LOCALIZATION AND MAPPING FROM SHORE CONTOURS AND DEPTH

ROBERT CODD-DOWNEY

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO
JANUARY, 2017

©Robert Codd-Downey, 2017

Abstract

Building a representation of space and estimating a robot's location within that space is a fundamental task in robotics known as simultaneous localization and mapping (SLAM). This work examines the problem of solving SLAM in aquatic environments using an unmanned surface vessel under conditions that restrict global knowledge of the robot's pose. These conditions refer specifically to the absence of a global positioning system to estimate position, a poor vehicle motion model, and the lack of a strong stable magnetic field to estimate absolute heading. These conditions can be found in terrestrial environments where the line of sight to GPS satellites is occluded by surrounding structures and local magnetic inference affects reliable compass measurements. Similar conditions are anticipated in extra-terrestrial environments such as on Titan where the lack of a global satellite network inhibits the use of traditional positioning sensors and the lack of a stable magnetic core limits the applicability of a compass. This work develops a solution to the SLAM problem that utilizes shore features coupled with information about the depth of the water column. The approach is validated experimentally using an autonomous surface vehicle utilizing omnidirectional video and a depth sounder. Solutions are compared to ground truth obtained using GPS and to solutions found when the restriction of a poor magnetic field is lifted.

Acknowledgements

First and foremost I would like to acknowledge the advice and support provided by my supervisor Dr. Michael Jenkin. I would be remiss without mentioning the contribution of fellow lab mates who assisted me during my research, namely Monica, Vivek, Shreyansh and Masoud. I would also like to thank my parents Bob and Judi as well as my sister Beckie for their assurance and help in all aspects of my studies. Last but certainly not least I want to thanks my dog Milo for always being by my side, never faltering in his steadfast resolve.

Table of Contents

List of Figures

List of Abbreviations

6DOF Six Degrees of Freedom

ASV Autonomous Surface Vessel

DOF Degrees of Freedom

DHCP Dynamic Host Configuration Protocol

EKF Extended Kalman Filter

EMI Electromagnetic Interference

FPS Frames per Second

GPS Global Positioning System

LAN Local Area Network

LiDAR Light Detection and Ranging

IP Internet Protocol

MCPTAM Multi-Camera Parallel Tracking and Mapping

NMEA National Marine Electronics Association

PTAM Parallel Tracking and Mapping

REHCOS ROS Echo Sounder

ROS Robot Operating System

RGB-D Red Green Blue Depth

RMS Root Mean Square

SLAM Simultaneous Localization and Mapping

SIR Sample Importance Resample

SONAR Sound Navigation and Ranging

UAV Unmanned Aerial Vehicle

USV Unmanned Surface Vessel

WLAN Wireless Local Area Network

List of Symbols

- $\mathbf{x}_t$: state vector describing the robot's location and orientation at time t.
- $\mathbf{u}_t$: control vector describing the robot's motion from time $t-1$ to t.
- $\mathbf{z}_t$: sensor measurement observed from the robot at time t.
- $\mathbf{m}_i$: vector describing the location of i-th landmark.
- $\mathbf{R}_t$: covariance describing uncertainty with a sensor measurement at time t.
- $\mathbf{P}_t$: covariance describing uncertainty with the robot's motion at time t.
- $\mathbf{X}_{0:t} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t)$: sequence of vehicle states.
- $\mathbf{U}_{0:t} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_t)$: sequence of vehicle controls.
- $\mathbf{Z}_{0:t} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_t)$: sequence of sensor measurements.
- $\mathbf{M} = \{\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_n\}$: the set of all landmarks.
- $\mathbf{Y}_{0:t} = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_t)$: the sequence of sets of particles (shows particle evolution over time).

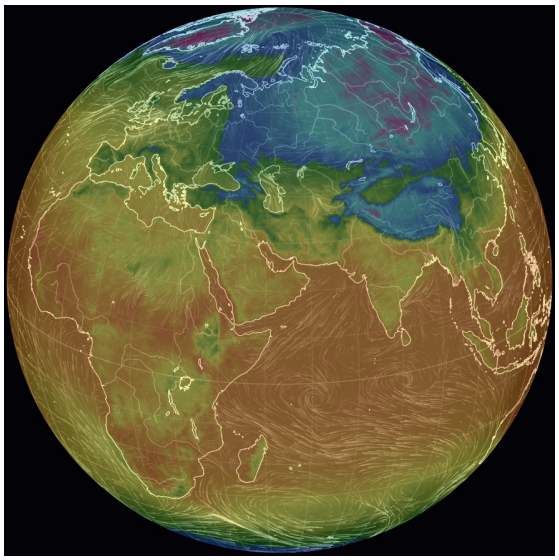
- $\mathbf{Y}_t = \{\mathbf{Y}_{0,t}, \mathbf{Y}_{1,t}, \ldots, \mathbf{Y}_{k,t}\}$: the set of particles at time t.
- $\mathbf{y}_t^k = \{\mathbf{x}_t^k, \mathbf{M}^k\}$: the k-th particle at time t, containing the current estimated state $\mathbf{x}_t^k$ and set of hypothesized landmarks $\mathbf{M}^k$.
- $\mathbf{D}_{\mathbf{z}_{l,t}}$: quantitative description of the l-th measurement observed at time t.
- $\mathbf{D}_{\mathbf{m}_i^k}$: quantitative description of the i-th landmarks.

Chapter 1

Introduction

The ability of a robot to build an accurate map of its environment and to localize itself within that map is an important step in creating fully autonomous robotic agents. In the literature and academic community this task is commonly referred to as the simultaneous localization and mapping (SLAM) problem [1]. For sufficiently well behaved environments, sensors, and vehicles SLAM is generally considered solved [2]. Standard implementations of SLAM exist for numerous structured domains and sensors; however, many aspects of the SLAM problem remain unexplored or unsolved for more complex environments, vehicle models and sensors. Moving forward, the challenge for SLAM algorithms and their implementation is on expanding the capabilities of existing approaches to cope with larger and more unstructured environments. Many difficult environments exist for SLAM algorithms. Demonstrating the consistent ability to navigate vast outdoor environments without aid from a global position estimation system such as a global positioning system (GPS) [3] is a problem of particular interest. Terrestrial environments such as forests, mountainous regions, and urban canyons are challenging SLAM environments, and of course GPS is completely unavailable underground, underwater and for non-terrestrial exploration. In such *GPS denied* environments SLAM solutions

(a) *The Fra Mauro world map created in the mid 15th century showing Europe, Asia and Africa.*



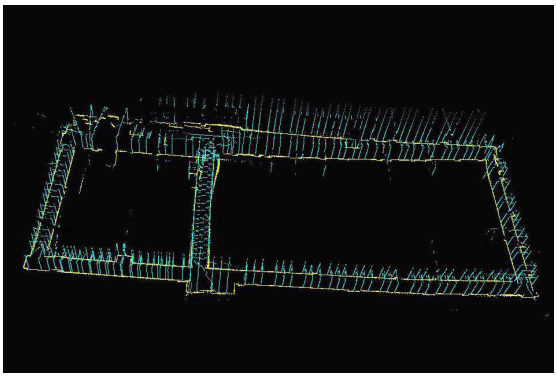
(b) *Modern day world map of Europe, Asia and Africa overlaid with temperature and wind.*
GFS / NCEP / US National Weather Service.

Figure 1.1: **Antiquated and modern world maps.**
A side by side comparison of the Fra Mauro map to a modern world map generated from satellite data.

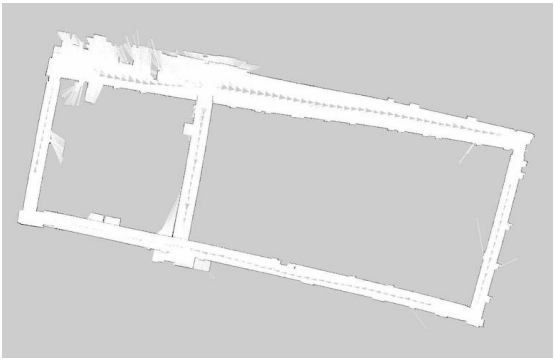
must rely on locally sensed landmarks and odometry models rather than relying on GPS signals.

Humans have used maps for centuries. Early human maps can be traced back to Babylon, Greece and Asia, and by the time of the Greek and Roman empires, maps were ubiquitous navigational aides. By the mid 15th century, humans had built maps that represented most of the known world. Figure 1.1a, for example, shows the Fra Mauro map, which depicts the known world – at least as known to Europeans – around 1450. Today, of course, we have access to more sophisticated sensing technologies and maps such as the one shown in Figure 1.1b which depicts weather information in a global reference frame are common.

Given the usefulness of maps for humans, it is perhaps not surprising then that maps can be equally useful for robots. In addition to providing a similar representation for spatial information and tasks for machines, maps can also provide a common framework for human-robot communication. Maps provide a workable definition of 'here' and 'there'. They provide a useful representation for information crucial for navigation –



(a) *3-dimensional point cloud of our office building at York University.*



(b) *2-dimensional occupancy grid of an office space.*

Figure 1.2: **Typical SLAM maps**

Examples of maps obtained with SLAM algorithms. (a) shows a 3D map of the ground floor of the Sherman Health Sciences building at York University obtained using a ground contact robot and LIDAR sensor. (b) shows a 2D map of the same space. In both cases the map is actually represented as a collection of sensor readings, here laser sensor measurements, embedded in a Cartesian space.

how to get from here to there. And they provide a useful framework within which events, locations, tasks and similar things can be embedded. Knowing the structure of the surrounding environment aids in navigation from one point to another within the environment. Maps that can be translated into a format that can be interpreted by a human operator and a machine can be used as the basis of communication between human and robot: In their simplest form such maps can be used to communicate location and trajectory through space. These basic maps can be annotated with additional information and used as a context for much more powerful and effective communication. Beyond providing a common framework for human-robot communication, maps provide an essential resource for a wide range of robot applications.

Maps are almost ubiquitous for humans in the present day. Terrestrial maps generated *a priori* are common in today's society with Google Maps [4], OpenStreetMap [5], and others providing nearly complete global coverage of the world. These massive cartographic undertakings focus on providing information to drivers and pedestrians travelling on public roadways and transit systems and build upon centuries of earlier work with paper-based maps. Other projects like OpenSeaMap [6] provide similarly relevant information to boat operators and commercial transport ships. The highest detailed civilian maps typically only include

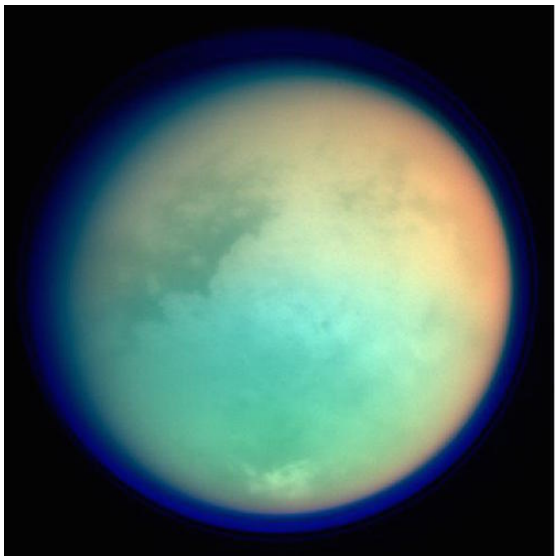
(a) *aerial photograph of a tailing pond.*

(b) *land-based view of a tailing pond.*

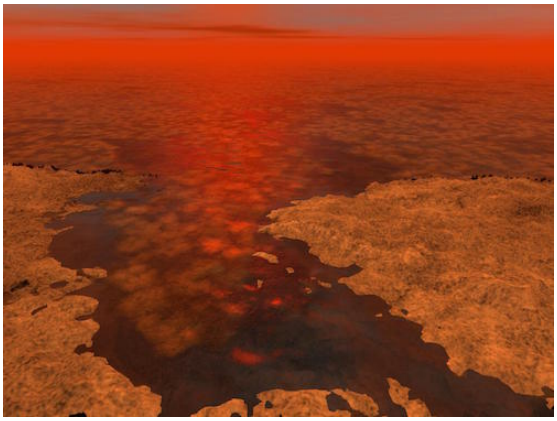
Figure 1.3: **Tailing pond**
Photographs of tailing ponds in Elko, Nevada.

topographical information at a resolution of tens of meters. Although these maps are very useful for a range of human activities these maps are mostly unsuited on their own for robotic navigation because they do not provide data that is crucial to robot operations. Knowing where the nearest coffee shop is on a map may be very useful for humans. It is less useful for today's robots. Similarly robot maps typically include representations of events – often sensor events – of particular interest to robots but of less utility for humans.

Terrestrial maps typically encode information about the ground. Although most robots operate on the ground plane, a range of interesting environments exist beyond this domain. To take but just one non-ground-plane application, consider the problem of navigating and representing the surface of bodies of water. Localization and mapping of bodies of liquids (primarily water terrestrially) finds a wide range of applications. Terrestrially, reservoir, lakes, ponds, rivers and the like provide a range of interesting environments for robot operations. Furthermore, many of these environments are GPS denied (e.g., underground reservoirs, surface water bodies in mountainous regions, etc.) preventing the use of a global localization solution. For example, Figure 6.1 shows a mining mail pond in Elko, Nevada. Understanding the current contents and scale of the water surface of such ponds is a critical day-to-day requirement of mining operation. Off-earth lakes, (not necessarily of water) on the surface of non-terrestrial planets and natural satellites are of particular importance to the future of space exploration. Titan, one of Saturn's moons, is the only known celestial body



(a) Multi spectral image of Titan. NASA/JPL/Space Science Institute.



(b) Artist rendering of a Titan lake. NASA/JPL-Caltech/USGS.

Figure 1.4: **Titan**

within our solar system with hydrocarbon lakes on its surface [7]. Other celestial bodies such as Europa and Enceladus may have interior bodies of water [8][9]; however, this makes them less accessible to exploration by an autonomous robot. Titan, shown in Figure 1.4 is of particular interest for space exploration because of its geological environment and its potential to support pre-biotic chemistry [10]. Titan's size and thus curvature presents additional problems for SLAM, being that even moderately sized lakes on Titan can have large sections in which lake shores are either not visible, or not discernible due to limitations on the resolution of standard cameras. Titan does not support a constant magnetosphere [11], limiting the usefulness of a compass in navigation and mapping. As a result, SLAM on but near the shore of the lakes of Titan must rely on local sensors, including visual sensors of shoreline features, a sun compass and information concerning the local depth of the "water" column. Terrestrial regions with poor compass performance also exist, with large metal bodies and electromagnetic fields generated by motors and the like typically lead to substantive failures in terms of compass performance. Although the work in this thesis is intended to be applicable to the SLAM task on bodies of liquid generally, one goal of this work is to investigate how the limits of Titan's sensing environment constrains SLAM algorithms operating in the environment found on Titan.

This research explores the viability of mapping bodies of water in a GPS denied and geo-magnetic absent environment using local measurements of environmental features, specifically shore/skyline features and the depth of the water column. Beyond the potential off-world application of such a solution to the SLAM problem under these circumstances facilitates autonomous navigation of terrestrial water bodies. Addressing the SLAM problem for an autonomous surface vehicle (ASV) operating in a GPS-denied environment involves developing appropriate sensor and locomotion models for such a device. This research utilizes a combination of sensors to address the sensing side of this problem. An omnidirectional camera is used to obtain visual information about the horizon structure and this information is integrated with information of the depth of the water column to estimate position/orientation information for the vehicle while simultaneously obtaining a map of the robot's environment. Experimental validation of the algorithm shows both the functionality of the basic approach, but also evaluates the importance of a compass and depth sensor in performing SLAM on the surface of bodies of water within sight of the shoreline.

1.1 What is SLAM and why is it hard?

Navigation without a map has been known to be a hard problem for a long time. Stories from ancient mythology and fairy tales such as "Theseus and the Minotaur" [12] and "Hansel and Gretel" [13] anecdotally demonstrate that it is easy to get lost without a map. In both of these stories the protagonists avoid getting lost by leaving a trail of breadcrumbs/string behind them to mark their path in order to find their way back. That is, they utilize landmarks – here artificial – to help solve the problem of building a representation of their space and navigating out of it. Given the difficulty of navigating without a map and the usefulness of maps for a variety of different tasks, humans have spent considerable effort in developing technologies to support map construction. The human task of building maps is known as cartography. Historically, cartography is a well studied and practiced art that relies heavily on mathematics. Over its long history new technological developments such as the compass, sextant and chronograph have allowed for the creation of more accurate maps by helping mapmakers to more accurately determine their location and the location of

features (landmarks). Some may point to old maps such as the Fra-Mauro map shown in Fig 1.1a and laugh at how 'poorly' they represent the environment by modern standards. This just helps to illustrate that map creation is a difficult task.

In general, robots do not have the luxury of being able to leave a trail of breadcrumbs to mark the path they have travelled. Although there are robot algorithms that solve SLAM in exactly this way, see [14], for example. If one could use breadcrumbs – and leaving aside the issue of other agents manipulating those markers – the map could be constructed incrementally by tracing the robot's position back through time. But what can an autonomous agent do if breadcrumbs are not available, or not appropriate for the specifics of the mapping task? In this case the agent might rely on an accurate odometry/motion model to solve the problem. If the robot's measured odometry was 100% reliable, map creation would be trivial. If the robot can remember exactly how far it moved, and exactly the order of these motions, then individual positions can be easily localized within some global coordinate system. The problem with this approach is that the motion of the robot is not 100% accurate and predictable. Every command motion of the agent is corrupted by noise and the motion error at each time step contributes to the error at the next time step. Furthermore, for many robots external forces acting on the vehicle can result in unintended un-commanded motion. This causes the estimate of the robots position to become less and less accurate over time, and eventually the agent's estimate of its current position is essentially meaningless. The most common source of odometry error for ground contact vehicles is wheel slippage, which happens when a wheel loses traction with the ground and turns without moving the robot. Aerial and aquatic vehicles are subject to a more severe form of odometry error caused by drag which is due to unknown motion of the environment (wind, wave, surge and the like).

Given the error associated with the robot's motion model it becomes necessary for the robot to rely on other sources of information to solve the SLAM problem. One option here would be to take advantage of naturally occurring landmarks or features in the environment to help solve the problem. Consider the case of a person (you) solving the problem. Although your estimate of your motion may be error prone,

if you encounter a unique landmark in the environment – a specific shop or building that is unique in the environment – then this landmark can anchor your representation of the space. Given enough such landmarks it then becomes possible to map the world even if the motion estimation process is noisy. This observation is the fundamental theory behind most modern SLAM algorithms. Rather than assuming unique features in space, a probabilisitic representation of these features is used enabling a probabilistic solution to both the mapping and localization problem.

1.2 Motivation

The motivation for the sensor restrictions considered in this work is to increase the generality of SLAM to mapping more eclectic environments. Although the specific restrictions considered here are motivated by the environment associated with Titan, these restrictions also have wide application in terrestrial environments – underground/indoor liquid bodies, and surface bodies for which GPS satellite visibility and a constant external magnetic field is not guaranteed. The terrestrial applications of this research are arguably more important given that missions to Titan [15] are still in the draft/proposal stage of development. The most interesting terrestrial environments to explore are underground rivers/lakes and reservoirs. For such environments there exists no line of sight to any GPS satellites, so they are by definition GPS denied environments. The presence of metallic deposits in the surrounding earth as well as ferrous material in the vessel itself – especially electric motors – can interfere with the local magnetic field rendering a compass useless. This work also asks the question, how important is a compass in terms of enabling SLAM to be solved? Access to a global directional signal can be very powerful, but how necessary is it in terms of solving SLAM?

1.3 Structure of this document

This thesis is organized into six chapters, this section is the culmination of the chapter entitled "Introduction". This chapter introduced the key aspects of the problems associated with this research as well as the motivations behind conducting this research. The second chapter entitled "Background" explores aspects of related fields of research with a primary focus on the simultaneous localization and mapping problem. The third chapter entitled "Eddy : an autonomous aquatic surface vessel" details the hardware and software tools that were leveraged in order to collect the data required to conduct this research. The fourth chapter entitled "Implementation" describes the mathematical and algorithmic framework that is used to integrate information from the robots sensors and measured odometery into an continuous estimate of its state and environment. The fifth chapter entitled "Experimental validation" describes the process of deploying the robot, collecting data and comparing the mapped environment against ground truth data. The final chapter entitled "Summary and future work" contains concluding remarks about the research and suggests possible avenues for further enhancements.

