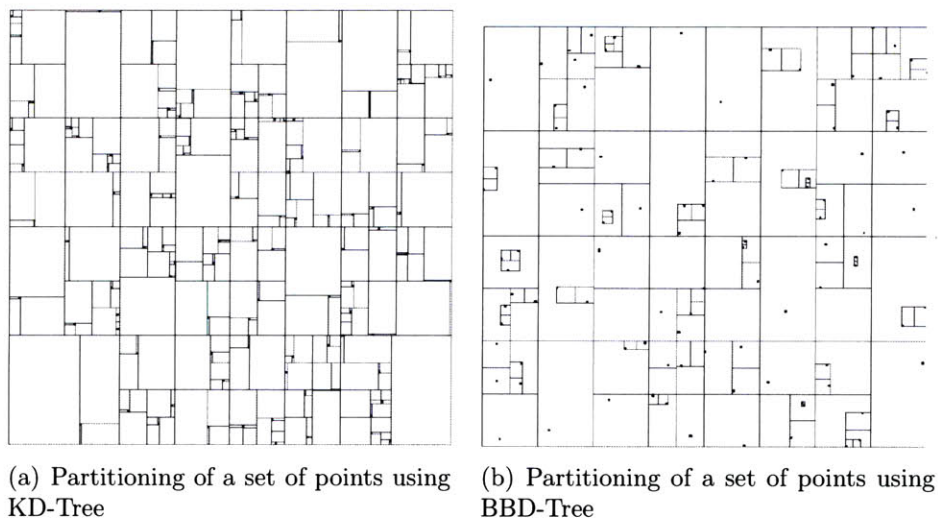


Figure 5-2: Comparison of the partitioning using KD-Tree vs BBD-Tree. The BBD-Tree has cells of much more even size while in the KD-tree you can get those skinny rectangles

hyperplane lying along one of the dimensions. The shrink defines a box inside the outer box, so the data is split into outside the box and inside the box. The kd-tree tends to generate skinny boxes while the bbd-tree avoids that by maintaining a fixed aspect ratio of its boxes. Doing a shrink is expensive because to check if a point is inside a d -dimensional rectangle takes $2d$ comparisons while checking if a point is on either side of an axis aligned hyperplane is only one comparison. In Figure 5-2 you see how the two algorithms partition data differently.

The bbd-tree has the same asymptotic running times as the kd-tree, but they work differently in different situations. In the experiments we performed we actually got better performance using the kd-tree implementation. In their paper [1] they show that for certain distribution, where the data is clustered, the bbd-tree can perform better, especially when running approximate nearest neighbor.

5.3.3 Results

To test the different algorithms we created a synthetic dataset. This was mainly because the real datasets available were of much smaller scale then we were interested in (only around few thousands scans). A library called Approximate Nearest

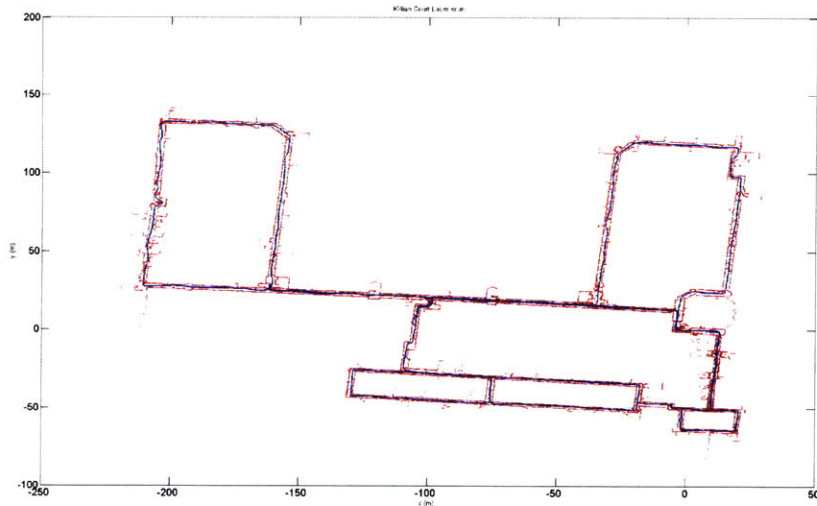


Figure 5-3: Killian court dataset

Neighbor [74] was used for these experiments. It implements both algorithms, and it supports both k-nn and fixed radius k-nn. It also has an option of giving an approximate solution. The approximate algorithm was not used, even though it might be interesting to explore how well approximate solutions might do in this application.