Chapter 2

Background

SLAM (Simultaneous Localization and Mapping) is a well studied problem which can be traced back to [16]. The modern Bayesian formulation of the problem finds its roots in [2]. Recent tutorial and survey papers (e.g., [1] and [17]) provide a good introduction to the problem although an abbreviated introduction to the problem is included in this chapter. A wide range of SLAM algorithms exist, although most have been developed/implemented with terrestrial ground-contact robots in mind. Such solutions typically assume a robot operating on a two dimensional ground plane having three degrees of freedom (x, y, θ), and a set of stationary features or landmarks that can be exploited to solve the SLAM problem. Well known implementations of SLAM such as GMAPPING[18], GridSlam[19] and DP-SLAM[20] are well suited to mapping large-scale indoor environments using laser range data (LiDAR). The maps generated by these algorithms typically come in the form of two dimensional occupancy grids depicting both the free space within the environments as well as its boundaries along with the localization of features (landmarks) that were used to build the representation. In concert with developing this map such algorithms estimate a sequence of robot waypoints within this map that are the hypothesized locations of the robot as the map was constructed. These im-

plementations struggle on rougher terrain when the assumption that the environment is well represented using a two dimensional grid is violated. In these situations more comprehensive approaches are necessary to properly map the surrounding environment. For 3D environments, RGB-D SLAM[21], SLAM6D[22] and PTAM[23] represent the environment as a three dimensional point cloud and track the robot's pose in 6 degrees-of-freedom (DOF).

Although terrestrial and aerial based visual SLAM algorithms can be effective, as is evidenced by their wide application both indoors and out, they are not ideal for the aquatic surface domain. The aquatic domain introduces two complexities for SLAM algorithms that are typically not encountered for ground contact or aerial robots. The first is that is the complex nature of the plant model for aquatic robots. For a terrestrial robot – especially wheeled terrestrial robots – the robot plant model is a particularly strong cue as to the relative motion of the robot. If the robot is not commanded to move, then it probably doesn't move, and if it is commanded to move a certain amount in a certain direction then it does with only a small and predictable error. A second issue for aquatic robots is the nature of the landmarks available to SLAM. The surface of the water is typically devoid of useful visual landmarks that can be used to aid mapping and localization. In general, visual landmarks are shore-based. This means that landmarks are typically not found in close proximity the robot nor are they distributed uniformly around the robot. Furthermore, the water surface itself acts as a reflector. The reflection of visual landmarks off the surface of the water is detrimental, causing the appearance of false landmarks below the water's surface. As the robot travels further away from shore the amount of visual real estate taken up by the sky and water increases. This presents a problem because overhead clouds and the geometry and reflection within waves can lead to the detection of false, non-static environmental features. Visual landmarks found within these regions need to be identified and removed so that they are not confused with real landmarks that can be measured repeatedly over time.

In the aerial domain, SLAM algorithms have also been developed for unmanned aerial vehicles (UAV). For such devices equipped with single camera a number of algorithms such as PTAM[23] have been developed for 6DOF pose estimation and environment mapping. MCPTAM[24][25] is an extension of PTAM to multiple

cameras that also supports cameras with a wide field of view. MCPTAM is well suited to UAVs equipped with any number of cameras. For UAV's, these approaches have many advantages over traditional ground-contact robot SLAM algorithms including real-time capabilities and efficiencies in terms of estimating pose in high dimensionality spaces. Furthermore they do not rely on a good motion model of the vehicle. However, in order to maintain accuracy they require an environment with a high number of stable visual features distributed over space. The limitation that most precludes the adoption of PTAM like algorithms is their inability to accommodate measurements from non-visual sensors.

SLAM has also been applied to autonomous underwater vehicles (AUV's). Underwater robots can use side scan SONAR to measure the distance to structures on the sea/ocean floor and this, coupled with an appropriate IMU, can be a very effective sensor combination. SLAM algorithms for the underwater domain need to be capable of fully estimating 6DOF motion and are usually designed to use SONAR rather than visual features due to particulate matter in the water column and refraction effects that obscure sight and reduce the availability of light. Although there are exceptions to this general rule, see [26] for example.

Although there is a large SLAM literature (see [1][17] for recent reviews of the field), the SLAM algorithms that are the most relevant to this research are those that operate on or map the surface of aquatic environments. In this domain there have been a number of efforts to solve the SLAM problem. For example, [27] describes the process of mapping large riverine environments with intermittent GPS data with an unmanned aerial vehicle (UAV). This system produces a mixed 2D and 3D map of the river using vision and LIDAR sensors, the 2D portion of the map depicts the extent of the river using an occupancy grid whereas the 3D map depicts objects above water level using a point cloud. The work in [28] describes an extension to the standard SLAM approach that models dynamic marine environments using RADAR. This approach tracks both stationary landmarks and moving objects using both as aids in the mapping process. This framework is quite desirable for applications where other aquatic vehicles are present or in the study of marine wildlife.

In the area of planetary exploration a number of marine robotics systems have been proposed (e.g., [29] and [30]). These systems include approaches that include autonomous aerial vehicles or geostationary satellites to aid in the process of localizing the robotic vehicle within the targeted environment.

Both the underwater and aerial domains share a commonality with the aquatic surface domain. Robots operating in these environments are subject external forces that can significantly alter their overall motion. This is in contrast with the operational domain of ground-contact robots. Ground-contact robots maintain a significant amount of friction between the robot and the ground which prevents all but severe weather conditions or ground tremors having any significant effect on their motion. For robots operating underwater or free-flying such un-commanded motion is a full 6DOF disturbance. Robots that operate on the surface of liquid bodies are not subject to fully 6DOF motion, unless capsized. The motion of a robotic boat can, in most circumstances be modelled using a 3DOF space using only its (x, y) position and θ orientation. Although it is important to note that the pose of a robot moving on a liquid surface is also subject to small variations in roll, pitch and heave as well.

The remainder of this chapter provides a formal definition of SLAM and a brief description of SLAM algorithms that have been applied to vehicles operating on the surface of, and underwater. This chapter also provides a review of the kinematics of differential drive vehicles and defines the coordinate frames used to describe the motion of a vehicle operating on the surface of a liquid.

2.1 Formal definition of SLAM

The formal definition and notation of the SLAM problem described here follows that presented in [1]. Consider a mobile robot beginning to explore an unknown environment starting from a known state $\mathbf{x}_0$, comprised of a point in Cartesian space and orientation. The robot is equipped with sensors to perceive its environment and some mechanism to control its motion within the environment. The effect of commanded motion of the vehicle is subject to noise, thus the robot's known position within the environment becomes

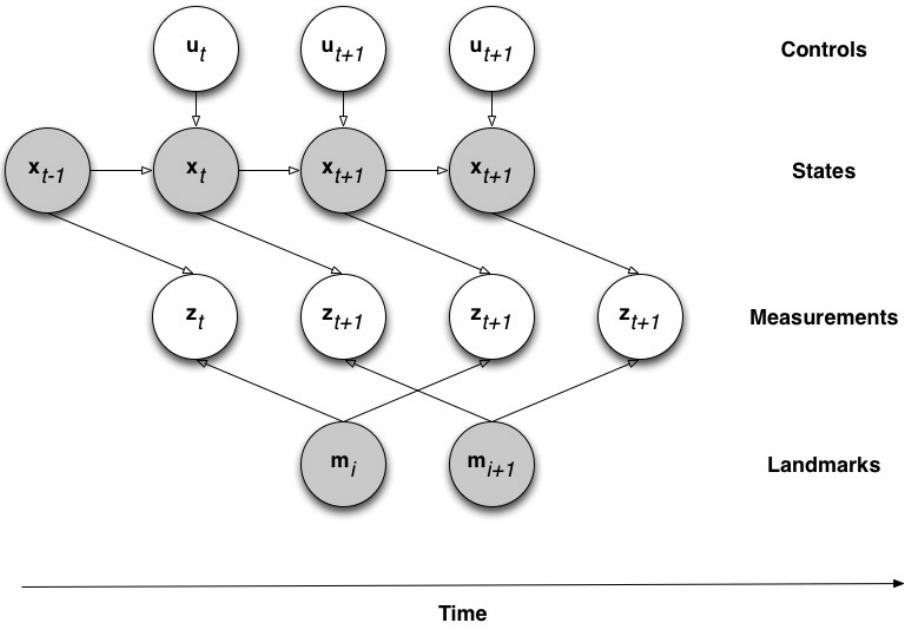


Figure 2.1: **Graphical representation of the SLAM problem**
At time t, the SLAM algorithm integrates state estimates with control inputs along with measurements of landmarks to update both the estimate of the robot's state and an estimate of a map of the environment. True measurements are shown in white, while estimated measures are shown in grey.

more uncertain as the robot moves. As the robot travels within the environment it takes measurements of its surroundings to localize recognizable and fixed landmarks repeatedly. These sensor measurements are also corrupted with noise. The goal of the SLAM problem is to establish, at each time t, the best representation of the world and a best estimation of the motion of the robot from time 0:t. The system has access to the commanded control inputs to the robot, an appropriate model of how the inputs impact the robot's state, and some set of measurements of visible landmarks. At time t the following variables are defined.

- $\mathbf{x}_t$: state vector describing the robot's location and orientation at time t.
- $\mathbf{u}_t$: control vector describing the robot's motion from time $t-1$ to t.
- $\mathbf{z}_t$: sensor measurement observed from the robot at time t.
- $\mathbf{m}_i$: vector describing the location of i-th landmark.

Additionally the following ordered and unordered collections are defined to the describe the experience of the robot throughout its exploration of the environment.

- $\mathbf{X}_{0:t} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t)$: sequence of vehicle states.
- $\mathbf{U}_{0:t} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_t)$: sequence of vehicle controls.
- $\mathbf{Z}_{0:t} = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_t)$: sequence of sensor measurements.
- $\mathbf{M} = \{\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_n\}$: the set of all landmarks.

Solving SLAM involves integrating measurements of the environment's features with commanded motion through the environment to estimate both robot state and the environment itself. Figure 2.1 represents the SLAM problem as a graph. Using this representation SLAM can be more easily viewed as a non-linear optimization process looking for the sequence of states $\mathbf{X}_{0:t}$ and set of landmarks $\mathbf{M}$ that best describe the controls $\mathbf{U}_{0:t}$, and measurements $\mathbf{Z}_{0:t}$ along with their associated error models.

2.2 SLAM within a Bayesian framework

Given the stochastic nature of measurements and motion estimates, a Bayesian probabilistic framework is typically adopted to solve the SLAM problem. Expressed in probabilistic form, the following probability distribution represents the essence of the SLAM problem.

$$p(\mathbf{x}_t, \mathbf{M} | \mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) \tag{2.1}$$

This is the joint probability of position and map, given the sequence of motion control signals, measurement inputs, and assuming some initial state. Solving the SLAM problem can be thought of as developing a representation of the posterior distribution and then collapsing this distribution to obtain an estimate

of the world $\mathbf{M}$ and the motion of the robot through it ($\mathbf{X}_{0:t}$). The posterior $p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ is typically decomposed into simpler probability terms. A typical decomposition of the posterior is in terms of a measurement model and a motion model. The measurement model describes the probability of making an observation $\mathbf{z}_t$ at time t from a given pose $\mathbf{x}_t$ within a known landmark map $\mathbf{m}_i$.

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_i) \tag{2.2}$$

The motion model describes the motion of the vehicle and the uncertainty of this information. The motion model can be expressed as a probability distribution which describes the effect a control $\mathbf{u}_t$ has on the state $\mathbf{x}_{t-1}$ of the robot. This model represents the probability of the robot being in a new pose $\mathbf{x}_t$ given a previous pose $\mathbf{x}_{t-1}$ and a control $\mathbf{u}_t$ applied to the robot at time $t-1$. A first-order Markovian assumption is typically applied.

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{2.3}$$

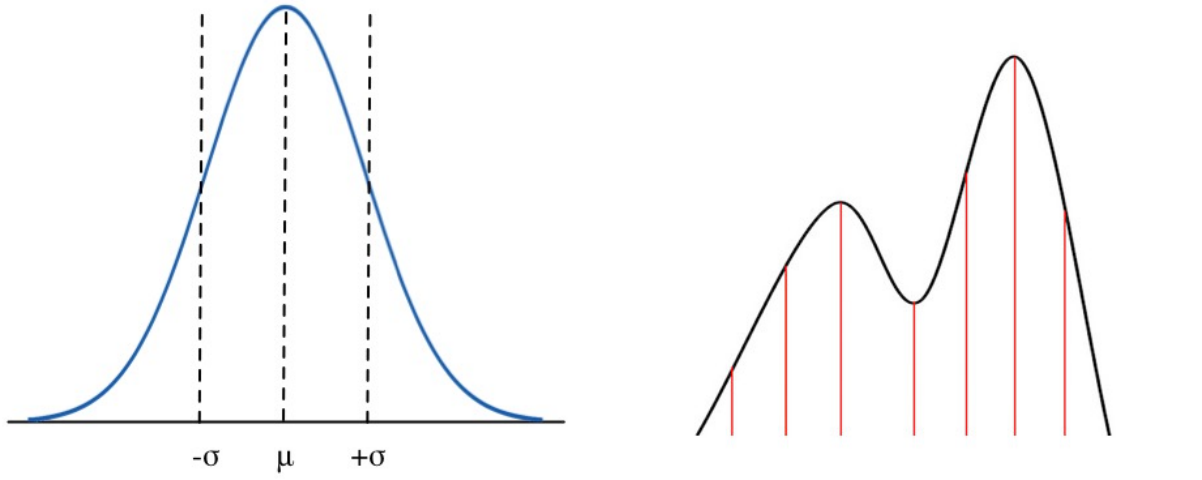
Using probabilistic formulations for both the motion and measurement models the SLAM algorithm can proceed using a two-step recursive process consisting of a time-update (prediction) and measurement (correction) computation.

$$p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t-1}, \mathbf{x}_0) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \quad p(\mathbf{x}_{t-1}, \mathbf{M}|\mathbf{U}_{0:t-1}, \mathbf{Z}_{0:t-1}, \mathbf{x}_0) d\mathbf{x}_{t-1} \tag{2.4}$$

$$p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) = \frac{p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}) \quad p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t-1}, \mathbf{x}_0)}{p(\mathbf{z}_t|\mathbf{Z}_{0:t}, \mathbf{U}_{0:t})} \tag{2.5}$$

Given this Bayesian formulation, the remaining task is one of implementing these probabilistic representations, performing the necessary optimization to determine $p(\mathbf{z}_t, \mathbf{x}_t|\mathbf{M})$, and collapsing this representation to a local 'best' estimate of the state of the robot and the map. The majority of SLAM implementations in the literature fall under one of three main categories in terms of addressing these issues: They can typically be categorized as either Extended Kalman (EKF) Filtering (e.g., [31]), Graph Optimization (e.g., [32]) or Particle Filtering (e.g., [33]) based on the mathematical representation used for the distribution functions. This work utilizes the particle filtering approach because of its efficiency in low dimensional state space, its ability to deal with non-Gaussian distributions of the various probability distributions and its ability to deal with unknown feature correspondences.

2.2.1 Representing distributions



(a) *Gaussian function with mean μ and variance σ^2 used to estimate the posterior.*

(b) *set of particles used to approximate the posterior distribution.*

Figure 2.2: **Approximating the SLAM posterior distribution**
Graphical depictions of two distinct models that are commonly used to represent the SLAM posterior distribution. (a) show the 1-dimensional analog of the Gaussian model. (b) shows the 1-dimensioanl analog of the particle filter model.

A key issue in SLAM algorithms is the mechanism used to represent the various distributions that are required by the algorithm. For some applications (e.g., underwater, flying) it may be necessary to represent the robot's pose as a full 6DOF parameter, and similar representations may be required for landmarks. Fortunately here only 3DOF parameters are required for the robot's pose. But for simplicity here, consider the problem of representing some 1D distribution.

A common approach – and the approach utilized by Kalman-filter based approaches – is to represent this distribution as a Gaussian distribution. Here, the distribution is fully modelled by its mean and standard deviation. Although this can be effective, it is also quite limiting and unfortunately for many of the distributions in SLAM algorithms it is insufficient. The problem is that the underlying distribution is not unimodal, or that it is not symmetric, both of which are properties assumed by a Gaussian distribution. The most likely value in the distribution is not well modelled by the mean of the Gaussian fit, nor is the spread of the distribution well modelled by its variance. An alternative approach is to utilize Sequential Importance Sampling (SIS) also known as a particle filter. The basic approach here is to represent the distribution by a collection of weighted samples, (x_i, w_i) then the distribution $p(x)$ is approximated by $\Sigma_i \ w_i delta(x_i - x)$. A critical issue using sequential importance sample (SIS) particle filters and sequential importance resample (SIR) particle filters is ensuring that the distribution is not under- or over- represented.

To make this abstract difference more concrete, imagine representing a 6DOF distribution with a Gaussian filter. EKF-style SLAM variants represent this distribution using a multivariate normal distribution with mean μ and covariance Σ. The dimensionality of this normal distribution represents the collective state space of the robot's pose and all discovered landmarks k. A 3DOF robot representing a two-dimensional environment requires a covariance matrix Σ that is $2k+3$ by $2k+3$. Particle filter-based approaches such as FastSLAM approximate the distribution through a set of weighted particles. A straightforward implementation would be to model the distribution as a collection of weighted particles of the same dimensionality as the underlying distribution, although more efficient options exist. One such efficiency used in SLAM algorithms involves separating parts of the representation that can be well modelled using a Gaussian representation

from parts that can not. For example the Rao-Blackwell theorem [34] is often used to factor the SLAM posterior into a path posterior and map posterior. This factorization is possible due to the observation that if the location of the robot was known with absolute certainty the map can be represented using a set of independent EKF-style landmarks. The particle filter directly represents the path posterior as each particle is a hypothesized trajectory of the robot through the environment. Indirectly the full posterior is represented because each particle contains its own map with a set of EKF-style landmarks.

2.2.2 Collapsing a distribution

Eventually every SLAM implementation must collapse the posterior probability distribution down to a small covariance, thus confidently rendering a map that conforms to all obtained measurements of the environment. In the literature this is commonly referred to as convergence. When the SLAM implementation is improperly formulated to model the explored environment the covariance never collapses and diverges from a solution.

For EKF-SLAM implementations the posterior distribution must meet a number of requirements in order for convergence to occur. The first condition is that there are numerous landmarks spread throughout the environment that can be used to aid in the localization process. The second condition requires that the motion model (see Eqn. 2.3) accurately represents the kinematics of the robots motion and the associated error. The third condition requires the same of the measurement model (see Eqn. 2.2) for all sensors used within the SLAM implementation. The last condition requires that the data association algorithm is accurate enough to ensure sufficient correct matches. This is especially important because EKF-style SLAM algorithms only accept a single hypothesis for data association. Any significant deviation from any of these conditions can result in divergence from a solution to the problem.

FastSLAM implementations also require that the first three conditions described above are met. The final condition only needs to be met by a single particle. This is due to the fact that each particle can have its own hypothesis as to the association between landmarks and measurements. FastSLAM employs

a sequential-importance-resample (SIR) particle filter to ensure that the posterior distribution is properly represented by the set of particles. After each time step the importance weighting of each particle is re-evaluated (using the inverse of the distance from the expected mean), a new set of particles is randomly selected with replacement from the current set, resulting in a set of particles that more accurately represent the new posterior.

2.3 Adapting terrestrial SLAM algorithms to the aquatic domain

There are a wide variety SLAM implementations in the literature that might be adapted to the aquatic surface domain, some of which have been touched on briefly above. Potential algorithms should be easily adapted, and it is worthwhile examining the desirable properties of potential SLAM algorithm candidates. Given that natural landmarks will occur at or near the shore, and at different elevations it is critical that the algorithm be well suited to representing 3D features at a distance. Thus the map must be representable as a three-dimensional point cloud. An AUV also has access to information about the depth of the water column. Such information should allow an agent to differentiate different locations based on depth, potentially an extremely useful source of information. Thus, a second requirement is the ability to integrate measurements from both a SONAR sensor and camera. The first requirement rules out occupancy grid based SLAM implementations such as GMAPPING[18], DP-SLAM[20] and, GridSLAM[19]. Other implementations such as PTAM[23] and MCPTAM[24][25] satisfy the first requirement; however, they rely on keyframes from image features to describe landmarks and require measurements from camera-like sensors (bearing). The most recent innovative SLAM implementations such as DTAM[35] and KinectFusions[36] are capable of mapping environments in astonishing detail as 3D models which can be considered equivalent to 3D point clouds. These state of the art algorithms produce excellent results using a single visual sensor. Mathematically it is difficult to incorporate non-visual sensors as these algorithms rely on being able to compute some form of optical-flow from one frame to another. These modern and state of the art SLAM implementation show immense promise in situations where GPUs can be exploited to employ the foundations of computer vision,

but they are not ideal candidates for the task here.

As the SLAM problem has become a major area of research in the academic literature most new implementations have become more restrictive in their specification in order to increase their efficiency and/or accuracy. The older and more generalized approaches to the SLAM problem actually provide a more flexible framework for incorporating novel sensors. EKF-SLAM[31] was one of the first SLAM algorithms. EKF-SLAM represents the posterior distribution as a single state variable that is a contiguous array of the robot's state followed by the position of all known landmarks. This state variable is continually estimated using an Extended Kalman Filter by maintaining a covariance matrix that describes the uncertainty between all components of the robot's state every observed landmark, this requires the Jacobian of a number of relationships to be defined mathematically. The largest problem with this approach is that it is highly volatile to incorrect data associations. FastSLAM[33] take a different approach that is conceptually simpler, by representing the posterior distribution as a set of particles or possible paths (based on motion model noise) that the robot has taken. Different paths generate different maps with separate data associations. The landmarks within the maps are then estimated using the EKF approach. At every time-step, particles are resampled based on their likelihood/importance. FastSLAM 2.0 improves upon the original version of the algorithm by incorporating the most recent sensor measurements to more precisely estimate the robot's position. Given its generality, the work presented here is developed within the context of the FastSLAM 2.0 framework. The specifics of FastSLAM 2.0 are detailed in the following section.

2.4 FastSLAM 2.0

Although there are many particle-filter based approaches to Bayesian SLAM (e.g., [18] and [37]) this work utilizes the approach described in FastSLAM 2.0 [38]. FastSLAM 2.0 is based on the observation that if the path of the robot can be measured with absolute certainty, the position of landmarks can be estimated independently of each other. This is represented in Eqn. 2.6 showing the factorization of the posterior

distribution: using a process known as Rao-Blackwellization. Applying the Rao-Blackwell theorem [34] to this problem using a single particle as an improved state estimator, the posterior distribution (Eqn. 2.1) can be factored into a path posterior $p(\mathbf{x}_t|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ and numerous landmark posteriors $p(\mathbf{M}_n|\mathbf{x}_t, \mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$.

$$p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) \prod_n p(\mathbf{M}_n|\mathbf{x}_t, \mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) \tag{2.6}$$

$$\mathbf{M}_n = \{(\mu, \Sigma)_0, (\mu, \Sigma)_1, ...\} \tag{2.7}$$

The Rao-Blackwell theorem states that any estimator of the posterior $p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ is inferior to the factorized estimate shown in Eqn. 2.6 given that the estimate of the path posterior $p(\mathbf{x}_t|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ is a sufficient statistic. In FastSLAM 2.0 each particle can be considered a sufficient statistic because it is an oracle that has absolute knowledge of the robot's position.

In FastSLAM the path posterior $p(\mathbf{x}_t|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ is estimated using a Rao-Blackwellized sample importance resample (SIR) particle filter. Each particle represents a hypothesized trajectory of the robot and maintains its own map of the environment. Each map consists of n landmarks with position represented as a Gaussian distribution described by a mean μ and covariance Σ updated using an extended Kalman filter.

As with many SLAM algorithms FastSLAM 2.0 represents both the measurement and motion models using non-linear functions with additive Gaussian white noise. Here the noise variables ϵ_t and δ_t have covariances R_t and P_t respectively.

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_i) = g(\mathbf{x}_t, \mathbf{m}_i) + \epsilon_t \tag{2.8}$$

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) = h(\mathbf{u}_t, \mathbf{x}_{t-1}) + \delta_t \tag{2.9}$$

In accordance with the sample importance resample (SIR) particle filter employed by the FastSLAM 2.0 algorithm, the first step generates a new location for each particle by sampling from a proposal distribution. Normally the motion model is used as the proposal distribution; however, taking into account the most recent measurement $\mathbf{z}_t$ and observed landmarks $\mathbf{M}$ in addition to the motion of the robot $\mathbf{u}_t$ renders a more accurate distribution. This ensures that a newly sampled pose is less likely to position the current measurement $\mathbf{z}_t$ somewhere that conflicts with the current knowledge of the surrounding environment.

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{z}_t, \mathbf{M}) \tag{2.10}$$

After a new pose has been sampled from the proposal distribution the observed landmark can be updated using a standard EKF-style measurement update to obtain a new landmark mean μ_i and covariance Σ_i. The next step involves calculating an importance weighting for each of the current set of particles. The importance weight associated with each particle indicates the amount of confidence that the particle represents the true path and thus an accurate map. The resample step samples a new set of particles from the current set of particles with replacement. The probability of drawing a particle depends on its normalized importance weight. The resampling step removes particles that represent an unlikely state with particles that represent a more likely state. Some SLAM implementations (e.g., GMAPPING[18]) utilize an adaptive resampling heuristic so that resampling is postponed until the importance weights exhibit a high variance or increasing entropy. This allows the algorithm to explore hypotheses until it is clear that major discrepancies exists between the particles. The SIR particle filter can be continually updated with new measurements and robotic motions throughout its exploration of the environment and halted at any time. A final map can be obtained

by selecting the particle with the highest importance weight. A number of reference implementations of SLAM exist in the literature. This research adapts the implementations described in [18] and [38] to the described problem.

2.4.1 The assignment problem

Visual sensors can produce multiple features in a single time step, and this presents a problem to basic SLAM algorithms. Features discovered in an image represent a single landmark within the environment, associating these features with an existing landmark in an optimal fashion is known as the assignment problem. The assignment problem is related to the field of operations research in mathematics and more specifically the subfield of combinatorial optimization. Given two sets of equal size (agents and tasks) and a cost associated with each agent-task assignment, the goal is to assign all agents to a single task while minimizing the total cost of all assignments. At first this definition may seems rather strict and limiting given that the number of agents and tasks must be equal. However, the assignment problem can be manipulated to obtain the desired flexibility. In the event that the number of agents and task are not equal the smaller set can be padded to include dummy objects. Padding the agent set with "ignore task" objects also requires that the cost of ignoring each task be known. Padding the task set with "do nothing" objects requires that the cost of an idling agent be known. Furthermore the cost matrix can be manipulated to find a maximum global cost instead of minimum.

The Hungarian algorithm [39] is a combinatorial optimization process that can solve the assignment problem in polynomial time $O(n^3)$. The algorithm consists of repetitive matrix row operations on the cost matrix that subtracts the least non-zero cost in the row from all its elements. This process ends when all columns have at least one zero, which signifies that an optimal assignment has been found.

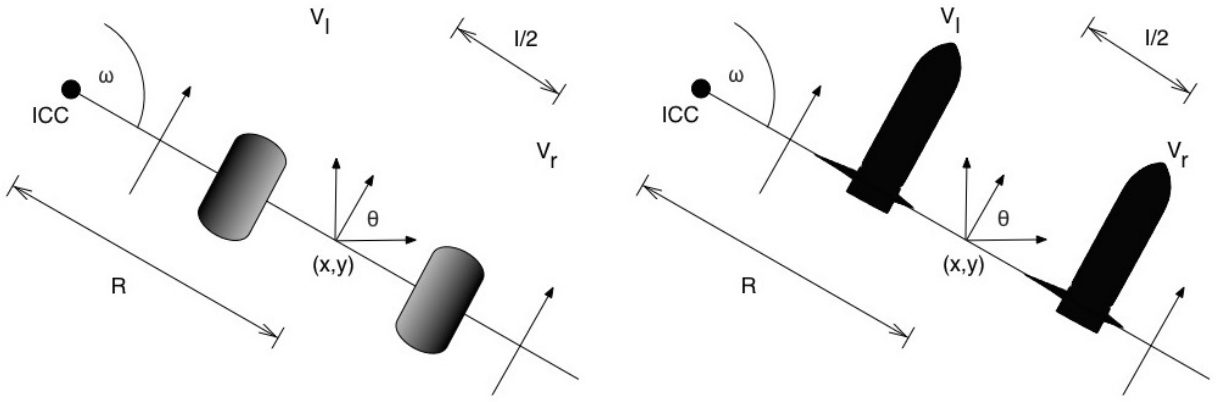
Plant and sensor models

A key issue in adapting a SLAM algorithm to a USV is the development of appropriate plant and sensor models. As discussed previously, the nature of the aquatic domain limits the power of the plant model. Nevertheless, some information can be exploited from the plant model. The following section provides a brief review of the plant model for a differential drive vehicle. Details of the sensors and their corresponding model are described in later chapters.

2.5 Differential drive mechanics

A mature literature exists on the kinematics and dynamics of autonomous vehicles (see [40] for a review). Given the vagaries associated with ground (or water) contact systems, such models are generally implemented within a probabilistic framework providing an expected motion given a control input and some measure of the variance in this estimate. Ground-contact mobile robots typically use wheel encoders to inform the odometry model. Such sensors provide an accurate estimate of wheel rotation, but the resulting model of vehicle motion is subject to errors due to wheel spin caused by a loss of traction. If these models were noise free than mobile robots would be 100% certain as to their location within their surrounding environment. With this information the task of mapping the surrounding environment becomes a trivial problem of independently estimating the position of each landmark[33].

A common drive mechanism used by ground contact robots is "differential drive" which consists of two independently controlled motors mounted on the same axis. Ground contact mobile robots use these motors to turn wheels that maintain friction with the ground allowing the robot to be driven along a curved arc. Fig. 2.3 shows all the variables involved in a differential drive system. The inputs to this system are the velocities of the left and right motors, by varying these velocities the rate of rotation and speed of the robot can be manipulated. Mathematically the motion of a ground contact vehicle can be described in terms of its



(a) *ground contact differential drive using wheels* (b) *aquatic differential drive using propellers*

Figure 2.3: **Kinematic variables pertaining to differential drive**

ICC. The ICC is the instantaneous center of curvature which represents the point that the robot is currently rotating around and ω represents the rate of rotation around the ICC. From the definition of ω its value holds for both wheels. Assuming that the wheels remain in contact with the ground, then:

$$V_r = \omega(R + l/2) \qquad V_l = \omega(R - l/2) \tag{2.11}$$

This can be re-arranged to solve for both R and ω given the velocities V_l and V_r.

$$R = \frac{l}{2}\frac{V_l + V_r}{V_r - V_l} \qquad \omega = \frac{V_r - V_l}{l} \tag{2.12}$$

Although it is possible to drive a differential drive vehicle using complex curved trajectories, a common simplification is to develop a small language of simple motions and to decompose more complex motion trajectories in terms of sequences of these simple motions. To see this, observe the following special cases:

- $V_r = V_l$: When both motors are set to the same speed the robot travels in a straight line. The rate of rotation ω becomes 0 and the ICC extends out to infinity on either side and in turn R become infinite.

- $V_r = -V_l$: If both motors are set to opposing speeds the robot then rotates around the midpoint between both motors. The rate of rotation $\omega = 2V_r/l$, the ICC equals (x, y) and R becomes 0.

- $V_r = 0 \oplus V_l = 0$: If only a single motor's speed is set to zero then the robot rotates about the position of this motor. The ICC is located at the position of the opposing motor, $R = \pm\frac{l}{2}$.

The mechanics of differential drive imposes non-holonomic constraints on the motion of the vehicle. This means that the controllable degrees are less that the total degrees of freedom. A differential drive vehicle has only two degrees of freedom. Changes in position are not fully decoupled from changes in orientation. The vehicle cannot, for example move along the line defined by the wheel axis without first changing the orientation of the vehicle. This implies that the vehicle may have to execute complex maneuvers in order to move from state to state. Eqns.2.11 and 2.12 can be used to develop a formal plant model for a differential drive vehicle. Suppose that the vehicle is embedded in a 2D cartesian space such that the vehicle has position (x,y) and orientation θ relative to the x axis. Then given V_l and V_r at some time t, we can identify the ICC of the vehicle in the cartesian space.

$$ICC = [x - Rsin(\theta), y + Rcos(\theta)] \tag{2.13}$$

This can be used to develop an expression for the updated state of the robot after a period of time dt.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} cos(\omega dt) & -sin(\omega dt) & 0 \\ sin(\omega dt) & cos(\omega dt) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega dt \end{bmatrix} \tag{2.14}$$

Eqn. 2.14 can be used to obtain an instantaneous update of the state of the robot and describes an agent rotating about the ICC at a rotational speed ω for a period of time dt. This equation can be simplified in the

event the conditions for either of the first two special cases described above are met. The first special case, when $V_r = V_l$ and travelling in a straight line renders the simplification shown in Eqns. 2.15. The second special case, when $V_r = -V_l$ and rotating about the midpoint between its two motors.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + vcos(\theta)dt \\ y + vsin(\theta)dt \\ \theta \end{bmatrix} \quad (2.15) \qquad \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + 2vdt/l \end{bmatrix} \quad (2.16)$$

The simplified forward kinematic model shown in Eqns. 2.15 and 2.16 motivates the use of a simple navigation strategy. The requirement that the robot move from one waypoint (x, y, θ) to another (x', y', θ'), can be decomposed in terms of combinations of motions give by the plant models given in Eqns. 2.15 and 2.16. The robot first rotates in place until it is pointed in the direction of the intended target and then travels straight towards the target. Once it gets there it rotates in place until it has the necessary final orientation. This can be an effective strategy with ground-contact robots that operate in office spaces with narrow hallways. The most optimal use of energy to travel to multiple waypoints uses more that a single degree of freedom at a time.

2.5.1 Differential drive kinematics for aquatic environments

A wide range of different power/control surfaces exist in the aquatic domain (see [41] for a review). Different combinations of control surfaces and thrusters meet different task-related constraints. For example, rear thruster/rudder control surfaces are well suited for vehicles that move long distances with little requirements for maneuvering, while vehicles with multiple thrusters are better suited for maneuvering but less well suited for long distance travel. Differential drive kinematic systems offer a good compromise for vehicles engaged in survey/sensing tasks as they still provide a high level of maneuverability although they are not fully

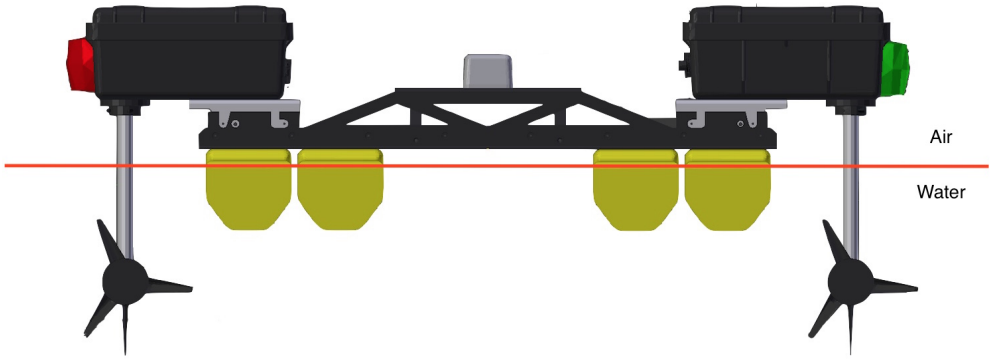


Figure 2.4: **Separation of drag forces**
The figure shows the waterline which separates the two different forms of drag that have a large effect of the robot's motion: aerial drag and fluidic drag.

holonomic, as described above. Fig. 2.3b illustrates a differential drive-like control system for an aquatic surface vehicle. The drive mechanism can be considered equivalent to ground-contact differential drive as long as drag is not taken into account. Ground-contact robots maintain a significant amount of friction on each ground contact point. This prevents most weather conditions from having a significant on the robot's motion. The situation is somewhat different for aquatic surface vehicles. Wind and wave action introduce aerodynamic drag (air resistance, fluid resistance) that act on an USV, both of which have a number of detrimental effects on the motion of the robot. To take but one example, the stopping distance of the vehicle is drastically increased compared to ground-contact vehicles and is affected by the drag created from robotic components under the waterline.

Modulo the issue of vehicle drag and wind/wave action, the kinematics of a differential drive system in an aquatic environment differs only slightly from one operating in a terrestrial environment. The kinematic model shown in Fig. 2.3 is sufficient to describe the motion of ground contact robots because very few commonly occurring natural forces can overcome either the static of kinetic friction maintained the robots wheels. Given that the ground-contract robot differential drive model will only provide a rough approximation to the motion of a differential drive USV, it is worth exploring the potential of developing a more sophisticated

hydrodynamic model for a differential drive USV. Integrating both air resistance and fluid resistance using the drag equation (shown in Eqn. 2.17) into a model for aquatic differential drive has numerous pitfalls. Calculating the drag force F_{drag} requires estimating values for all the dependant variables [42]: flow velocity u, fluid/gas density ρ, drag coefficient C_{drag} and, specific area A.

$$F_{drag} = \frac{1}{2}\rho u^2 C_{drag} A \tag{2.17}$$

Flow velocity u_{wind} of the wind relative to the robot can be measured using a vane anemometer which can measure both the direction and speed of the prevailing winds. Vane anemometers are also appropriate for measuring flow velocity u_{fluid} of a fluid. Measuring the density of a fluid/gas is simple if its volume and mass can be obtained. However, commercial off-the-self sensors that can effectively and efficiently measure density are not available. This value can be estimated beforehand; however, there exist variances in the chemical composition of both any fluidic body and atmosphere.

The drag coefficient C_{drag} is a ratio that combines both skin friction and form drag. Skin friction describes the friction between a fluid and the surface of a object moving through that fluid. The caveat here is that the chemical composition and location of particulate matter are unknown and not constant. Form drag describes the friction created from an object‘s resistance to movement through a fluid, this type of drag depends highly on the geometry of the object and its orientation with respect to the direction of motion. The problem presented here is that traditionally these friction coefficients are measured in highly controlled environments in order to obtain accurate values.

The specific area A refers to the cross-sectional area that is perpendicular to the relative flow velocity u. In stagnate waters and calm weather only an accurate three-dimensional model of the robot, its moment of inertia and fluid density ρ_{water} are needed to calculate the volume of the robot submerged within the fluid. This calculation can be done by using cameras that observe the waterline contacting the robot frame

by frame; however, this would require 360 deg observation of any portion of the robot that is partially submerged.

Even though estimating and/or measuring these values is possible, each has an associated error. These values and their associated uncertainty are compounded twice over due to the relationship between force and position when integrated into the forward kinematics of a standard differential drive. This presents a major problem as these errors are further compounded over time to produce substantial odometry errors. Given the above, and the difficulty of even this more sophisticated treatment to properly incorporate wind, wave and current action on the vehicle, this work uses the simple differential drive model given in Eqn. 2.14 under the assumption that errors will be large.

2.6 Summary

This chapter provided a review of SLAM algorithms in general and their applicability to USV more specifically. As more recent SLAM algorithms are typically highly targeted to specific domains far removed from the USV domain, the work here utilizes FastSLAM 2.0 as a framework for SLAM for USV's, and thus FastSLAM 2.0 is described in some detail. This chapter has also provided a review of the notation associated with USV and a simple plant model for a differential drive USV that is the straightforward adoption of the differential drive vehicle from ground contact robotics.

Chapter 3

Eddy : an autonomous aquatic surface vessel

The previous chapter reviewed the theoretical underpinnings of the work described here. This chapter reviews the practical underpinnings of this work. More specifically, this chapter describes the hardware and software infrastructure upon which the work was performed and the development/modifications required to provide sensor systems that can obtain visual and water column features to support the SLAM algorithm described in the following chapter.