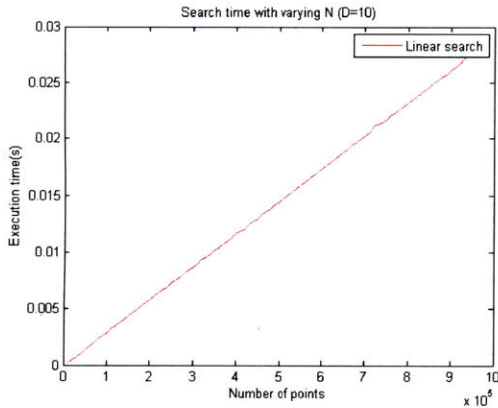
We tested the system with up to one million data points. For comparison, a complete laser scan map that covers one floor of Killian Court [11] (see Figure 5-3) is only 1900 scans. First, N similar feature vectors were sampled, then using those as means a few similar places were created, and then revisits to those places. The model can be described in the following way

$$\mathbf{f} = \mathbf{x} + \mathbf{u} + \mathbf{v} \quad \mathbf{f} \in \mathbf{R}^2$$

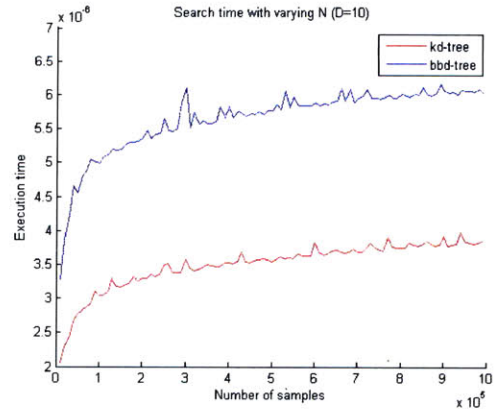
and

$$\mathbf{x} = U(0, r) \quad \mathbf{u} = N(0, \sigma_u) \quad \mathbf{v} = N(0, \sigma_v)$$

where U and N are the uniform and normal distribution respectively. The \mathbf{x} are the similar places, \mathbf{u} are the variation of places that are similar and \mathbf{v} are the revisits to a the place.



(a) Search time for the linear algorithm, execution time is linear with number of samples.

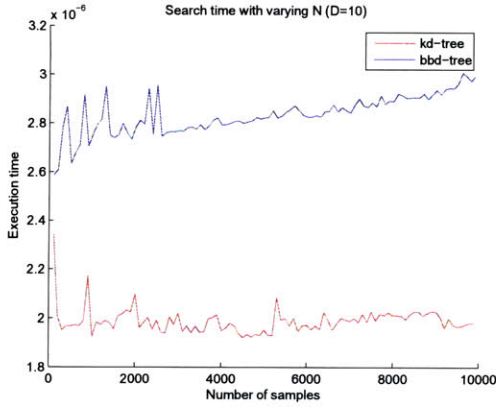


(b) Search time for the two tree algorithms

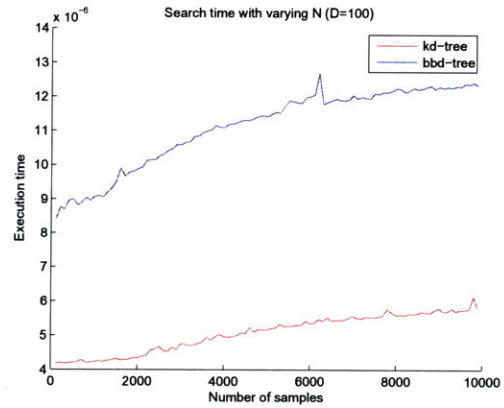
Figure 5-4: This figure shows search time for the different algorithms as function of number of samples. Execution time is logarithmic with the number of samples. The data set was fixed at dimension 10.