3.1 Robotic platform

The work conducted in this thesis utilizes the Kingfisher M100 robot [43]. Shown in Figure 3.1, the Kingfisher M100 is a ROS-based unmanned surface vessel manufactured by Clearpath Robotics^{TM} for the purpose of marine autonomy and surveillance research. As shipped, the Kingfisher comes equipped with a motor controller, a digital compass and differential GPS enabled by a land locked GPS base station. This equipment facilitates basic operation of the vessel through tele-operation and accurate localization of the robot's pose

(a) *Kingfisher M100 USV with mounted SONAR and omnidirectional sensors.*

(b) *GPS Base Station and Monitoring Computer.*

Figure 3.1: **Clearpath Kingfisher**
(a) The Kingfisher robot deployed on lake Okanagan with the omnidirectional sensor and the depth sensor described in this work. The original Kingfisher robot has been modified through the addition of sensors, buoyancy and computational resources. (b) The GPS base station that is used to provide off-robot command and communications infrastructure as well as differential GPS signals to the robot.

and acquired sensor data. In order to conduct this research, the stock vehicle has been reconfigured (as shown in Fig. 3.2) and outfitted with a range of different sensor systems and upgraded onboard computing power. Additions relevant to the work described in the thesis are as follows: (1) An ultrasonic transducer or depth sensor, (2) an omnidirectional camera and (3) a number of software packages to integrate and process sensor data. The process of augmenting the Kingfisher robot is described elsewhere (see [44][45]) and includes the introduction of substantive additional mass to the vehicle and the addition of extra pontoons to enhance vehicle buoyancy to compensate for the additional mass.

3.2 Depth sensor

The SLAM algorithm developed in this work relies on water column depth and visual targets associated with the shore. This section describes the depth sensor, while the following section describes the omnidirectional visual sensor. A custom depth sensor was developed in order to outfit the Kingfisher with the necessary sensing capabilities to conduct this research (see Fig. 3.3). The RECHOS [45] (ROS Echo Sounder sensor)

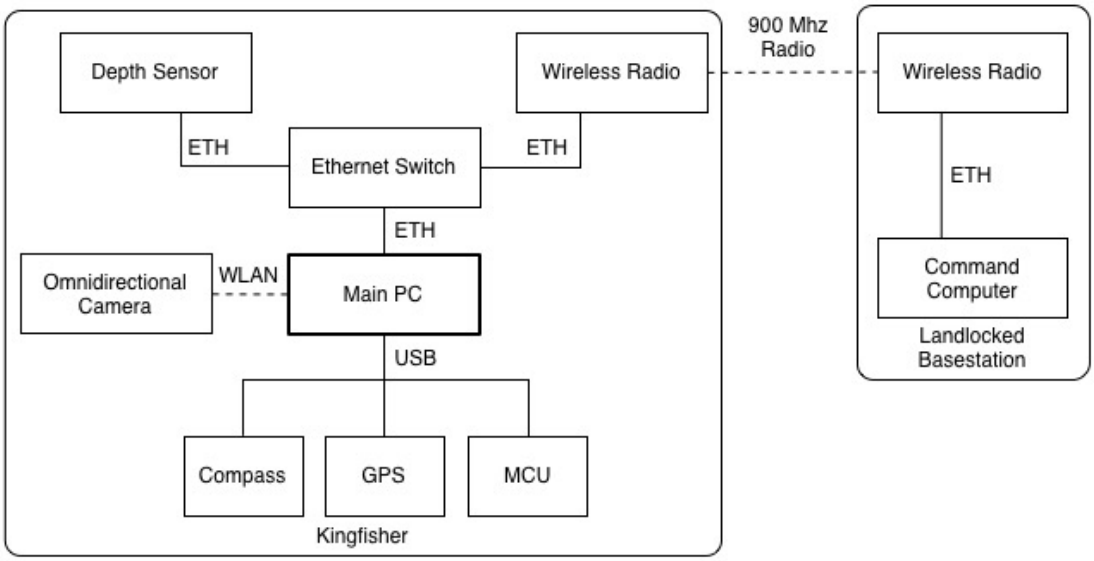


Figure 3.2: **Kingfisher hardware**
Standard peripherals such as the compass, GPS and motors control unit connect to the primary computer via usb. The onboard omnidirectional camera hosts its own wireless network which the robot connects to via its wlan0 network interface. The onboard wireless radio creates a local area network (LAN) that facilitates communication between the depth sensor, primary computer and finally the onshore command computer used for controlling the robot.

was designed to be a low cost fully self-contained system that can operate independently of the target robotic platform and costs approximately $500. If the sensor is connected via ethernet to a LAN it scans all local hosts for an instance of ROS to connect to, otherwise a local instance is started. This behaviour allows the sensor to operate on a network utilizing DCHP to assign IP addresses. The sensor is comprised of three major components:

- **An echo sounder.** The CruzPro-ATU120AT [46] is an active depth and temperature transducer with a 0.1m resolution requiring 9.5-16.0 VDC at 350 mA that transmits data at 1Hz using the NMEA 0183 communications protocol. This device provides the necessary acoustic emitter/receiver hardware.

- **Signal transducing computer.** The RECHOS uses a Raspberry Pi Model-B Rev-01 [47] for signal processing. This computer is a small single-board computer requiring 5VDC at 700 mA.

- **ADXL345 Accelerometer.** A triple-axis accelerometer [48] with digital I2C interface provides tilt

(a) *Disassembled sensor showing: environmental housing, internal hardware structure, status LEDs and power/data ports.*

(b) *Sensor mounted on the robot. The lower end of the sensor protrudes below the surface of the water. The internet connection to the onboard ROS infrastructure can be seen at the top of the sensor*

Figure 3.3: **The RECHOS Sensor**

information which is used to correct depth readings due to pitch and roll of the device due to wave action.

Apart from these major components the RECHOS also includes separate and isolated power to cope with the incompatible power requirements of the Raspberry Pi and NMEA depth and temperature transducer. The RECHOS performs the task of converting messages from the transducer and accelerometer and presenting them as custom ROS messages. In addition to acting as a signal transducer, data from both sensors are combined to produce corrected depth readings less subject to error due to device tilt. A more detailed description of the design and construction of this sensor can be found in [45].

3.2.1 Calibration

SONAR measures distance using time of flight by sending out ultrasonic pulses and measuring the time it takes for the sound to return. Calculating the distance traveled requires knowledge of the speed in which sound propagates within the target medium, on earth this medium is usually fresh or salt water. The speed

at which sound propagates through fresh water is slower then in salt water and this speed is further affected by the temperature. Both the salinity and temperature of the water vary as a function of depth, location and time. The temperature sensor on the CruzPro-ATU120AT only measures the water temperature at the surface which can be drastically different from the temperature at the lake or sea bed. The speed, s, of sound in water can be expressed as a function of temperature, t, in celsius, depth, m, in meters and salinity, sl, in parts per thousand [49].

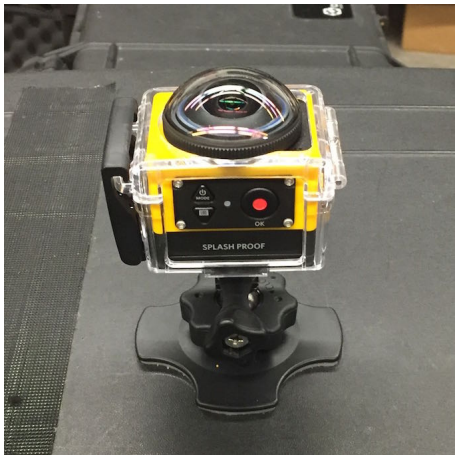
$$s_{water} = 1337.462 + (3.429 \times (\frac{9}{5}t + 32)) + (0.00554736 \times \frac{m}{3.28}) + 0.3048sl \tag{3.1}$$

The CruzPro-ATU120AT comes calibrated with a predetermined and constant salinity value. In shallow waters (less than 20m) the overall effect of salinity on the time of flight and thus measured depth is considerably smaller than the sensor's 0.1m resolution. Only at greater depths do large changes in salinity begin to have an effect on measured depths greater than the sensors resolution.

3.3 Omnidirectional camera

The second primary sensor for the SLAM algorithm developed here is an omnidirectional visual sensor. An omnidirectional camera is a visual sensor that has 360 degree horizontal field of view and a large vertical field of view. Normally standard dioptric cameras can be converted into omnidirectional catadioptric cameras using curved mirrors. The Kodak Pixpro SP360[50] shown in Fig. 3.4 is a compact dioptric omnidirectional camera with a 214 degree field of view. This camera is capable of capturing video at 30 FPS with a resolution of 1072x1072 and is equipped with electronic image stabilization and a water resistant housing [51]. Images can be extracted by connecting to the camera's wifi network and scrapping image data from the onboard web server.

(a) *360 degree omnidirectional water resistant camera.*

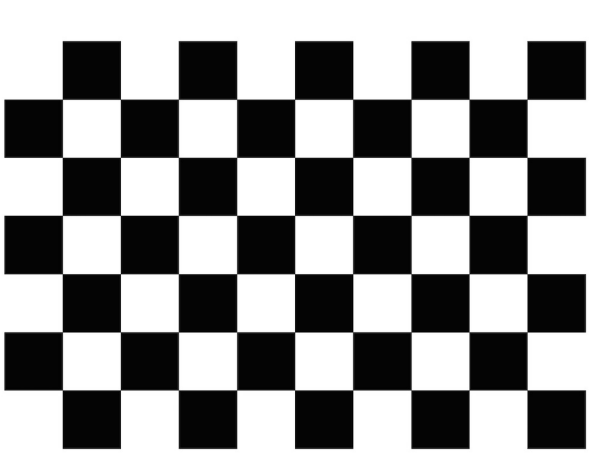


(b) *Camera in a standard housing mounted to the robot using a ball and socket joint.*

Figure 3.4: **Kodak Pixpro SP360 Camera**
(a) shows a vanity picture of the Pixpro SP360. (b) shows the Pixpro SP360 in the standard housing mounted on top of the robot's central pelican case.

3.3.1 Calibration

It is often useful to calibrate visual sensors in order to map pixel locations on the image sensor to directions in the real world. That is, to define a mapping from a sensor pixel location to a ray in the global coordinate system. A large calibration literature exists for traditional 'pin hole' cameras (see [52], for example). The literature for wide field 'fisheye' lens cameras is also quite mature, although techniques differ somewhat from those used with pinhole cameras. The fisheye lens on this camera cannot be accurately modelled using the standard plumb bob camera model [52] used by computer vision libraries like OpenCV[53]. This is due to the large field of view and high levels of radial distortion (see Fig. 3.5b). The plumb bob camera model is rendered ineffective when dealing with FOVs larger than 180 degrees due to the physical limitations of the mathematical model. Large radial distortion negatively effects standard techniques for reliably detecting standard checkerboard pattern calibration targets (as shown in Fig. 3.5a). Fisheye lenses can also increase visual blur close to the image border. The standard process for calibrating fisheye omnidirectional cameras still uses the same checkerboard pattern used in traditional camera calibration, with some major behind the



(a) *Standard 9x6 black and white chessboard image used as a visual calibration target. The fisheye distortion of this target is readily apparent in (b).*

(b) *Fisheye camera in a standard housing mounted to the robot using a ball and socket joint.*

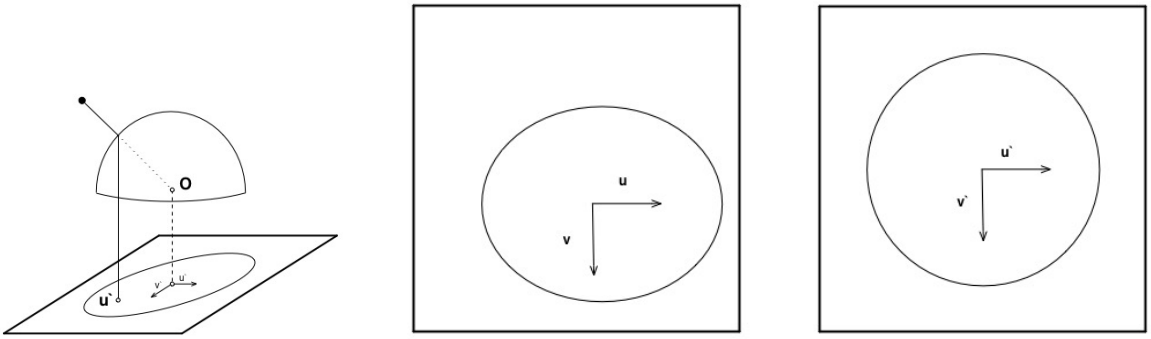
Figure 3.5: **Omnidirectional Camera Calibration**

scene changes. Techniques for detecting calibration targets in images with large radial distortion and large FOVs are described in [54]. [55][56] describe an alternative camera model that uses a Taylor polynomial series to represent the geometry of the lens. The mathematical modelling of the wide FOV sensor described below follows that presented in [55].

Let X be a point in the world space. Then $\mathbf{u'} = [u', v']^T$ is the projection of X onto the sensor plane in metric coordinates, and $\mathbf{u} = [u, v]^T$ is the mapping of $\mathbf{u'}$ onto the camera's image plane in pixel coordinates. Both $\mathbf{u'}$ and $\mathbf{u}$ are related by an affine transformation which incorporates pixilation errors and axial misalignment, thus $\mathbf{u} = A\mathbf{u'} + t$. The image projection function g, defines the relationship between a point $\mathbf{u'}$ in the sensor plane and a vector $\mathbf{x}$ stemming from the camera's viewpoint O.

$$\lambda \cdot p = \lambda \cdot g(u, v, f(u, v)) = \lambda \cdot g(A\mathbf{u} + t) = PX, \lambda > 0 \tag{3.2}$$

The above equation defines the complete omnidirectional camera model, where $X \in \mathbb{R}^4$ is expressed in



(a) *Geometric projection model of a dioptric omnidirectional camera.*

(b) *Dioptric projection of a scene onto the camera's sensor.*

(c) *Image plane after an affine transformation.*

Figure 3.6: **Wide FOV camera model**

homogeneous coordinates and $P \in \mathbb{R}^{3x4}$ is a perspective projection matrix. The geometric properties of the camera's lens are modelled by Eqn. 3.3.

$$f(u, v) = a_0 + a_1\rho + a_2\rho^2 + \cdots + a_N\rho^N \tag{3.3}$$

The lens geometry is assumed to be radially symmetric with respect to the sensor axis. The modelling function f is a Taylor series approximation of the more complex true nature of the function where ρ is the metric distance of (u, v) from the sensor axis, thus $\rho = \sqrt{u^2 + v^2}$. The calibration of an omnidirectional camera using this model involves the estimation of the variables of $[A, t, a_0, \ldots, a_N]$ so as to satisfy Eq 3.2. These variables can be solved for using a similar process to that used by the standard plumb bob camera calibration process. This process requires a collection of pictures taken of a planar surface with known geometric properties (such as, a checkerboard or asymmetric grid of circles).

This function defined by the camera model does not have an inverse but can be used to define the inverse projection of a pixel in the camera's image plane onto a point on the unit sphere, from which a bearing and elevation can be computed.

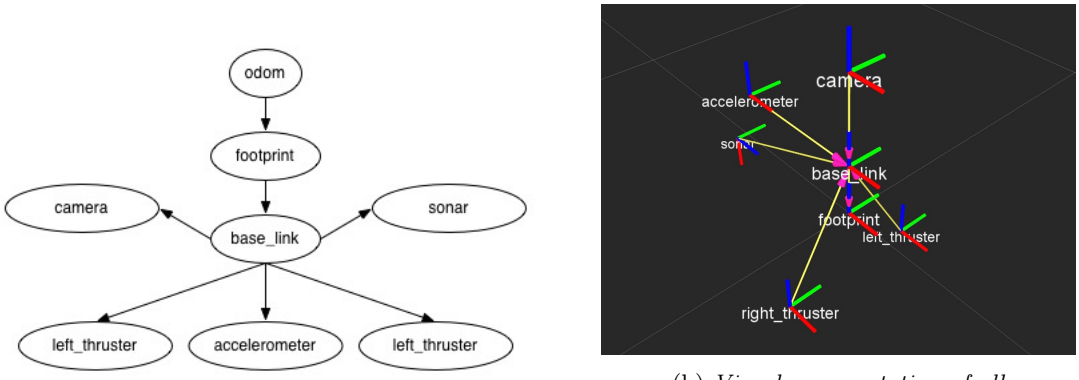
ROS packages	Description
`clearpath_base`	contains the logic for monitoring power levels and applying voltage to each motors inorder to produce the desired command velocity.
`clearpath_bringup`	manages the announcement and presence of robots within the ROS environment.
`clearpath_sensors`	contains ROS nodes for reading data from both the GPS sensor and compass and publishing this data to ROS. Also includes .launch files for starting each node in its standard configuration.
`clearpath_teleop`	contains a simple ROS node that translates joystick messages from ROS into velocity commands to the robots motor controller logic.
`kingfisher_bringup`	this meta-package contains .launch files for starting the nodes necessary to read data from the sensors and drive the onboard motors.
`kingfisher_teleop`	this meta-package contains a .launch file that starts all nodes required to teleoperate the kingfisher.
`joy`	translates joystick input data from linux into standard ROS messages for use as a controller.
`pixpro`	connects to an video stream from a Kodak Pixpro SP360 omnidirectional camera and publishes images to ROS while running a camera info server for dynamic configuration of the camera's intrinsic and extrinsic parameters.

Table 3.1: **ROS software packages**

3.4 Software

ROS [57] has become a de-facto standard for much of the autonomous robot community and its use is especially prevalent in the research community. Within ROS, processes are represented as nodes that communicate through messages within a publisher/subscriber framework. Nodes may also advertise services that act as remote procedure calls using a request/reply framework. Apart from this flexible architecture ROS also provides powerful tools for the purpose of data collection, management and visualization. Due to the popularity of ROS within the robotics community numerous open source software packages that provide high-level features common to the majority of robotics applications have been developed.

The Kingfisher comes with a number of software packages that are used to implement the robot's basic functions including sensor monitoring and teleoperation. The basic functionally of the robot is provided



(a) *Transformation Tree*

(b) *Visual representation of all coordinate frames*

Figure 3.7: **Kingfisher transformations frames**
(a) An abstract representation of the robots tree of transformation frames and the relationship between those frames. (b) A geometric representation of the robots transformation frames.

by the ROS packages detailed in Table. 3.1. The majority of these packages were provided by Clearpath Robotics^{TM}and later modified to operate on the replacement computer. These packages provide the ability to drive the motors and to monitor the output of the onboard compass, GPS sensor and motors. These packages were modified to include a heartbeat sensor that requires constant input to ensure the robot does not move unless continually connected to the base station, upon the restablishment of a connection the heartbeat must be reset to enable control. The *joy* package is a standard ROS package that translates input from a joystick device into ROS messages. The final package *pixpro* was developed to grab images from the omnidirectional directional camera's web server that streams images from the camera at 30 FPS.

Most robotic platforms utilize a wide variety sensors each with their own inherent local coordinate systems. In order for measurements from these sensors to be entirely useful there needs to be common coordinate frame into which all measurements can be transformed. Following the normal ROS standard, the unified coordinate frame is named *base_link* and is positioned at the rotational center of the robot. For convenience sake a companion frame name *footprint* is defined as the projection of the origin of *base_link* onto the ground plane, in this case the ground plane is synonymous with the surface of the water. The transformation from *base_link* to *footprint* encompasses all extraneous pose variables that are too computationally expensive to

be modelled within FastSLAM.

The tf library [58] bundled within ROS provides a variety of tools for defining both static and dynamic transformations between coordinate frames as well as viewing these relationships. The sensors on the Kingfisher are rigidly attached to the robot so only static transformations from the *base_link* frame are required to fully define a tree of transformations connecting all relevant coordinate frames. As previously described the transformation from *base_link* to *footprint* includes information about extra pose parameters: roll, pitch and heave. This information is calculated dynamically based on incoming sensor data and thus the *base_link* to *footprint* transformation also needs to be dynamically defined using the appropriate tools provided by the tf library.

Chapter 4

PyFSlam

This chapter describes the theoretical underpinnings of PyFSlam, a SLAM implementation for unmanned surface vessels relying on shore features and depth measurements. PyFSlam utilizes the framework introduced in FastSLAM 2.0[38] and utilizes the multiple landmark extension introduced in [59]. The system relies on ROS and the bulk of the code above the sensor conditioning level is implemented in Python.

4.1 Adapting FastSLAM 2.0

Adapting FastSLAM 2.0 to the mapping problem considered here requires generalizing its model of sensor data to deal with the omnidirectional camera data, depth and (optionally) compass data that are available to the robot. It also involves providing explicit models for the vehicle plant and sensors including estimating noise properties for same. Each of these issues are addressed in this chapter. One particular issue here is that as originally developed, FastSLAM assumes only a single landmark measurement in each iteration of the algorithm. The omnidirectional camera used here typically provides many measurements at once.

The FastSLAM 2.0 algorithm addresses the SLAM problem to processing of a single measurement $\mathbf{z}_t$ with a known landmark association $\mathbf{m}_i$ at each time step. A single incoming measurement $\mathbf{z}_t$ can be associated with a mapped landmark $\mathbf{m}_i$ using a maximum likelihood algorithm; however, this strategy does not scale well to multiple measurements at the same time time step for a number of reasons. Perhaps the most important of these relates to the data association process. Consider two measurements, where the first measurement has only slightly more likelihood of association with a landmark A than the other while only the second measurement has a high likelihood of association with a second landmark B. Using a maximum likelihood algorithm for each landmark/measurement association separately could lead to a non-optimal solution to this problem. With multiple measurements the goal is to maximize the overall likelihood for the entire set of measurement associations. This data association problem can be solved in $O(n^3)$ time using the Hungarian algorithm[39]. Without additional information both the Hungarian algorithm and the maximum likelihood algorithm do not allow for the observation of new landmarks as every measurement must be assigned to a landmark. The introduction of a threshold likelihood for the association of a measurement with a new landmark is essential to overcome this limitation. Choosing an optimal value for this threshold is a common problem for SLAM algorithms. For this work a threshold value(approximately 72%) was chosen through careful tuning. Likelihood thresholds not within $\pm 5\%$ of this value began to rapidly degrade PySLAM's effectiveness.

A second problem associated with incorporating simultaneous sensor measurements per step of FastSLAM 2.0 involves the usage of only a single control variable. Dealing with this problem under the basic FastSLAM 2.0 formulation involves choosing a single measurement to process alongside a given control variable and processing the rest of the measurements with a control variable that produces no motion. However, this approach does not solve the initial problem as the proposal distribution that is used to sample a new pose is then highly dependant on the order in which measurements are processed. Obviously the original FastSLAM 2.0 algorithm needs to be reformulated to deal with simultaneous measurements. The multiple measurement modification of FastSLAM 2.0 used here follows the approach presented by Gamallo et al. in [59] but refines

this approach as described below.

In order to properly represent this modification to the FastSLAM algorithm it is necessary to generalize slightly the notation used in the original algorithm to account for multiple measurements at a given time t. (The initial description of FastSLAM 2.0 assumed only a single scalar measurement at each time step.) Let

- $\mathbf{Z}_{0:t} = (\mathbf{Z}_0, \mathbf{Z}_1, \ldots, \mathbf{Z}_t)$ represent the sequence of sets of sensor measurements. (Note that unlike the formulation of FastSLAM given in [38] and described earlier, now each measurement is an unordered collection of sensor measurements.)
- $\mathbf{Z}_t = \mathbf{z}_{0,t}, \mathbf{z}_{1,t}, \ldots, \mathbf{z}_{l,t}$ represent the set of sensor measurement observed from the robot at time t.
- $\mathbf{z}_{l,t}$ represent the l-th sensor measurement observed from the robot at time t.

The FastSLAM 2.0 algorithm estimates the posterior distribution $p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0)$ using particles that represent a single hypothesis. To properly refer to variables in each particle the following definitions are required.

- $\mathbf{Y}_{0:t} = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_t)$: the sequence of sets of particles (shows particle evolution over time).
- $\mathbf{Y}_t = \{\mathbf{y}_t^0, \mathbf{y}_t^1, \ldots, \mathbf{y}_t^t\}$: the set of particles at time t.
- $\mathbf{y}_t^k = \{\mathbf{x}_t^k, \mathbf{M}^k\}$: the k-th particle at time t, containing the current estimated state $\mathbf{x}_t^k$ and set of hypothesized landmarks $\mathbf{M}^k$.

In the scalar FastSLAM 2.0 formulation, a measurement can be matched to at most one landmark. As measurements become simultaneous, the assignment process becomes one of determining possible matches from the collection of measurements to the collection of landmarks. Given this state of affairs, the non-scalar FastSLAM algorithm was generalized in [59] to deal with non-scalar values. These generalizations are detailed in Algorithm 4.1. This algorithm relies on a number of complex operations as described in Algorithms 4.2-4.4.

Culling non-useful landmarks

Given the large number of possible landmarks and the cost associated with processing them, it is necessary to prune landmarks that are not used in the map. In order to do this the algorithm keeps track of the number of times each landmark has been sighted and the number of steps between sightings. This is done so that landmarks that are unlikely to be be seen again or have a large uncertainty can be culled from the map. This helps to reduce the number of landmarks and improve the computational efficiency of the data association process.

- $i^k_{\mathbf{m}_j,t}$: number of times the landmark $\mathbf{m}_j$ of the k-th particle has been sighted within the environment at time t.
- $TTL^k_{\mathbf{m}_j,t}$: A value that represents the liveliness of a given landmark $\mathbf{m}_j$ of a given particle k at time t. When this value falls below a given threshold the marker is removed. Liveliness is a integer value that represents the number of consecutive steps a landmark can go unseen before it is removed.
- $vol(\Sigma_{\mathbf{m}_j,t_{\mathbf{m}_j}})$: volume of the landmark $\mathbf{m}_j$'s uncertainty at time $t_{\mathbf{m}_j}$ (the time when the landmark was created).

We wish to prune landmarks that have not be 'seen' by any particle recently. To do this, we define three constants that define how each landmark's time to live $TTL^k_{\mathbf{m}_j,t}$ value is modified by specific events. These constants are specific to the measurement type of $\mathbf{z}_{t,l}$

- $\alpha_{\mathbf{z}_{t,l}}$: multiplier to apply to an existing landmark's $TTL^k_{\mathbf{m}_i,t}$ value in the event that it has been sighted again within the environment.
- $\beta_{\mathbf{z}_{t,l}}$: number of sightings $i^k_{\mathbf{m}_j,t}$ before a landmark of the measurement type $\mathbf{z}_{t,l}$ is eligible to become a permanent fixture of the current particle's map.

Algorithm 4.1 FastSLAM 2.0

procedure FastSLAM
  for each $\mathbf{y}^k \in \mathbf{Y}_{t-1}$ **do**
    $\hat{\mathbf{x}}_t^k = g(\mathbf{x}_{t-1}^k, \mathbf{u}_t)$
    $\Phi = measurementsLikelihood(\hat{\mathbf{x}}_t^k, M^k, \mathbf{Z}_t)$
    $\Psi = dataAssociation(\Phi)$
    $robotPoseUpdate()$
    $landmarksUpdate()$
  end for
  $Y_t = sampling(\hat{Y}_t)$
end procedure

- $\gamma_{\mathbf{z}_{t,l}}$: numerical value used to initialize the the liveliness value $TTL_{\mathbf{m}_j,t}^k$ of a landmark created from the measurement $\mathbf{z}_{t,l}$.

This algorithm also incorporates landmark/measurement descriptions that are used to further discriminate between measurements and landmarks. The inclusion of these descriptors helps to rule out measurements that might otherwise be associated with a landmark given different criteria, mainly proximity.

- $\mathbf{D}_{\mathbf{z}_{l,t}}$: quantitative description of the l-th measurement observed at time t.
- $\mathbf{D}_{\mathbf{m}_i^k}$: quantitative description of the i-th landmark in the k-th particle.

4.1.1 Measurement likelihood

The measurement likelihood algorithm from [59] shown in Algorithm 4.2 calculates a two dimensional matrix ϕ which encapsulates the probability that a landmark $\mathbf{m}_j^k$ corresponds to a measurement $\mathbf{z}_l$ ($\phi_{j,l}$). Unknown data associations in traditional scalar FastSLAM 2.0 implementations use a maximal likelihood approach to solve for the correspondence $\mathbf{n}_t$ between a landmark $\mathbf{m}_i$ and an incoming measurement $\mathbf{z}_t$. Generalizing this

approach to multiple simultaneous measurements uses the matrix ϕ in order to determine a globally optimal solution to the data association problem.

Calculating Φ requires the evaluation of each new measurement against all known landmarks, rendering a likelihood $\Phi_{j,l}$. Lines 2 to 12 iterate through the entire set of landmarks M^k in the current particle and all measurements Z_t taken at time t to calculate $\Phi_{j,l}$. The proximal likelihood $\phi_{j,l}$ is calculated using a Gaussian distribution with mean $\hat{z}_{j,l}$ (predicted measurement) and $Q_{j,l}$ (innovation covariance). The proximal likelihood $\phi_{j,l}$ is further augmented using a weighting function $w(\phi_{j,l})$ which adjusts its influence on the overall likelihood $\Phi_{j,l}$ based on the certainty of the landmark $\mathbf{m}_j^k$. The measurement innovation (or residual) covariance matrix $Q_{j,l}$ takes into account the measurement uncertainty $P_{\mathbf{z}_l}$, the previous landmark covariance $\Sigma_{\mathbf{m}_j^k}$ and the Jacobian of the measurement model $\nabla_{\mathbf{x}_t} h$. The predicted measurement $\hat{z}_{j,l}$ of landmark $\mathbf{m}_j^k$ is calculated using the measurement model h at the predicted robot pose $\mu_{\mathbf{x}_t,j,l}$ based on the assignment of the measurement $\mathbf{z}_l$ to the landmark $\mathbf{m}_j^k$. The predicted robot pose and accompanying covariance are calculated using Eqns. 13-15 given in the introductory paper on FastSLAM 2.0[38] and reprinted in Algorithm 4.3. Finally the likelihood is multiplied by $\lambda_{j,l}$ which represents the probability that the landmark description $\mathbf{D}_{\mathbf{m}_j^k}$ corresponds to the measurement description $\mathbf{D}_{\mathbf{z}_{l,t}}$.

4.1.2 Data association

The Data association performed in line 5 of Algorithm 4.1 assigns each incoming measurement to either an existing landmark or identifies it as a new landmark. This is a simple assignment problem and can be solved using the Hungarian Method in polynomial time (see Chapter 2). The Hungarian Method takes a cost matrix Φ and seeks to minimize/maximize the overall cost of all assignments. In order to incorporate new landmarks, Φ is expanded to include the probability of observing a new landmark. This requires Φ to be a $(\left|\mathbf{M}^k\right| + |\mathbf{Z}_t|) \times |\mathbf{Z}_t|$ matrix. Each value $\Phi_{j,l}$ such that $j \leq \left|\mathbf{M}^k\right|$, represents the probability that

Algorithm 4.2 Measurement Likelihood

1: **procedure** MEASUREMENT LIKELIHOOD($\hat{\mathbf{x}}_t^k, M^k, \mathbf{Z}_t$)
2: **for** each $\mathbf{m}_j^k \in \mathbf{M}^k$ **do**
3: $\hat{z}_j = h(\hat{\mathbf{x}}_t, \mathbf{m}_j^k)$ ▷ Predicted measurement
4: $H_{\mathbf{x}_t} = \nabla_{\mathbf{x}_t} h(\hat{\mathbf{x}}_t, \mathbf{m}_j^k)$ ▷ Measurement model Jacobian wrt. landmark
5: $H_{\mathbf{m}_j} = \nabla_{\mathbf{m}_j} h(\hat{\mathbf{x}}_t, \mathbf{m}_j^k)$ ▷ Measurement model Jacobian wrt. state
6: **for** each $\mathbf{z}_l \in \mathbf{Z}_t$ **do**
7: $Q_{j,l} = P_{\mathbf{z}_l} + H_{\mathbf{m}_j^k} \Sigma_{\mathbf{m}_j^k} H_{\mathbf{m}_j^k}^T$ ▷ Measurement innovation covariance
8: $\Sigma_{\mathbf{x}_t^k,j,l} = [H_{\mathbf{x}_t^k}^T Q_{j,l}^{-1} H_{\mathbf{x}_t^k} + R_t^{-1}]^{-1}$ ▷ Predicted landmark covariance
9: $\mu_{\mathbf{x}_t^k,j,l} = \Sigma_{\mathbf{x}_t^k,j,l} H_{\mathbf{x}_t^k}^T Q_{j,l}^{-1} (\mathbf{z}_j - \hat{z}_j) + \hat{\mathbf{x}}_t^k$ ▷ Predicted landmark mean
10: $\hat{z}_{j,l} = h(\mu_{\mathbf{x}_t^k,j,l}, \mathbf{m}_j^k)$ ▷ Updated measurement prediction
11: $\phi_{j,l} = \frac{1}{\sqrt{(2\pi)^d |Q_{j,l}|}} exp(-\frac{1}{2}(\mathbf{z}_l - \hat{z}_{j,l})^T Q_{j,l}^{-1} (\mathbf{z}_l - \hat{z}_{j,l}))$ ▷ Likelihood of the measurement $\hat{z}_{j,l}$
12: $\Phi_{j,l} = \phi_{j,l}^{w(\phi_{j,l})} \cdot \lambda_{j,l}$ ▷ Likelihood of the measurement descriptor $\lambda_{j,l}$
13: **end for**
14: **end for**
15: **end procedure**

the measurement $\mathbf{z}_l$ corresponds to the landmark $\mathbf{m}_j^k$. Each value $\Phi_{j,l}$ such that $j > \left|\mathbf{M}^k\right|$, represents the probability that the measurement $\mathbf{z}_l$ corresponds to a new landmark.

4.1.3 Pose update

Line 6 in Algorithm 4.1 updates the robot's pose based on the data association performed. A new pose for the robot is calculated by sampling a new value from the proposal distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{Z}_t, \mathbf{M})$ that considers both the motion of the robot $\mathbf{u}_t$ and all measurements $\mathbf{Z}_t$ taken at time t. The proposal distribution is modelled as a Gaussian with mean $\mu_{\mathbf{x}_t}$ and covariance $\Sigma_{\mathbf{x}_t}$. The proposal distribution is calculated successively using Eqns. 13-15 as outlined in the paper on FastSLAM[38] and reprinted within Algorithm 4.3 in lines 9-10. This iterative process continually shrinks the proposal distribution giving greater weight to landmark/measurement pairs that are processed first; thus, it is important that landmarks are processed in order of increasing uncertainty. An individual landmark's uncertainty is quantized using the volume of the ellipsoid described by the eigenvectors and eigenvalues of the landmark's covariance

Algorithm 4.3 Pose update algorithm

1: **procedure** PoseUpdate
2: **if** $\Sigma_{j=1}^{|\mathbf{M}^k|} == 0$ **then** ▷ If there were no known landmarks observed
3: $x_t^k \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ ▷ Calculated new location used the measurement model
4: **else**
5: $\Sigma_{\mathbf{x}_t,0} = R_t$ ▷ Initial state covariance
6: $\mu_{\mathbf{x}_t,0} = \hat{\mathbf{x}}_t$ ▷ Initial state mean
7: **for** each $\mathbf{m}_j \in \mathbf{M}^k$ **do**
8: **if** $\psi_j > 0$ **then**
9: $\Sigma_{\mathbf{x}_t,j} = [H_{\mathbf{m}_j}^T Q_{j,\psi_j}^{-1} H_{\mathbf{m}_j} \Sigma_{\mathbf{x}_t,j-1}^{-1}]^{-1}$ ▷ Updated prediction of state covariance
10: $\mu_{\mathbf{x}_t,j} = \mu_{\mathbf{x}_t,j-1} + \Sigma_{\mathbf{x}_t,j} H_{\mathbf{x}_t,j}^T Q_{j,\psi_j}^{-1}(z_{t,\psi_j} - \hat{z}_j)$ ▷ Updated prediction of state mean
11: **end if**
12: **end for**
13: $\Sigma_{\mathbf{x}_t} = \Sigma_{\mathbf{x}_t,j}$ ▷ Final prediction of state covariance
14: $\mu_{\mathbf{x}_t} = \mu_{\mathbf{x}_t,j}$ ▷ Final prediction of state mean
15: $x_t^k \sim N(\mu_{\mathbf{x}_t}, \Sigma_{\mathbf{x}_t})$ ▷ Sample new pose from the predicted distribution
16: **end if**
17: **end procedure**

matrix. If no measurements have been associated with a known landmark then one is calculated by directly sampling a pose from the motion models's probability distribution.

4.1.4 Map update

The state and covariance of each landmark is updated in lines 9-10 of Algorithm 4.4. The landmark update process calculates a new mean $\mu_{\mathbf{m}_j,t}$ and covariance $\Sigma_{\mathbf{m}_j,t}$ for all known landmarks with an associated measurement. Landmarks that have not been observed during this iteration of the algorithm have their liveliness $TTL_{\mathbf{m}_j,t}^k$ value reduced by 1. If a landmark's liveliness value falls to zero the landmark must meet two condition or it is deleted from the map. The first condition is that the landmark must have been observed at least $\beta_{\mathbf{z}_{t,l}}$ times, the second condition is that its certainty (not uncertainty) improve by $K_{certainty}$ (the

value of 60% has proven to be an effective threshold) over its initial value. The update process for an observed landmark the standard EKF update method from FastSLAM and FastSLAM 2.0. The first step in this EKF update process calculates a new predicted measurement $\bar{z}_j$ for each landmark $\mathbf{m}_j$ in using the robot's updated pose x_t^k. Then the Jacobian of the measurement model with respect to the landmark $\nabla_{\mathbf{m}_j} h$ is computed using the updated robot pose x_t^k. Next, a new measurement innovation covariance Q_j is calculated using the measurement noise $P_{\psi_j,t}$, the previous landmark covariance $\Sigma_{\mathbf{m}_j,t-1}$ and the measurement Jacobian $\nabla_{\mathbf{m}_j} h$. The Kalman gain, K, which represents the perceived confidence in the new measurement relative to the current estimate and can be calculated using: the measurement innovation covariance Q_j, the previous landmark covariance $\Sigma_{\mathbf{m}_j,t-1}$ and the measurement Jacobian $\nabla_{\mathbf{m}_j} h$. The Kalman gain is used to identify in what proportion should the difference between the actual measurement z_{t,ψ_j} and the predicted measurement $\bar{z}_j$ be added to landmarks mean $\mu_{\mathbf{m}_j,t}$. Next the previous landmark covariance $\Sigma_{\mathbf{m}_j,t-1}$ is scaled using the Kalman gain K and the measurement model jacobian $\nabla_{\mathbf{m}_j} h$ to obtain an updated covariance $\Sigma_{\mathbf{m}_j,t}$. Finally, the landmark $TTL^k_{\mathbf{m}_j,t-1}$ is multiplied by $\alpha_{\mathbf{z}_{t,\psi_j}}$.

At this point the contribution of all landmark/measurement pairs that contribute to the weight of the particle needs to be calculated. This is accomplished by separating landmarks into two categories. The first category contains landmarks that should contribute to the weight of the particle, the other contains landmarks that are not sufficiently reliable to contribute. Criteria must be established in order for a landmark/measurement pair to contribute to this updating processes. Typically, the most important criteria is that the landmark must been a permanent fixture of the map. This method prevents new and unreliable landmarks from having an effect of the particles weight possibly preventing the particle from progressing to the next iteration.

Once all existing landmarks have been updated, the remaining positive associations within ψ correspond to newly discovered landmarks within the environment. For each of these positive associations a new landmark $\mathbf{m}_i$ is created and added to the map and its sighting $i^k_{\mathbf{m}_i,t}$ and liveliness $TTL^k_{\mathbf{m}_i,t}$ values are set to 1 and $\gamma_{\mathbf{z}_{t,\psi_i}}$ respectively.