The measurement of query time was done by performing 1000-10000 queries for each data point, which was picked uniformly at random from the collection of points in the map. Both the tree structures gave very good performance and can be seen in Figures 5-4, 5-5 and 5-6. We tested for different number of points and looked at using different dimensions to see how that would affect the results. One thing observed was that when using the bbd-tree for data with dimension higher than 80, it started to use up a lot of memory, we did not look further into that issue but dropped those data points because the computer started thrashing the memory and the comparison would not have been useful.

The effect of different distributions of the data points was compared, by using $r = 1000$ and $r = 10$. That had a big effect on the results, and can be seen clearly by comparing Figures 5-4 and Figure 5-6. When the data is more dense then the tree algorithms will need to consider more cells than when the data is well spread out. When the data is spread out, it quickly goes to the correct cluster, in our situation each cluster had only around 10 points. This also agrees with what is known in machine learning, problems with large margins are usually easier to solve.

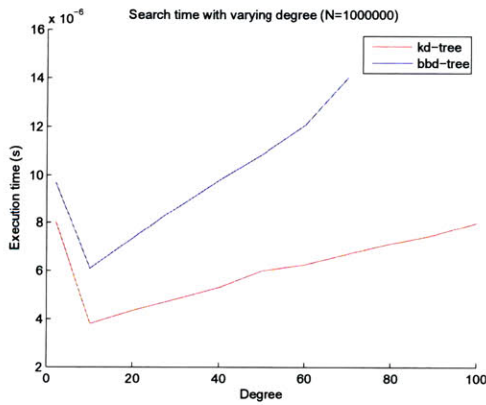


(a) Search time for the two tree algorithms with $d=10$.

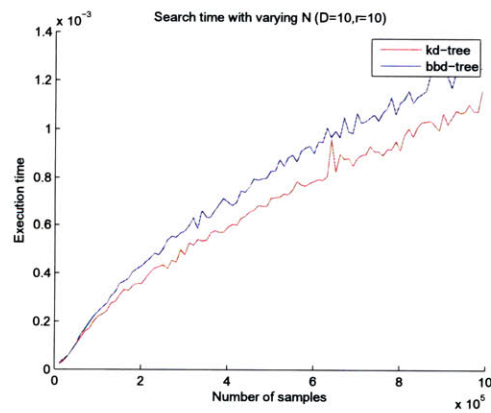


(b) Search time for the two tree algorithms with $d=100$.

Figure 5-5: Here are smaller data sets showing how execution varies with time. The data in the first graph has dimension 10 while the second graph has dimension 100.



(a) This figures shows how execution time grows with increasing dimension of the data.



(b) Here the data points are packed more closely together. The points were sampled from $U(0,10)$ instead of $U(0,1000)$ for all the other tests.

Figure 5-6: Looking at execution time for different dimensions and denser data

5.4 Customized search in database systems

In previous sections we looked at methods to first encode different views and then efficient algorithms to search in the database. Let us now look at how this might fit into a database system.

If all the place description are loaded into the database, using the following table

```
places(placeid,x,y,th,f1,f2,...fn)
```

Then using the standard indexes, hash or b+tree, that databases provides, it would most likely end up performing a linear scan through the table to find places that are at a certain distance from the current view.

PostgreSQL supports R-trees and spatial operators but that is mainly for 2d data. But PostgreSQL supports user defined access methods called Generalized Search Tree (GiST). A good overview of GiST can be found in [43].

In the latest version of PostgreSQL, the R-tree is actually implemented using GiST. To implement a new access method few functions have to be implemented: `consistent(E,q)`, `union(P)`, `penalty(E1, E2)` and `PickSplit(P)`. Where E is an entry (p,ptr), q is a query predicate and P is a set of entries.

`Consistent` returns false if $p \wedge q$ is unsatisfiable, for instance for a kd-tree if the entry corresponds to one side of the split plane. Then if q is in it it would return true, and false otherwise. `Union` gives a predicate that holds for a set of elements, `penalty` is used to help with creating a balanced tree, this can include the number of nodes. Finally `picksplit(P)` splits up the entries in P.

GiST looks like a very promising tool to use to extend the database to support efficient nearest neighbor queries and would be key to move the map application to a standard database management system (DBMS).

5.5 Summary

This chapter described a method to encode individual places in the map so it might be efficiently looked up at later point in time. It was shown that the kd-tree and

bbd-tree allow for fast searches for similar places. One thing to note is that, in terms of performance, the distribution of the feature vectors is important. This also affects the accuracy of the matches. In the next chapter we will review the contribution of this thesis and discuss what improvements and future directions might be of interest.

Chapter 6

Conclusion

This thesis has described an online navigation system that can keep an underwater vehicle localized over an extended period of time. This was achieved by using imagery from an inspection sonar to concurrently build a map of the environment and localize the vehicle. The performance of the system was verified in several experiments in which the system ran in real-time. The improvement in navigation accuracy was shown compared to dead reckoning using the DVL and gyro. In addition, an architecture for long term operations was designed and key components implemented.

6.1 Summary of contribution

In this thesis we have explored several problems related to long term autonomous navigation and proposed solutions to some of the issues in the context of underwater harbor surveillance. Many of the techniques are general enough that they can also apply outside of the underwater domain. The main contributions this thesis makes are:

1. A sonar-aided navigation system that enables drift-free operations in an unknown underwater environment, without need for any additional infrastructure, by using only existing natural features in the environment. This was achieved by developing a dense feature extraction algorithm, that was then used to compute pairwise registrations. Registrations between frames were computed using the

NDT algorithm. The benefits of using dense feature extraction is that it allows for accurate registration in a wide range of environments. The operation of the system was demonstrated in several experiments, where the system operated in real-time.

2. An architecture to support repeated and long term operations. We stated the requirements crucial for such a system. The key components were durability, flexibility, robustness, multi-session and multi-robot support. These requirements we believe pave the road for a system that can support robots for long term operations. We claim that such a system should be supportive of development tasks, and not be restricted to a specific kind of robot architecture. Following the requirements, a design of the system was presented. Parts of that system were implemented and potential usage scenarios demonstrated.

6.2 Future work

This thesis has made contribution in sonar-aided underwater navigation and long term operations. As with any work there is always room for improvement. Below we will discuss several important topics needing further research.

6.2.1 Sonar-aided navigation

The robustness of the current method can be improved by handling a wider range of environments, and to be able to detect and recover from erroneous registrations. Extending the registration method to handle more complex geometry is required to navigate on complex structures. Testing the methods developed on larger ships would be of interest; for example successfully inspecting a large ship like in Figure 6-1 would be a worthwhile goal. There is ongoing work to use vision methods for inspection and improved navigation using the same vehicle [54]. Negahdaripour et al. [76] worked on using an optical camera and an acoustic camera for opto-acoustic stereo. Further work in fusing those two sensors is of interest.



Figure 6-1: A liquefied natural gas carrier slowly making its way through Boston harbor. These ships are between 300 and 400 meters in length and autonomous inspection of a ship of this size would be an impressive feat.

It would be useful to run experiments using an independent measurement system to obtain ground truth data for the vehicle trajectory. This could be done by placing objects at known locations using differential GPS. Another possibility is to operate in an area where an acoustic tracking system can be deployed; then ground truth along the full vehicle trajectory could be acquired.

Robustly dealing with mistakes in the estimation process is still an open problem in robotics. Can we detect when a wrong constraint gets added and, more importantly, can we recover from it? Can we learn what features in the environment are useful for registration? The main reason for these issues is that, in general, solving the data association problem is a challenging task. Re-localization is also an important issue. One such method was considered in this thesis. There has been promising progress made in this area using appearance based methods on optical images [17]. Using similar methods by introducing an acoustic vocabulary to describe places in the map could be a viable approach.

Alternatives to wide field of view imaging sonar are the multi-beam sonars or

profiling sonars. On the DIDSON it is possible to use an acoustic lens to narrow the vertical field of view down to 1 degree. This allows the sonar to be used more like a laser range finder. In combination with the low drift of the DVL and the gyro in the short term, a set of submaps can be created and then registered with each other to derive constraints for the map estimation. This has been done by Roman et al. [86] with multi-beam sonars for improved bathymetry mapping and would be interesting to apply to the harbor inspection task.