Algorithm 4.4 Map update algorithm

procedure MAPUPDATE
  $w^k = 1$
  for each $\mathbf{m}_j \in \mathbf{M}^k$ **do** ▷ update existing landmarks
    if $\psi_j > 0$ **then** ▷ update sighted landmark
      $\bar{z}_j = h(x_t^k, \mathbf{m}_j)$ ▷ predicted measurement
      $H_{\mathbf{m}_j} = \nabla_{\mathbf{m}_j} h(x_t^k, \mathbf{m}_j)$ ▷ measurement model Jacobian wrt. landmark
      $Q_j = P_{\psi_j,t} + H_{\mathbf{m}_j} \Sigma_{\mathbf{m}_j,t-1} H_{\mathbf{m}_j}^T$ ▷ measurement innovation covariance
      $K = \Sigma_{\mathbf{m}_j,t-1} H_{\mathbf{m}_j}^T Q_j^{-1}$ ▷ Kalman gain
      $\mu_{\mathbf{m}_j,t} = \mu_{\mathbf{m}_j,t-1} + K(z_{t,\psi_j} - \bar{z}_j)$ ▷ updated landmark position
      $\Sigma_{\mathbf{m}_j,t} = (I - K H_{\mathbf{m}_j}) \Sigma_{\mathbf{m}_j,t-1}$ ▷ updated landmark covariance
      if $i_{j,t}^k \geq \beta_{\mathbf{z}_{t,\psi_j}}$ **then**
        $H_{\mathbf{x}_t} = \nabla_{\mathbf{x}_t} h(\hat{\mathbf{x}}_t, \mathbf{m}_j)$ ▷ measurement model Jacobian wrt. state
        $L = H_{\mathbf{x}_t} R_t H_{\mathbf{x}_t}^T + Q_j$ ▷ predicted covariance of $\bar{z}_j$
        $\hat{w} = \frac{1}{\sqrt{(2\pi)^r |L|}} exp(-\frac{1}{2}(\mathbf{z}_l - \hat{z}_{j,l})^T L^{-1} (\mathbf{z}_l - \hat{z}_{j,l}))$ ▷ weighting of the measurement $\hat{z}_{j,l}$
      else
        $\hat{w} = 1$
      end if
      $i_{\mathbf{m}_j,t}^k = i_{\mathbf{m}_j,t-1}^k + 1$ ▷ update landmark's sightings
      $TTL_{\mathbf{m}_j,t}^k = TTL_{\mathbf{m}_j,t-1}^k * \alpha_{\mathbf{z}_{t,\psi_j}}$ ▷ update landmark's TTL
    else ▷ update unsighted landmark
      $TTL_{\mathbf{m}_j,t}^k = TTL_{\mathbf{m}_j,t-1}^k - 1$ ▷ update time to live
      if $TTL_{\mathbf{m}_j,t}^k = 0$ and $i_{\mathbf{m}_j,t}^k < \beta_{\mathbf{z}_{t,\psi_j}}$ and $vol(\Sigma_{\mathbf{m}_j,t}) > 0.4 vol(\Sigma_{\mathbf{m}_j,t_{\mathbf{m}_j}})$ **then**
        delete $\mathbf{m}_j$ ▷ delete landmark
      end if
      $\hat{w} = 1$
    end if
    $w^k = w^k \cdot \hat{w}$ ▷ update particle weight
  end for
  for $i = \left|\mathbf{M}^k\right|$, $i < |\psi|$, $i{+}{+}$ **do**
    if $\psi_i > 0$ **then** ▷ add new landmarks
      $\mathbf{m}_i = createLandmark(x_t^k, \mathbf{z}_{t,\psi_i})$ ▷ initialize landmark mean and covariance
      $i_{\mathbf{m}_i,t}^k = 1$ ▷ initialize landmark sight count
      $TTL_{\mathbf{m}_i,t}^k = \gamma_{\mathbf{z}_{t,\psi_i}}$ ▷ initialize landmark time to live
    end if
  end for
end procedure

4.2 Plant model

The Clearpath Kingfisher is modelled as a differential drive aquatic surface vehicle whose motion is controlled by the vehicle's linear velocity $\mathbf{v}_x$, it's angular velocity $\mathbf{v}_\theta$ and the time step delta dt. Using these parameters we can define the motion model $h(\mathbf{u}_t, \mathbf{x}_{t-1})$. The motion model computes the expected pose of the robot $\mathbf{x}_t$ given the previous pose $\mathbf{x}_{t-1}$ of the robot and a control vector $\mathbf{u}_t$.

$$h(\mathbf{u}_t, \mathbf{x}_{t-1}) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + dt * \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_\theta \end{bmatrix} \tag{4.1}$$

4.2.1 Error model

The plant model of the vehicle is corrupted by noise. This presents a significant problem as the errors in overall motion of the robot are not only proportional to the controls but also carry an overhead. Simply put, even with a lack of intended motion by the robot, over time, wind and wave action can move the robot from one pose to another. Obtaining an accurate measurement of the error or covariance in the control variables is essential to properly integrating motion commands into a SLAM algorithm. Underestimating this noise narrows the space in which measurements are deemed possible to be taken from, and this can result in the algorithm not finding a solution. Overestimating noise in the motion model expands the search space and increases the possibility some particles will stray from the true distribution. Increasing the number of particles tracked by the filter can compensate for this, at the expense of increased memory and computation resources.

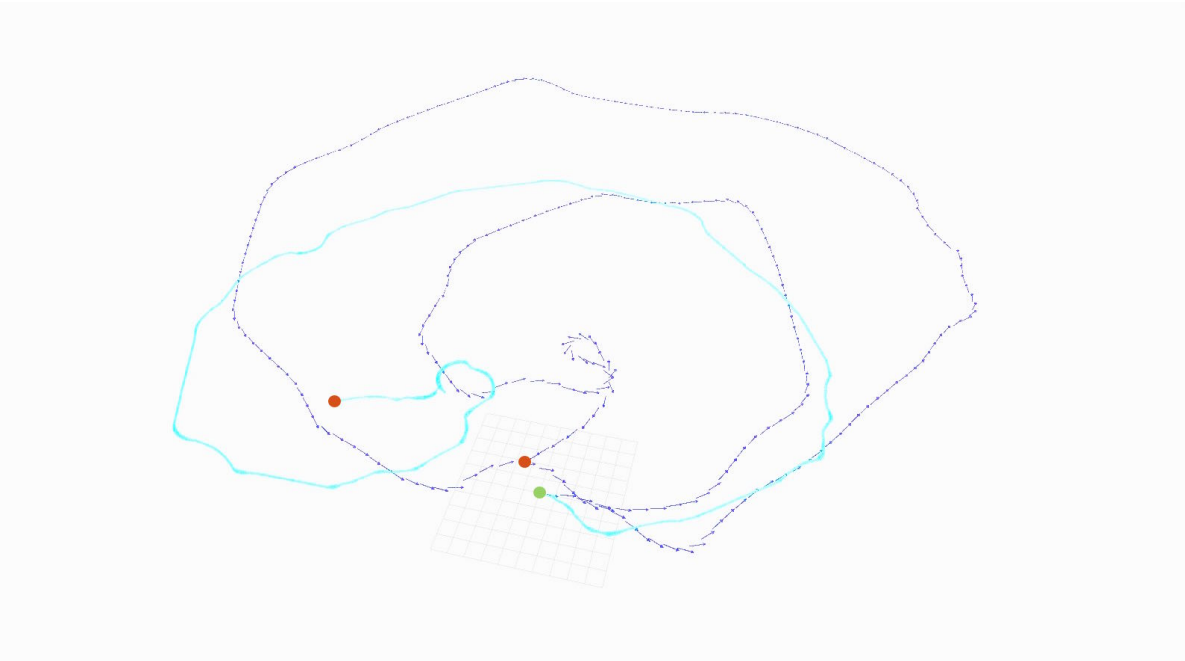


Figure 4.1: **Visualization of Robot Odometry and Ground Truth**
Computed pose of the robot using the above plant model (blue) composed with the computed ground truth odometry using onboard GPS (teal). Both tracks start at the common point in the centre of the superimposed grid marked in green. The end points of both tracks are marked with a red dot.

So how bad can we expect the plant noise process to be? In order to investigate this a control experiment was performed with the Kingfisher robot on Stong Pond at York University. The motion of the robot from a given starting pose was computed using the hand-tuned plant model and this was compared with estimates obtained using a differential GPS receiver and compass mounted on board the robot (see Figure 4.1). These sensors generate measurements at 1HZ, a much slower rate than odometry can be calculated on the robot. The plant model trajectory is shown in red. There is clearly a discrepancy between the two trajectories, and some mechanism is required to inject this noise into the state estimate measurement process. The standard method of modelling this error is state estimation algorithms involves modelling this error as an error proportional to the commanded motion. Unfortunately, such a model does not fully encapsulate the motion model error for an aquatic surface robot. Observe that in the traditional estimation approach, should no commanded motion be received, no error is introduced into the state estimate. A more sophisticated error model is required for autonomous surface vehicles that includes both a term proportional to the control inputs

and also includes a term to represent the stationary error. This approach is followed here. Specifically, it is assumed that the state is corrupted by the sum of two zero-mean Gaussian noise processes. One that is proportional to commanded motion, the other that is proportional to time spent stationary.

To calculate the stationary error of the robot there are two options. The first involves running a calibration step at the beginning of each set of data collection where the robot is commanded to remain stationary for a set amount of time and the actual motion of the robot is measured. This method has the benefit of more accurately incorporating the effects of current weather patterns on the robot. The second method involves conducting an independent test where the robot is left to float in the water while measuring the effects on the robot's motion and then using this estimate – properly adjusted for changes in the test weather and sea conditions relative to the calibration conditions. For the work described here, the first of these two approaches was followed. The second approach is an interesting direction for further research.

In order to compute the covariance Σ_h of the error in the odometry model the covariance $\Sigma_{\mathbf{u}}$ of error in control variable must first be estimated. This is done by calculating the control variable between matching time segments of both the ground truth odometry and the estimated odometry and calculating the covariance of the absolute difference over a large sample. This renders an error covariance $\Sigma_{\mathbf{u}}$ for the inputs of the odometry model. Using this covariance, the covariance Σ_h of the error in the odometry model can be calculated using the Jacobian of the odometry model (shown in Eqn. 4.2) via Eqn. 4.3.

$$\Delta H_{\mathbf{u}} = \begin{bmatrix} \frac{\partial x}{\partial \mathbf{v}_x} & \frac{\partial x}{\partial \mathbf{v}_y} & \frac{\partial x}{\partial \mathbf{v}_\theta} \\ \frac{\partial y}{\partial \mathbf{v}_x} & \frac{\partial y}{\partial \mathbf{v}_y} & \frac{\partial y}{\partial \mathbf{v}_\theta} \\ \frac{\partial \theta}{\partial \mathbf{v}_x} & \frac{\partial \theta}{\partial \mathbf{v}_y} & \frac{\partial \theta}{\partial \mathbf{v}_\theta} \end{bmatrix} = \begin{bmatrix} -dt * sin(\theta) & -dt * cos(\theta) & 0 \\ dt * cos(\theta) & -dt * sin(\theta) & 0 \\ 0 & 0 & dt \end{bmatrix} \tag{4.2}$$

$$\Sigma_h = \Delta H_{\mathbf{u}} \Sigma_{\mathbf{u}} \Delta H_{\mathbf{u}}^T \tag{4.3}$$

4.3 Measurement model

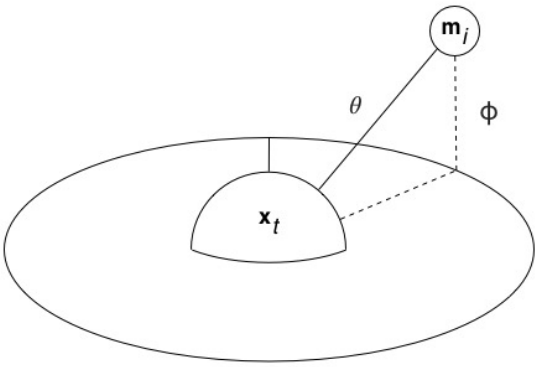
Incorporating measurements from a sensor into the FastSLAM 2.0 framework requires a number of mathematical definitions. Among these are how sensor measurements are generated with respect to the robot's state and known landmarks. This behaviour is more formally known as the sensor's measurement model. How the measurement model reacts to changes in the robot's state as well as changes in location of the landmark is also important to integrating new measurements into the FastSLAM framework. These are represented in terms of Jacobians of the measurement model.

4.3.1 Omnidirectional sensor

The Kodiak Pixpro SP360[50] omnidirectional camera onboard the robot produces a single 1024x1024 image at a rate of approximately $30Hz$. Multiple keypoints can be extracted from each image using an appropriate keypoint detection algorithm such as SIFT[60]. Using the Taylor camera model described is Subsection 3.3.1 each keypoint can be converted into a bearing and elevation measurement.

Bearing elevation measurements

The omnidirectional camera maps points from a 3-dimensional world frame into a 2-dimensional image frame (see Figure 4.2), this process strips away depth cues from the measurement. Without such information the function transforming points from the world frame to the image frame cannot be used to obtain an exact location of the landmark m_i. The basic sensor model for a measurement $\mathbf{m}_i$ and landmark $\mathbf{x}_t$ is given in Eqns. 4.4. This takes a landmark m and a known robot state x_t and computes the appropriate elevation and azimuth from these values. Let $r = \sqrt{(\mathbf{m}_{i_x} - \mathbf{x}_{t_x})^2 + (\mathbf{m}_{i_y} - \mathbf{x}_{t_y})^2}$ then



(a) *Diagram depicting the relevant variables in the camera's measurement model*

(b) *A single measurement/image retrieved from the camera showing the cropping of image by the CMOS sensor*

Figure 4.2: **Omnidirectional camera: model and measurement**

$$g(\mathbf{x}_t, \mathbf{m}_i) = \begin{bmatrix} \theta \\ \phi \end{bmatrix} = \begin{bmatrix} tan^{-1}(\frac{\mathbf{m}_{iy} - \mathbf{x}_{t_y}}{\mathbf{m}_{i_x} - \mathbf{x}_{t_x}}) - \mathbf{x}_{t_\theta} \\ tan^{-1}(\frac{\mathbf{m}_{iz}}{r}) \end{bmatrix} \tag{4.4}$$

Given these definitions, one can obtain the Jacobian relating changes in landmark position $\Delta G_{\mathbf{x}}$ and landmark position $\Delta G_{\mathbf{m}}$ as shown in Eqns. 4.5 and 4.6.

$$\Delta G_{\mathbf{x}} = \begin{bmatrix} \frac{\partial\theta}{\partial \mathbf{x}_{t_x}} & \frac{\partial\theta}{\partial \mathbf{x}_{t_y}} & \frac{\partial\theta}{\partial \mathbf{x}_{t_\theta}} \\ \frac{\partial\phi}{\partial \mathbf{x}_{t_x}} & \frac{\partial\phi}{\partial \mathbf{x}_{t_y}} & \frac{\partial\phi}{\partial \mathbf{x}_{t_\theta}} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{m}_{iy} - \mathbf{x}_{t_y}}{r^2} & \frac{\mathbf{x}_{tz} - \mathbf{m}_{i_x}}{r^2} & -1 \\ \frac{\mathbf{m}_{iz}(\mathbf{m}_{i_x} - \mathbf{x}_{t_x})}{r^3(1+\frac{\mathbf{m}_{iz}^2}{r^2})} & \frac{\mathbf{m}_{iz}(\mathbf{m}_{iy} - \mathbf{x}_{t_y})}{r^3(1+\frac{\mathbf{m}_{iz}^2}{r^2})} & \frac{r}{r^2+\mathbf{m}_{iz}^2} \end{bmatrix} \tag{4.5}$$

$$\Delta G_{\mathbf{m}} = \begin{bmatrix} \frac{\partial\theta}{\partial \mathbf{m}_{i_x}} & \frac{\partial\theta}{\partial \mathbf{m}_{i_y}} & \frac{\partial\theta}{\partial \mathbf{m}_{i_z}} \\ \frac{\partial\phi}{\partial \mathbf{m}_{i_x}} & \frac{\partial\phi}{\partial \mathbf{m}_{i_y}} & \frac{\partial\phi}{\partial \mathbf{m}_{i_z}} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{x}_{t_y} - \mathbf{m}_{iy}}{r^2} & \frac{\mathbf{m}_{i_x} - \mathbf{x}_{t_x}}{r^2} & 0 \\ -\frac{\mathbf{m}_{iz}(\mathbf{m}_{i_x} - \mathbf{x}_{t_x})}{r^3(1+\frac{\mathbf{m}_{iz}^2}{r^2})} & -\frac{\mathbf{m}_{iz}(\mathbf{m}_{iy} - \mathbf{x}_{t_y})}{r^3(1+\frac{\mathbf{m}_{iz}^2}{r^2})} & \frac{r}{r^2+\mathbf{m}_{iz}^2} \end{bmatrix} \tag{4.6}$$

Error model The error associated with respect to the bearing and elevation measurements produced by this sensor can be approximated by the Taylor camera model described previously in Section 3.3.1. Using

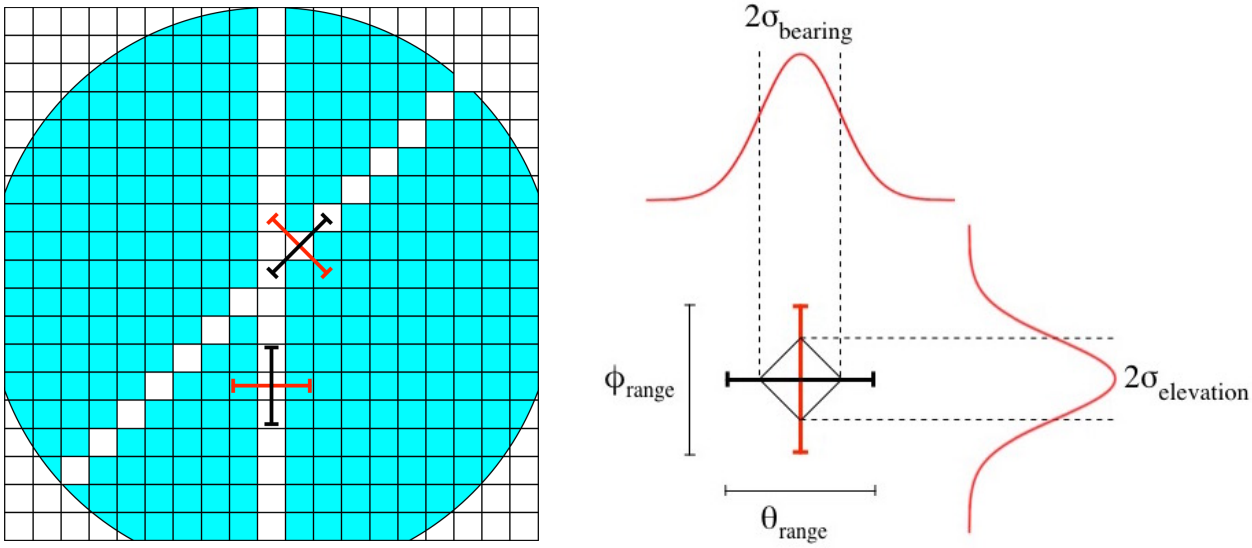
the calibration parameters obtained for the camera model, each pixel (u_x, u_y) in the image is back-projected onto the unit sphere. Eqn. 4.7 describes the relationship between the bearing and elevation (θ, ϕ) associated with a point (x, y, z) on the unit sphere.

$$\begin{bmatrix} \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \tan^{-1}(y/x) \\ \sin^{-1}(z) \end{bmatrix} \tag{4.7}$$

Determining the covariance matrix that describes the error associated with the bearing and elevation measurements can be simplified somewhat by observing that the Taylor camera model assumes that the refractive and reflective properties of the camera's optics are radially symmetrical. Using this model the bearing-elevation error associated with any given pixel is dependant only on the pixel's elevation. As these values will be used repeatedly a lookup-table could be used to store the covariance matrix of all possible values indexed by elevation. Unfortunately, the assumption of radial symmetry is slightly invalidated when considering the discretization and misalignment of the sensor plane. To accommodate this minor asymmetry the simple lookup table is generalized to represent the covariance matrix from different bearings. The covariance matrix for pixels not in the lookup table can be interpolated using the nearest covariance from each opposing axis.