6.2.2 Long term navigation

There are several improvements to the MapDB that might be interesting to explore. In our work we evaluated an existing relational database, PostgreSQL. We are interested in exploring further some of the extensibility mechanisms in that database and how they can be used to better support the application of localization and mapping. It might also be interesting to consider a custom database architecture, i.e to design the storage and access methods for the mapping application. It has been shown in other areas that dramatic improvements can be achieved by designing the database systems for specific tasks. An example of this is the C-Store [90], which organizes the data by columns among other things; this improves the performance for data warehouse applications. Other examples include the H-Store for transactional system and Aurora for stream processing. A recent initiative is the SciDB [91], an open source DBMS for scientific research. The challenge they are trying to address, is how to store and give access to extremely large datasets. The expected usage patterns have specific characteristics like write once, read many.

The large online websites like Google have also developed their database system tailored to specific needs. In Google's case scalability is one of the big issues. They have developed a system called BigTable [15] to address these challenges. BigTable is a distributed storage system and has been used by applications such as web indexing and Google Earth. It is designed for petabytes of data and can scale across thousands of servers. The data model consists of tables that are sparse, distributed, persistent multidimensional sorted maps. To achieve this scalability, BigTable relaxes support

for multi row atomicity that is standard in transactional systems.

For continuous operations it might not be feasible to store complete datasets for the full duration. The amount of data recorded by a single color camera, running at 30 frames per second, at moderate resolution over one year period would be on the order of 900TB, uncompressed. Compressing the still images might be an option, and for a single session use video compression to archive this data. Other lossy compression schemes might also be considered such as only retaining a limited number of keyframes, and only if changes are detected. Developing an informative filtering and archival process is likely a critical component for any map database system. Video compresses well because it is usually a sequence of frames that are locally very similar. Using this idea, grouping data spatially or by some similarity metric might allow for a more efficient compression or storage in the database. For example if the system is maintaining a map of sonar images and some area has not changed in any meaningful way, then there is no need to store both views. On the other hand, if the view has changed, it might be sufficient to store only the parts that are different.

When operating over an extended period of time, including the full trajectory in the estimation process is not feasible. Repeated passes through an environment that has already been mapped will cause the estimate problem to become less sparse, losing the efficiency benefits gained from the sparsity. This implies that the history needs to be summarized. One option is to provide a view of a current map that the vehicle uses to localize while simultaneously building a new map of the current session. When the session ends, it could be migrated into the current map.

Bibliography

- [1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.
- [2] A. Bachrach. Autonomous flight in unstructured and unknown indoor environments. Master’s thesis, Massachusetts Institute of Technology, Sept 2009.
- [3] A. Bachrach, R. He, and N. Roy. Autonomous flight in unstructured and unknown indoor environments. In *European Micro Aerial Vehicle Conference*, Sep 2009.
- [4] T. Bailey and H.F. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part II. *Robotics & Automation Magazine*, 13(3):108–117, Sep 2006.
- [5] E. Belcher, W. Hanot, and J. Burch. Dual-frequency identification sonar (DIDSON). In *Underwater Technology, 2002. Proceedings of the 2002 International Symposium on*, pages 187–192, 2002.
- [6] P. Biber and T. Duckett. Experimental analysis of sample-based maps for long-term SLAM. *Intl. J. of Robotics Research*, 28:20–33, 2009.
- [7] P. Biber and W. Strasser. The normal distributions transform: a new approach to laser scan matching. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2743–2748, Oct 2003.
- [8] M. Bosse, P. Newman, J. Leonard, and S. Teller. An atlas framework for scalable mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1899–1906, 2003.
- [9] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *Intl. J. of Robotics Research*, 23(12):1113–1139, Dec 2004.
- [10] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based SLAM. *Intl. J. of Robotics Research*, 27:667–691, 2008.
- [11] M.C. Bosse. *ATLAS: A framework for large scale automated mapping and localization*. PhD thesis, Massachusetts Institute of Technology, 2004.

- [12] N. Bowditch. *American practical navigator: an epitome of navigation and nautical astronomy*. United States Hydrographic Office, 1920.
- [13] R.F. Cahill and E. Udd. Phase-nulling fiber-optic laser gyro. *Optics Letters*, 4:93–95, 1979.
- [14] R.N. Carpenter. Concurrent mapping and localization with FLS. In *Autonomous Underwater Vehicles, 1998. AUV'98. Proceedings Of The 1998 Workshop on*, pages 133–148, 20-21 1998.
- [15] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):1–26, 2008.
- [16] P.A. Collinder. *A history of marine navigation*. St. Martin's press, 1955.
- [17] M. Cummins and P. Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Robotics: Science and Systems (RSS)*, Seattle, USA, Jun 2009.
- [18] F. Dellaert. Square Root SAM: Simultaneous location and mapping via square root information smoothing. In *Robotics: Science and Systems (RSS)*, 2005.
- [19] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.
- [20] H.F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, RA-4(1):23–31, Feb 1988.
- [21] H.F. Durrant-Whyte and T. Bailey. Simultaneous localisation and mapping (SLAM): Part I. *Robotics & Automation Magazine*, 13(2):99–110, Jun 2006.
- [22] B. Englot, H. Johannsson, and F. Hover. Perception, stability analysis, and motion planning for autonomous ship hull inspection. In *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology (UUST)*, 2009.
- [23] R. Eustice. *Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, Jun 2005.
- [24] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS titanic with SLAM information filters. In *Robotics: Science and Systems (RSS)*, Jun 2005.
- [25] R. Eustice, L. Whitcomb, H. Singh, and M. Grund. Recent advances in synchronous-clock one-way-travel-time acoustic navigation. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, pages 1–6, Boston, MA, USA, 2006.

- [26] R.M. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Trans. Robotics*, 22(6):1100–1114, Dec 2006.
- [27] N. Fairfield, A. G. Kantor, and Wettergreen D. Real-time SLAM with octree evidence grids for exploration in underwater tunnels. *J. of Field Robotics*, 2007.
- [28] N. Fairfield and D. Wettergreen. Active localization on the ocean floor with multibeam sonar. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, pages 1–10, 15-18 2008.
- [29] J. Favill. *Primer of celestial navigation*. Cornell maritime press, New York, 1940.
- [30] J.P. Fish and H.A. Carr. *Sound underwater images: a guide to the generation and interpretation of side scan sonar data*. Lower Cape Pub Co, 1990.
- [31] J. Folkesson and H.I. Christensen. Graphical SLAM - a self-correcting map. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 383–390, 2004.
- [32] J. Folkesson, J. Leederkerken, R. Williams, A. Patrikalakis, and J. Leonard. A feature based navigation system for an autonomous underwater robot. In *Field and Service Robotics (FSR)*, volume 42, pages 105–114, 2008.
- [33] T.I. Fossen. *Guidance and control of ocean vehicles*. John Wiley & Sons Inc, 1994.
- [34] J.H. Freidman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.
- [35] U. Frese. Efficient 6-DOF SLAM with Treemap as a generic backend. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4814–4819, 10-14 2007.
- [36] A. Gelb. *Applied optimal estimation*. MIT press, 1974.
- [37] N.R.E. Gracias. *Mosaic-based Visual Navigation for Autonomous Underwater Vehicles*. PhD thesis, Instituto Superior Técnico, 2002.
- [38] M.S. Grewal, L.R. Weill, L.R. Weill, and A.P. Andrews. *Global positioning systems, inertial navigation, and integration*. Wiley-Blackwell, 2007.
- [39] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, May 2010.
- [40] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems (RSS)*, Jun 2007.

- [41] A. Gurney. *Compass: A Story of Exploration and Innovation*. WW Norton & Company, 2004.
- [42] S.E. Harris and E.V. Slate. Lamp Ray: ship hull assessment for value, safety and readiness. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 493–500 vol.1, 1999.
- [43] J.M. Hellerstein, J.F. Naughton, and A. Pfeffer. Generalized search trees for database systems. *Readings in database systems*, page 101, 1998.
- [44] JB Hewson. *A history of the practice of navigation*. Brown, Son & Ferguson, 1951.
- [45] A. Huang, E. Olson, and D. Moore. Lightweight communications and marshalling for low-latency interprocess communication. Technical Report MIT-CSAIL-TR-2009-041, Computer Science and Artificial Intelligence Laboratory, MIT, Sep 2009.
- [46] RD Instruments. Acoustic doppler current profiler: Principles of operation a practical primer. Technical report, RD Instruments, San Diego, CA, USA, 1996.
- [47] B. Jalving, K. Gade, O.K. Hagen, and K. Vestgard. A toolbox of aiding techniques for the HUGIN AUV integrated inertial navigation system. *Modeling, identification and control*, 25(3):173–190, 2004.
- [48] A. Johnson and M. Hebert. Refinement of seafloor elevation using acoustic backscatter. Technical Report CMU-RI-TR-95-01, Robotics Institute, Pittsburgh, PA, March 1995.
- [49] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238 – 4243 vol.4, 2001.
- [50] S.J. Julier, J.K. Uhlmann, I. Ind, and MO Jefferson City. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.
- [51] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Journal of Robotics and Autonomous Systems*, 57:1198–1210, Dec 2009.
- [52] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The Bayes tree: Enabling incremental reordering and fluid relinearization for online mapping. Technical Report MIT-CSAIL-TR-2010-021, Computer Science and Artificial Intelligence Laboratory, MIT, Jan 2010.
- [53] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, Dec 2008.