Although not a perfect solution this approximation is preferable to the alternative of creating a lookup-table for every possible bearing and elevation. The pixels situated along the image diagonal and vertical are ideal candidates for this lookup table because they cover the largest range of both bearing and elevation due to their orientation with respect to the polar coordinate system.

Landmark initialization and update



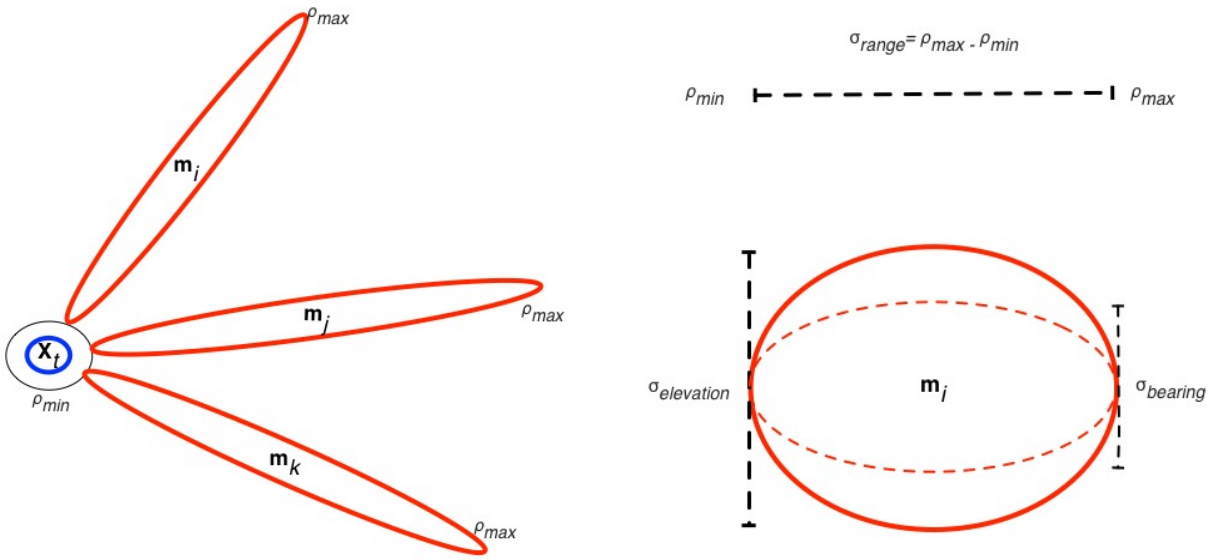
(a) *diagram depicting the pixels in the lookup table as well as the pixels involved in calculating the uncertainty.*

(b) *shows how a pixel's covariance is determined using the range of both bearing and elevation calculated using diagonally adjacent pixels.*

Figure 4.3: **Pixel covariance estimation**

Given the location and orientation of the camera and a single bearing measurement, the 3D location of a landmark cannot be fully determined. The problem of initializing new landmarks from 'bearing-only' data is a common problem with visual SLAM and is widely addressed in the literature and many solutions have arisen. These solutions can be categorized broadly into two groups: delayed initialization (see, e.g., [59]) and un-delayed initialization (see, e.g., [61][62][63][64]). Delayed approaches keep track of bearing measurements of a single landmark and these bearing measurements are aggregated over small motions of the robot until a realistic estimate of the full state of the landmark can be obtained. Determining the criteria for landmark initialization is complex and many solutions can be described as ad hoc.

Un-delayed initialization approaches take a somewhat different tack. Rather than waiting until enough bearing measurements have been taken to obtain a good estimate of the full state of the landmark, initialization proceeds immediately using only the bearing data. In the un-delayed approach a new landmark is initialized immediately at some distance ρ_c from the robot. The uncertainty of the distance of the landmark from the robot is set so that the uncertainty covers the entire distance range from ρ_{max} to ρ_{min}, while the



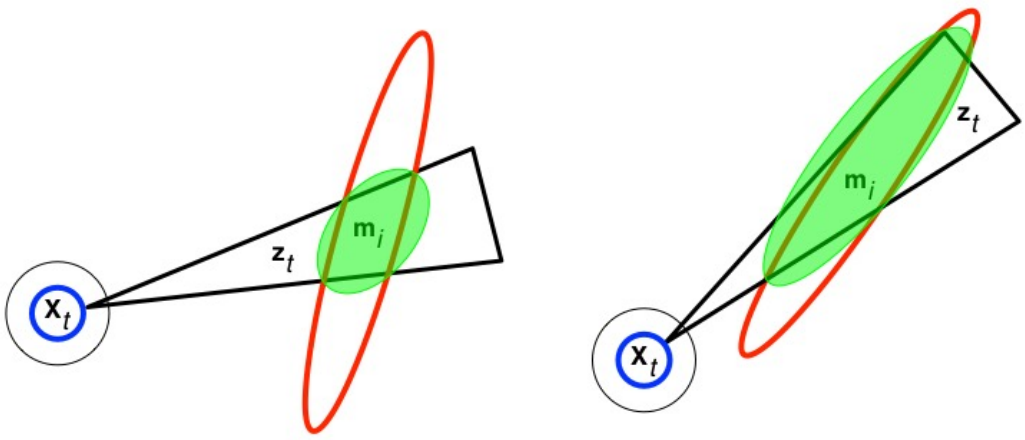
(a) *2-dimensional diagram depicting multiple landmarks being initialized from a single camera image containing multiple bearing and elevation measurements.*

(b) *2-dimensional representation of a 3-dimensional ellipsoid(not to scale) that depicts the initial error of a new landmark. The variance associated with the range is huge in comparison to all other variables.*

Figure 4.4: **Initialization of landmark covariance**
Initialization of landmark covariance. This figure depicts graphically the initialization of both the position and covariance of the landmark. (a) shows the 2-dimensional case and (b) depicted the 3-dimensional case.

variance in bearing and elevation is set from the known sensor error properties. A covariance representation in Cartesian space is then constructed from these values.

One problem with the immediate initialization of landmarks in this manner is that it can introduce spurious, or at least very noisy landmarks into the map. Rather than entering landmarks into the map immediately, a common approach is the use of a meta phase during which landmark positions as additional measurements are made, but the landmark is not inserted into the map immediately. A common mechanism for performing this update is through the EKF update process [61][62] shown in Figure 4.4b. Figure 4.4 illustrates the process of landmark initialization. In Figure 4.4a the process is shown in Cartesian space with three new landmarks being initialized at a given bearing and elevation with an assumed target distance. The uncertainty in range is set to encompass all likely target distances while the uncertainty in bearing and elevation is set from the predicted uncertainty in these values. Figure 4.4b illustrates the error ellipsoids.



(a) *Ideal angular separation between measurements, reducing uncertainty of a landmarks position.*

(b) *Inadequate angular separating between measurements, resulting in a minimal or increased uncertainty of a landmarks position.*

Figure 4.5: **Landmark update**
The local motion of the robot causes updates to the landmark estimate as shown above. Ideally the robot moves in such a way to reduce the error ellipse as shown in (a). Motion orthogonal to the bearing results in very small changes in the error ellipse as shown in (b).

The initial landmark uncertainty is represented with a single covariance matrix and uses a standard EKF style landmark update procedure (shown in Figure 4.5). The Gaussian Sum Filter [63][64] method represents the initial landmark uncertainty using a Gaussian sum filter. Landmarks go through a meta phase where they receive measurement updates but are not included in the map. Upon exiting this meta phase the distribution with the highest likelihood is selected and added to the map. A comprehensive evaluation of these methods and their use in different SLAM algorithms is presented in [65]. This work shows that the EKF landmark initialization approach is prone to error in environments with a high landmark density, a problem that we are unlikely to encounter given the nature of the SIFT descriptors associated with each landmark (see below).

This research uses the un-delayed approach and avoids the density issue by including an additional likelihood ratio that is able to differentiate between locally similar measurements using its visual description.

Landmark association FastSLAM 2.0[38] provides a framework that incorporates per particle data association. Rather than using the entire image or image patches as potential features, here we follow a similar approach to that of [59] and reduce the image into a smaller number of potential feature points. Many such feature selection algorithms exist (see[66][67] for a review). The SIFT feature detector is used in this work. SIFT features were chosen because of their stability relatively to other popular image features. SIFT[60] features provide a 128 dimensional descriptor that describes their local neighbourhood in a scale invariant fashion which simplifies feature-to-feature matching. The recommended way for comparing SIFT features is through the use of the distance ratio of the best match against the next best match. This ratio should be less than 0.6-0.75 of the distance to the closest keypoint in order to signify a match [60]. One issue with using this approach in our application is that there exist a large number of visual landmarks that can be discounted using their estimated location within the environment. Although this simplifies the problem of matching features, the number of visual landmarks that remain within the possible neighbourhood of an incoming measurement can then become too few for a proper ratio test. Performing a ratio test against the entire set of landmarks introduces the possibility that a similar feature elsewhere in the environment may interfere with the results. A solution to this problem is to devise a mapping function that maps the distance between two features in SIFT space to the likelihood of these two features representing a correct visual match. Note that this mapping will be environment dependent, and it will be necessary to adapt this mapping for different operating environments.

When matching SIFT features there is a need to map from SIFT feature descriptor value distance to match likelihood. In the basic SIFT implementation this is done by seeking local minima that are substantively lower than other SIFT values in some local neighbourhood. Lacking such dense local features, it is necessary to estimate this mapping more directly. Here we sample typical images from the sensor for a given environment and construct our own model of SIFT feature distance and match probability. In order collect the amount data necessary to properly estimate a mapping function a custom application was created to facilitate the process of manually matching SIFT features between pairs of images. On startup a ratio test is performed

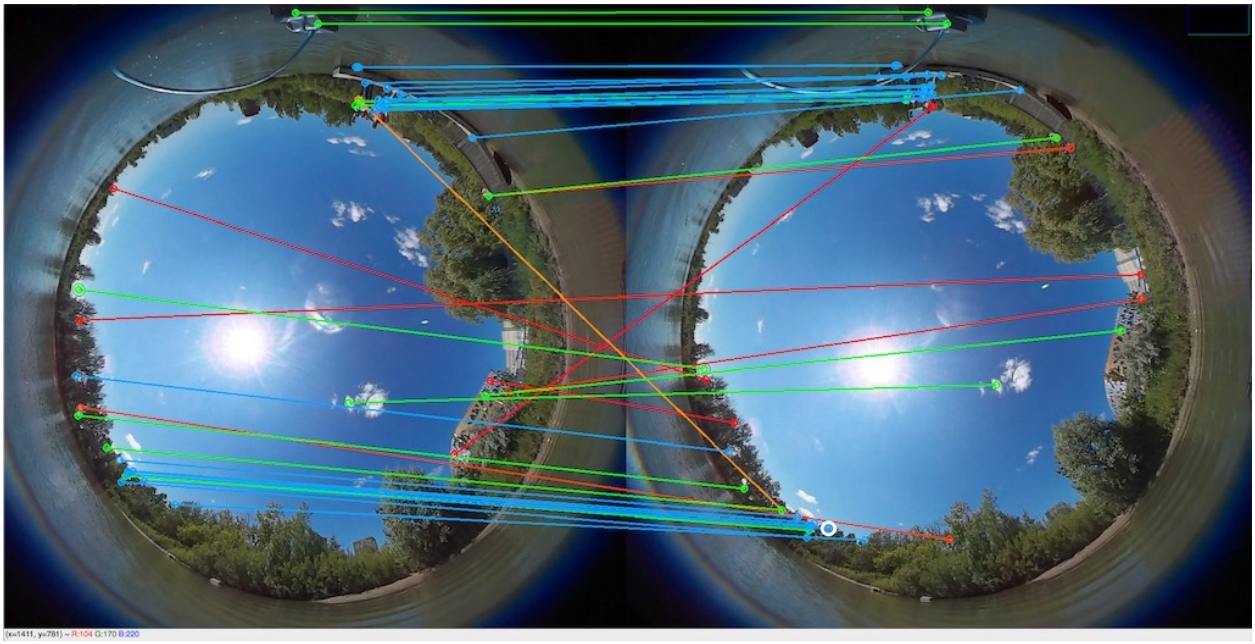


Figure 4.6: **SIFT feature matching application**
Matches coloured in green were labeled as correct by the user, Matches coloured in blue are possible matches that have yet to be labeled as correct or incorrect. Incorrect matches are coloured on a scale from yellow to red based on ratio on their difference with respect to the distance between the largest correct match.

on the key-points between both images to identify the first set of possible matches. The user can either select existing matching or create a new one and then delete them or classify them as either incorrect (yellow-red) or correct (green). A step in this process is shown in Figure 4.7. After at least one correct match has been identified a set of possible matches is generated automatically. This set is a globally optimum matching of features that are less than the maximum distance between all positive matches. When all the correct matches have been identified remaining key-point pairings are labelled as incorrect and a list of all matches both positive and negative and their distance are saved.

The Likelihood function shown in Fig. 4.7 was created using data collected from 18 image pairs across three different scenes. In total there were 191,873 potential matches, 629 of those matches were labelled as being correct, the other 191,244 were incorrect matches. The raw likelihood data was calculated by counting the number of correct and incorrect matches within ranges of width 20 units in SIFT space to determine the probability of a match. There is a sudden dip in likelihood values between (120,180]. This is possibly due to

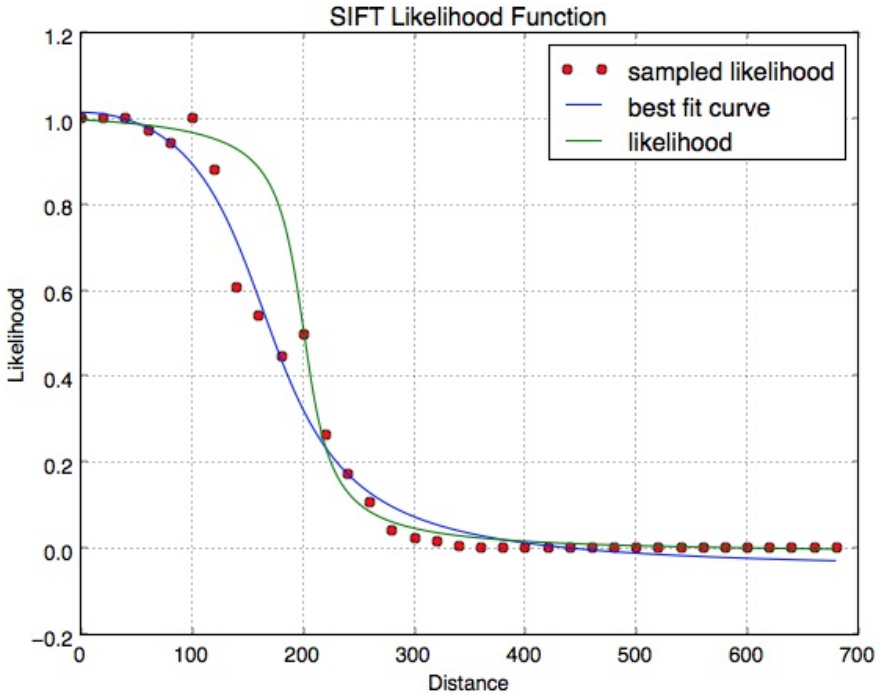
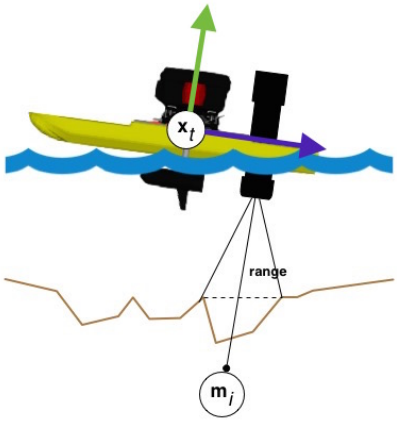


Figure 4.7: **SIFT likelihood function**
The red data points represents the ratio of correct matches to all N matches within a given range where N is sufficiently large. This ratio corresponds to the likelihood of a match within that range. The function drawn in blue is the curve that best fits the available data. The function drawn in green is the finalized likelihood function.

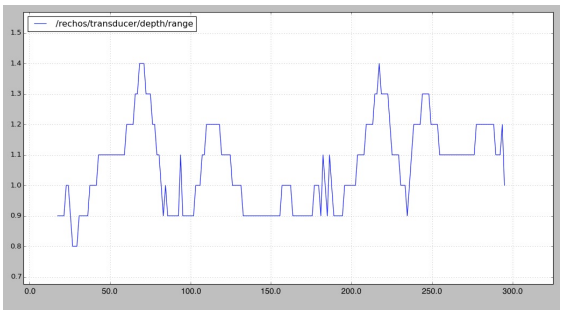
a lack of data and a higher number of false-negatives which can be attributed to the difficulty of identifying matches with such dissimilarity in their descriptors. This hypothesis was factored into the final likelihood so at to not disqualify this possibility. An arctangent function was fit to the data by finding the best-fit polynomial to the data and then using this fit to seed the non-linear arctangent fit (Figure 4.7).

4.3.2 SONAR sensor

The RECHOS depth sensor [45] produces a single range measurement (see Figure 4.8) at a rate of 1Hz. This range measurement is dependant on the distance from the landmark $\mathbf{m}_i$ to the location (x, y, z) of the depth sensor, which is fixed relative to the state of the robot.



(a) *Diagram depicting the relevant variables in the camera's measurement model*



(b) *A series of depth measurements.*

Figure 4.8: **Echo sounder: model and measurements**
The RECHOS sounder is mounted on the Kingfisher as illustrated in (a). (b) shows a typical sensor run as the robot follows a path through its environment.

Range measurements

Whereas distinct visual features can be identified using some appropriate feature descriptor such as SIFT given the continuous nature of the visual sensor the same is not true for the depth sensor. Landmarks discovered by this sensor have no inherit descriptor and can only be identified by their location within the environment. Here measurement density is quite low and an alternative approach is desirable.

Before considering this challenge; however, it is important to observe that the measurement (Figure 4.8b) returned from the depth sensor is impacted by its placement on the robot (its relative position and orientation) as well as the pitch, roll and yaw of the vehicle itself. Eqn. 4.8 below relates the returned depth measurement to the robot's state, assuming that the relative (x, y, z) offset (a, b, c) has already been corrected. Note that this analysis incorporates vessel roll, pitch and yaw.