- [54] A. Kim and R. Eustice. Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2009.
- [55] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple relative pose graphs for robust cooperative mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3185–3192, Anchorage, Alaska, May 2010.
- [56] W.J. Kirkwood and D.W. Caress. Comparison of MBARI autonomous underwater mapping results for ORION Monterey accelerated research system (MARS) and Neptune Canada. In *Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2007. Symposium on*, pages 13–20, 17-20 2007.
- [57] K. Konolige. Large-scale map-making. In *AAAI'04: Proceedings of the 19th national conference on Artificial intelligence*, pages 457–463. AAAI Press / The MIT Press, 2004.
- [58] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Trans. Robotics*, 24(5):1066–1077, Oct 2008.
- [59] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent. Centibots: Very large scale distributed robotic teams. *Experimental Robotics IX*, 21:131–140, 2006.
- [60] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, et al. A perception-driven autonomous urban vehicle. *J. of Field Robotics*, 25(10):727–774, 2008.
- [61] J. Leonard and P. Newman. Consistent, convergent, and constant-time SLAM. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1143–1150. Morgan Kaufmann Publishers Inc., 2003.
- [62] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, pages 1442–1447, Osaka, Japan, 1991.
- [63] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 333–349, Apr 1997.
- [64] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, pages 249–275, Apr 1997.
- [65] B. Jalving M. Mandt, K. Gade. Integrating DGPS-USBL positioning measurements with inertial navigation in the HUGIN 3000 AUV. In *Saint Petersburg International Conference on Integrated Navigation Systems*, Russia, May 2001.

- [66] M. Magnusson, H. Andreasson, A. Nuechter, and A. J. Lilienthal. Appearance-based loop detection from 3D laser data using the normal distributions transform. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 23–28, May 2009.
- [67] I. Mahon, S.B. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based SLAM using visual loop closures. *IEEE Trans. Robotics*, 24(5):1002–1014, Oct 2008.
- [68] A. Mallios, P. Ridaou, E. Hernandez, D. Ribas, F. Maurelli, and Y. Petillot. Pose-based SLAM with probabilistic scan matching algorithm using a mechanical scanned imaging sonar. In *OCEANS 2009-EUROPE, 2009. OCEANS '09.*, pages 1 –6, May 2009.
- [69] H. Medwin and C. S. Clay. *Fundamentals of Acoustical Oceanography*. Academic Press, San Diego, CA, USA, 1998.
- [70] M.J. Milford and G.F. Wyeth. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Trans. Robotics*, 24(5):1038–1053, Oct 2008.
- [71] M.J. Milford and G.F. Wyeth. Persistent navigation and mapping using a biologically inspired SLAM system. *Intl. J. of Robotics Research*, 2009.
- [72] D.A. Mindell. Precision navigation and remote sensing for underwater archaeology. *Remote sensing in archaeology*, page 499, 2007.
- [73] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1151–1156. Morgan Kaufmann Publishers Inc., 2003.
- [74] D.M. Mount. ANN programming manual, 1998.
- [75] S. Negahdaripour, P. Firoozfam, and P. Sabzmeydani. On processing and registration of forward-scan acoustic video imagery. In *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, pages 452–459, 2005.
- [76] S. Negahdaripour, H. Sekkati, and H. Pirsivash. Opti-acoustic stereo imaging: On system calibration and 3-D target reconstruction. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 2009.
- [77] P. Newman and J. Leonard. Pure range-only sub-sea slam. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1921 – 1926 vol.2, 14-19 2003.

- [78] P.M. Newman, J.J. Leonard, and R.J. Rikoski. Towards constant-time SLAM on an autonomous underwater vehicle using synthetic aperture sonar. *Robotics Research*, pages 409–420, 2003.
- [79] K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact, out-of-core, submap-based SLAM. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1678–1685, Apr 2007.
- [80] I.R. Nourbakhsh, C. Kunz, and T. Willeke. The mobot museum robot installations: a five year experiment. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, volume 4, pages 3636 – 3641 vol.3, 2003.
- [81] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2262–2269, May 2006.
- [82] E. Olson, J. Leonard, and S. Teller. Spatially-adaptive learning rates for online incremental SLAM. In *Robotics: Science and Systems (RSS)*, Jun 2007.
- [83] M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1157–1164, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.
- [84] D. Ribas, P. Ridao, J. Neira, and J.D. Tardós. SLAM using an imaging sonar for partially structured underwater environments. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.
- [85] D. Ribas, P. Ridao, J.D. Tardós, and J. Neira. Underwater SLAM in man-made structured environments. *Journal of Field Robotics*, 25(11-12):898–921, 2008.
- [86] C. Roman and H. Singh. Improved vehicle based multibeam bathymetry using sub-maps and SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3662–3669, Aug 2005.
- [87] H. Sekkati and S. Negahdaripour. 3-D motion estimation for positioning from 2-D acoustic video imagery. *Lecture Notes in Computer Science*, 4478:80, 2007.
- [88] G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *Intl. J. of Robotics Research*, page 0278364910369268, 2010.
- [89] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.
- [90] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O’Neil, P. O’Neil, A. Rasin, N. Tran, and S. Zdonik. C-store: a column-oriented DBMS. In *VLDB '05: Proceedings*

- of the 31st international conference on Very large data bases, pages 553–564. VLDB Endowment, 2005.
- [91] M. Stonebraker, J. Becla, D. Dewitt, K.T. Lim, D. Maier, O. Ratzeberger, and S. Zdonik. Requirements for science data bases and SciDB. *CIDR 2009*, 2009.
- [92] M. Talwani. Developments in navigation and measurement of gravity at sea. *Geoexploration*, 8(3-4):151 – 183, 1970.
- [93] I. Tena Ruiz, S. de Raucourt, Y. Petillot, and D.M. Lane. Concurrent mapping and localization using sidescan sonar. *Journal of Oceanic Engineering*, 29(2):442 – 456, april 2004.
- [94] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, AB Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Intl. J. of Robotics Research*, 19(11):972–999, 2000.
- [95] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.
- [96] S. Thrun and Y. Liu. Simultaneous localization and mapping with sparse extended information filters. *Intl. J. of Robotics Research*, 23(7), 2004.
- [97] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 4(3):380–407, 2004.
- [98] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, page 372. Springer-Verlag, 1999.
- [99] J. Vaganay, M. Elkins, D. Esposito, W. O’Halloran, F. Hover, and M. Kokko. Ship hull inspection with the HAUV: US Navy and NATO demonstrations results. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, volume 1, pages 761–766, 2007.
- [100] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *SenSys ’05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 154–165, New York, NY, USA, 2005. ACM.
- [101] T. Vilhjalmsón. Time and travel in old norse society. *Disputatio, II*, pages 89 – 114, 1997.

- [102] M. Walter, F. Hover, and J. Leonard. SLAM for ship hull inspection using exactly sparse extended information filters. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1463–1470, 2008.
- [103] M.R. Walter. *Sparse Bayesian information filters for localization and mapping*. PhD thesis, Massachusetts Institute of Technology, 2008.
- [104] L. Whitcomb, D. Yoerger, and H. Singh. Advances in doppler-based navigation of underwater robotic vehicles. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 399–406 vol.1, 1999.
- [105] L. Whitcomb, D. Yoerger, and H. Singh. Combined Doppler/LBL based navigation of underwater vehicles. In *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology (UUST)*, May 1999.
- [106] L. Whitcomb, D. Yoerger, H. Singh, and D. Mindell. Towards precision robotic maneuvering, survey, and manipulation in unstructured undersea environments. In *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, volume 8, pages 45–54, 1998.
- [107] D.R. Yoerger and D.A. Mindell. Precise navigation and control of an ROV at 2200 meters depth. In *Proceedings of Intervention/ROV*, volume 92, 1992.