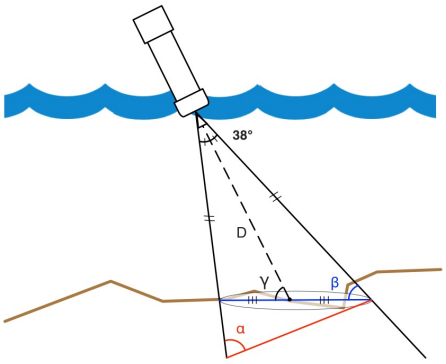
$$g(\mathbf{x}_t, \mathbf{m}_i) = \left[r\right] = \sqrt{\begin{array}{l}(\mathbf{m}_{i_x} - \mathbf{x}_x - acos(\mathbf{x}_\theta) + asin(\mathbf{x}_\theta))^2 \quad + \\ (\mathbf{m}_{i_y} - \mathbf{x}_y - bcos(\mathbf{x}_\theta) - bsin(\mathbf{x}_\theta))^2 \quad + \\ (\mathbf{m}_z - c)^2\end{array}} \tag{4.8}$$

As with the omnidirectional camera model, this model can be used to compute explicit forms for the Jacobian of the measurement in terms of changes in robot state (4.9) and landmark (4.10).

$$\Delta G_{\mathbf{x}} = \begin{bmatrix}\frac{\partial r}{\partial \mathbf{x}_{t_x}} \\ \frac{\partial r}{\partial \mathbf{x}_{t_y}} \\ \frac{\partial r}{\partial \mathbf{x}_{t_\theta}}\end{bmatrix}^T = \begin{bmatrix}\frac{\mathbf{m}_{i_x} - \mathbf{x}_{t_x} - acos(\mathbf{x}_{t_\theta}) + asin(\mathbf{x}_{t_\theta})}{r} \\ \frac{\mathbf{m}_{i_y} - \mathbf{x}_{t_y} - bcos(\mathbf{x}_{t_\theta}) - bsin(\mathbf{x}_{t_\theta})}{r} \\ \frac{(b^2 - a^2)cos(2\mathbf{x}_{t_\theta})}{r} \quad + \\ \frac{sin(\mathbf{x}_{t_\theta})(a\mathbf{m}_{i_x} - a\mathbf{x}_{t_x} + b\mathbf{m}_{i_y} - b\mathbf{x}_{t_y})}{r} \quad + \\ \frac{cos(\mathbf{x}_{t_\theta})(a(\mathbf{m}_{t_x} - \mathbf{x}_{t_x}) + b(\mathbf{m}_{t_y} - \mathbf{x}_{t_y}))}{r}\end{bmatrix}^T \tag{4.9}$$

$$\Delta G_{\mathbf{m}} = \begin{bmatrix}\frac{\partial r}{\partial \mathbf{m}_{i_x}} \\ \frac{\partial r}{\partial \mathbf{m}_{i_y}} \\ \frac{\partial r}{\partial \mathbf{m}_{i_z}}\end{bmatrix}^T = \begin{bmatrix}\frac{-asin(\mathbf{x}_{t_\theta}) - acos(\mathbf{x}_{t_\theta}) + \mathbf{m}_{i_x} - \mathbf{x}_{t_x}}{r} \\ \frac{-bsin(\mathbf{x}_{t_\theta}) - bcos(\mathbf{x}_{t_\theta}) + \mathbf{m}_{i_y} - \mathbf{x}_{t_y}}{r} \\ \frac{\mathbf{m}_{i_z} - c}{r}\end{bmatrix}^T \tag{4.10}$$

Error model Creating an error model for the CruzPro ATU120AT[46] is difficult because the sensor does not provide open access to raw sensor data. Instead measurements from each pulse (beam width 38°) are passed through an exponential filter which is a type of infinite impulse response (IIR) filter within the sensor returning only a single value. This value is an average of the depth measurements sampled from the intersection of the beam and the lakebed. The area of the intersection is best represented by a conic section, in this case an ellipse. The specifics of this exponential filter (shown in Eqn. 4.12) were provided by CruzPro upon request. This equation is a simple exponential filter which is a form of low-pass filter. This filter gives



(a) *geometry of the intersection of the sensor beam with the lakebed.*

$$
\begin{aligned}
\theta &= \cos^{-1}\left(\frac{\vec{D}\cdot proj_{xy}(\vec{D})}{\|\vec{D}\|\|proj_{xy}(\vec{D})\|}\right) \\
1 &= \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 \qquad \text{(ellipse)} \\
e &= \frac{\sin(\beta)}{\sin(\alpha)} = \sqrt{a^2+b^2} \qquad \text{(eccentricity)} \\
a &= \frac{\|\vec{D}\|\cdot\sin\left(\frac{19\pi}{180}\right)}{\sin\left(\frac{161\pi}{180}-\Gamma\right)} \qquad \text{(sine law)} \\
b &= \sqrt{e^2-a^2} \\
A &= \pi \times a \times b
\end{aligned}
\tag{4.11}
$$

(b) *Calculating the area of the beam's intersection with the lake bed.*

Figure 4.9: **SONAR sampling area**

The intersection of the sensor's beam with the lakebed can be approximated by an ellipse that intersects the cone representing the sensor beam. The geometry of this intersection is depicted in (a) and the Eqns. 4.11 in (b) detail the math needed to solve for the area of the sampling ellipse.

a slight priority to shallow depths due to the time it takes for the sonar pulse to reach the bottom and return back to the top.

$$
filtered\_depth_{n+1} = \lambda \times filtered\_depth_n + (1-\lambda)\times depth, \quad \lambda = \frac{6}{7} \tag{4.12}
$$

Due to the effects of this filter any error model associated with this sensor in fact represents the assumed variability in a patch of terrain. This is because the raw measurement error can be assumed to be zero mean Gaussian white noise. Measurement noise is filtered out by the sensor and any variability indicates a change in the position or orientation of the sensor. In keeping with this assessment an assumption about the smoothness/variability of a patch ($1m^2$) of lakebed must be made. The assumption made by this error model assumes the lakebed does not have any drastic changes in topography. Under this assumption a patch variability, $\sigma^2_{patch} = 0.04$ was chosen. Using this information combined with the set of equations shown in Eqn. 4.11 an error model/covariance can be constructed and rotated into the robots frame of reference (see

Eqn. 4.13).

$$\Sigma = R_z(\theta) \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & A\sigma^2_{patch} \end{bmatrix} R_z(\theta)^T \tag{4.13}$$

Landmark initialization and update Range measurements from the onboard SONAR sensor can be used to fully determine the location of any sensed landmark given that the position and orientation of the sensor is fixed with respect to the robot. To determine the covariance matrix associated with a new landmark the measurement variance σ_{depth} must be combined with a positional covariance. The positional covariance can be estimated using the intersection of the beam with the lake bed. If this ellipse is regarded as the 95% confidence interval of the positional covariance then the length of the semi-major and semi-minor axes can be used to solve for the eigenvalues of the covariance [68]. The eigenvectors of the covariance can be calculated using the projection of range vector on to the xy-plane.

$$\lambda_1 = (\frac{a}{2\sqrt{5.991}})^2, \quad \lambda_2 = (\frac{b}{2\sqrt{5.991}})^2 \tag{4.14}$$

The landmark update process is the same as that used in previous FastSLAM 2.0 implementations and that used for visual landmarks previously described in this chapter.

4.3.3 Compass sensor

The Kingfisher M100 includes an onboard tilt-compensated compass that provides orientation data relative to magnetic north at approximately $40Hz$. Compass measurements are often corrupted by nearby electromagnetic interference(EMI) from ferrous metals or operating electronic equipment. Simple calibration

methods can be used to compensate for minor interference sources, but recalibration is needed when new sources of EMI are introduced into the surrounding area. Shielding techniques can be used to cancel out minor sources of interference but cannot shield against larger sources without affecting overall performance.

Orientation measurements

Orientation measurement can be interpreted in two different ways. The first method treats each measurement as if it is observing a landmark within the environment. This landmark is the location of the magnetic north pole relative to the map frame of reference. The caveat with this approach is that the pole is so far away that even significant lateral motion would not result in a significant change in orientation. This makes estimating the location of magnetic north practically infeasible. The second method views these measurements as observing an unobservable landmark, thus they are not dependant on the known map. This motivates the need for a new compass specific measurement mode that reflects this relationship (shown in Equation. 4.15).

$$p(\tilde{\mathbf{z}}_t|\mathbf{x}_t) = f(\mathbf{x}_t) + \zeta_t \tag{4.15}$$

This model is only valid in the FastSLAM framework because the pose of the robot is viewed as an oracle representing the true pose of the robot. With this in mind the measured orientation only differs from the true orientation of the robot by zero mean Gaussian white noise error.

Error model The mean and variance of the error ζ_t associated with the compass can be measured statically by logging continuous measurements from the sensor while the robot remains stationary. This is a simple estimate of the error but neglects the changes in the EMI pattern due to changes in the rotational speeds of the propellers. This is negligible but may introduce a bias meaning that the error may no longer be zero-mean.

4.4 Algorithm walkthrough

The following chapter provides experimental validation of the SLAM algorithm along with comparisons of different sensor groups on overall algorithm performance. Here we provide a walkthrough of the algorithm in order to illustrate the various aspects of the algorithm and their integration. Each iteration of this modified FastSLAM 2.0 algorithm is agnostic to the mixture of measurements types (visual, SONAR, others) that are captured at each time step. Although due to the timing of incoming measurements from the various sensors, each sensor class is processed separately. This section will provide a step-by-step walkthrough of the algorithm using measurement from the SONAR sensor and measurements from the omnidirectional camera.

4.4.1 Visual measurement walkthrough

The number of SIFT features that can be extracted from a single image numbers in the hundreds and beyond. A large number of these features are weak and will likely not reappear in later images of the environment. Matching a large number of measurements against a large number of landmarks becomes computationally inefficient when the $O(n^3)$ complexity of the Hungarian algorithm is considered. This motivates the need for a solution which limits the number of SIFT features collected at any given time step while still select features that persist over time. This process selects a large number of initial keypoints from the current image and tries to match these with those selected in the previously processed image, the set of keypoints that cannot be matched are then sorted by their response values which indicates the feature's strength. All matched keypoints are selected as well as the N(typically 20) unmatched keypoints with the highest response. The reason for selecting unmatched features is so that new environmental features can be introduced into the map so that they may be re-measured at a later time. This process is able to consistently lower the number of visual features down to a manageable quantity.

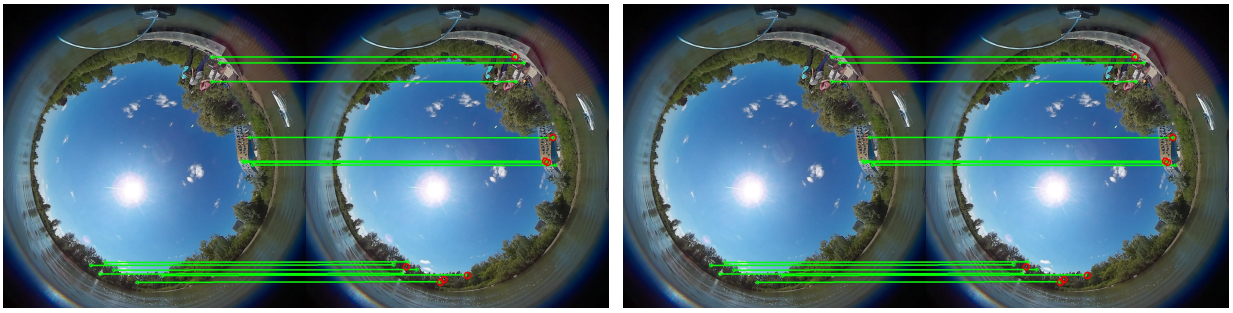
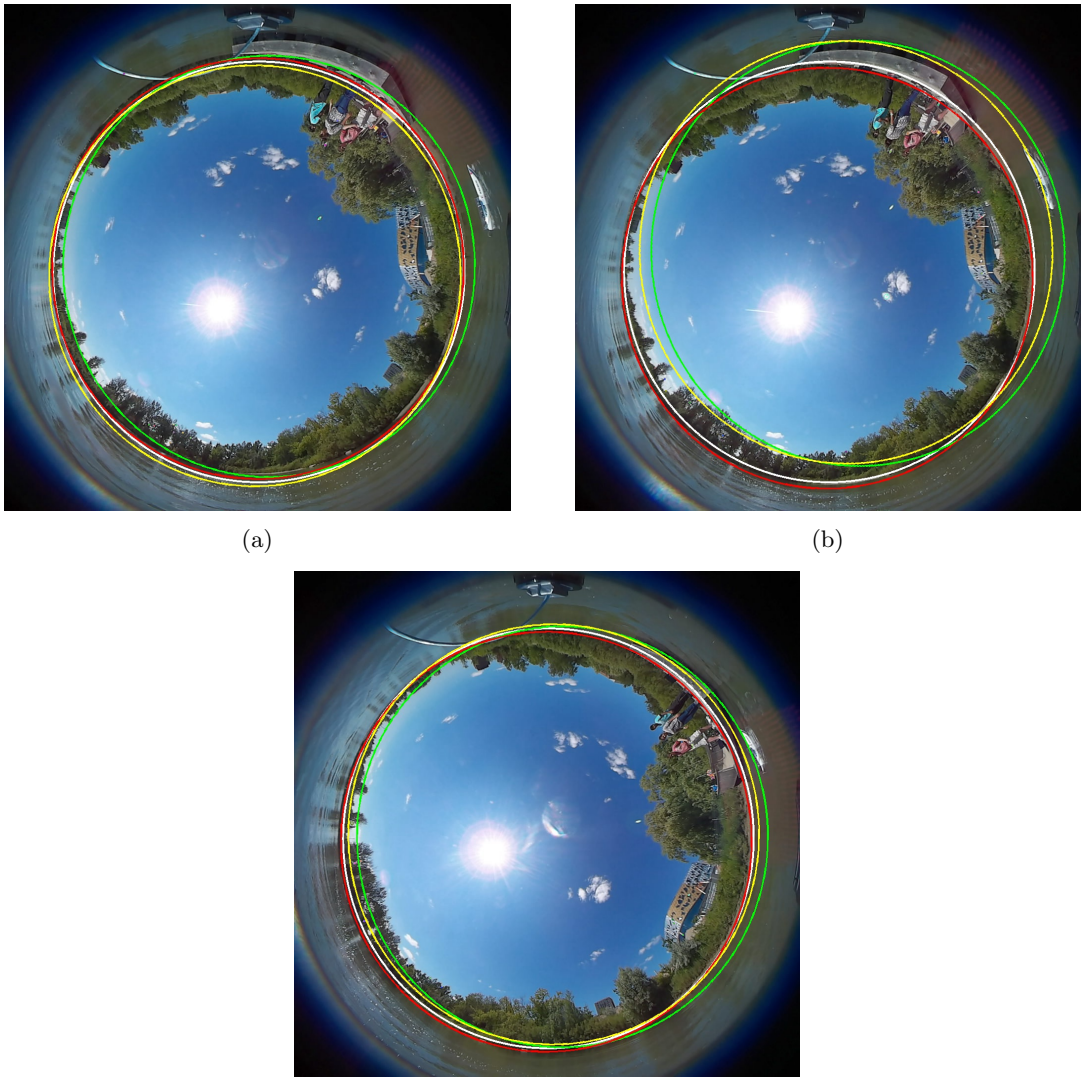


Figure 4.10: **Feature selection**
For every incoming image only matching features from the of the best map are chosen, those that match features in the previous image (green) and a small set of new features (red) are selected to become measurements in the SLAM process. This helps to decrease the number of measurements while still choosing measurement that are likely to persist overtime. For a given omnidirectional view, approximately 10-25 features are maintained.

The FastSLAM 2.0 implementation presented here does not account for the uncontrollable degrees of freedom of the robot namely its roll, pitch and heave. There are number of different methods that can be used to estimate the robot's roll and pitch. Perhaps the most straightforward is to use an on-board IMU to obtain an instantaneous estimate of the vehicle's roll/pitch state. Unfortunately, this state estimation process is limited by filtering within the IMU which integrates estimates over time in order to determine gravity and remove gravity' effect from the accelerometer. This delay can introduce errors into the estimation process, especially in the presence of waves. An alternative approach is to use the visual scene to estimate instantaneous pitch and roll. The two sets of matched keypoints from each image are back-projected onto the unit sphere rendering two 3D point clouds related by an affine transformation under the assumption of a static scene and no heave. This affine transformation can be estimated by use of the RANSAC[69] algorithm. To improve the accuracy of the method multiple previous images are used to create a collection of relative motion estimates. Using this collection of estimates any outliers are discarded and new mean is calculated for use as the final motion estimate. The relative performance of the three explored methods are shown in Fig 4.11 in contrast to the absence of motion estimation. Once the instantaneous pitch and roll of the vehicle is determined the elevation of each incoming measurement can be adjusted to account for the the vehicle's tilt. Estimating the robot's tilt using image features was the most stable and accurate of the methods shown in Fig. 4.11 and is the method utilized in this work. This method is not without its drawbacks, as it drastically



(a) (b)

(c)

Figure 4.11: **Tilt estimation**
Various methods used to estimate the roll and pitch of the vehicle, represented by their zero elevation line. Camera model static motion (white). Tilt estimated using accelerometer (green). Tilt estimation using gyroscope (yellow). Tilt estimation using image features (red).

increases the computational time required to compute the roll and tile of the robot.

Control values from the odometry model are not obtained synchronously with new measurements from the camera. To account for this asynchronicity the ability to compute a control value for any point in time is necessary. In order to compute the control at a given time two other values are needed depending on the

situation. If the queried time falls between two known states then the control value can be interpolated from these states, If the queried time falls after the last known state then the control value can be extrapolated using the last known control.

State update Once a control and set of image measurements are obtained they can be fed into the FastSLAM 2.0 algorithm. The first step in each iteration of the algorithm calculates a new hypothesized state for each particle in the filter using the given control as described in Eqn. 4.1.

Measurement likelihood Using this new state each measurement is compared to all existing landmarks, generating a probability that the landmark could produce such a measurement in terms of proximity for each comparison. Each probability is then augmented by the likelihood that such a measurement corresponds to the landmark in terms of likeness (SIFT likelihood). This $\left|\mathbf{M}^k\right| \times |\mathbf{Z}_t|$ matrix is extended by a $|\mathbf{Z}_t| \times |\mathbf{Z}_t|$ identity matrix multiplied by the probability of observing a new landmark. This new matrix is used so that incoming measurements can be assigned a probability representing whether it corresponds to a new landmarks.

Data association Given this probability matrix where each column i carries the probability that a particular measurement $\mathbf{z}_{i,t}$ corresponds to the each landmark $\mathbf{m}_j$ the Hungarian algorithm can be applied to this matrix to determine a globally optimal assignment of measurements to landmarks. Any measurement $\mathbf{z}_{i,t}$ assigned to a landmark $\mathbf{m}_j$ where $j \geq \left|\mathbf{M}^k\right|$ represents the sighting of a new landmark.

Robot pose update Using new incoming measurements of existing landmarks a newly proposed state of the robot and covariance (proposal distribution) is estimated. These landmark/measurement pairs help to direct and shape the proposal distribution to more accurately represent the robot‘s motion within the environment as described by these new measurements. A newly hypothesized state is randomly sampled

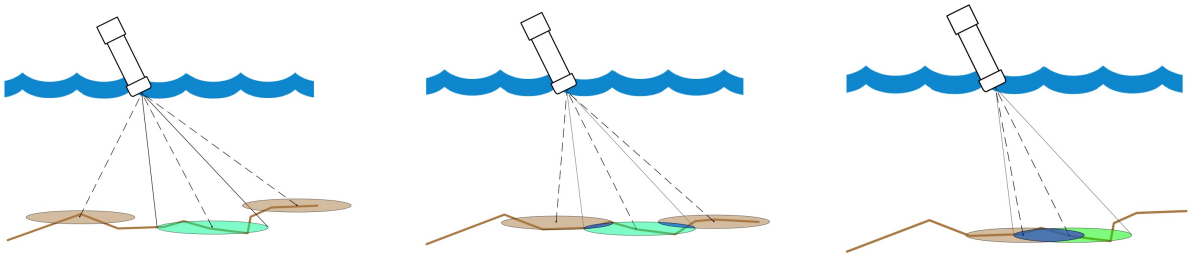
from this proposal distribution to introduce some variability into this model so that each particle ends up exploring a different path.

Map update With this new hypothesized state and the set of landmark/measurement associations the map is updated. The map update process can be conceptually divided in to three stages. The first stage deals with landmarks that have not been observed within this time step. These landmarks are divided into two categories; those that are permanent fixtures of the environment, and landmarks that are not permanent fixtures of the environment and thus need their time to live counter decreased. Landmarks are deleted from the map if this counter falls to zero. The second stage updates sighted landmarks using the standard EKF update process and has their time to live counter and sightings counter updated. Measurements that have not been associated with an existing landmark are used to initialize a new landmark that is added to the map, this includes an initial time to live counter.

Resampling The previous five paragraphs have outlined the steps taken for each of k particles within the Rao-Blackwellized particle filter used in this FastSLAM 2.0 variant. Once each particle and its weighting has been updated, the weightings across all particles are then normalized. Then a resampling step takes place that chooses a new set of k particles from the pervious set with replacement, where the probability of choosing a landmark is dependant on the particle's weight w^k.

4.4.2 SONAR measurement walkthrough

Dealing with SONAR measurements is conceptually much simpler than dealing with visual measurements. The previous walkthrough demonstrated the effectiveness of estimating the robots pitch and roll using features in the image plane. The roll and pitch are updated at a much higher frequency; so, the latest estimate can safely be used to adjust the range measurement from the SONAR sensor. The control value is calculated in the same way as it is for visual measurements.



(a) no likely match with any known landmark.

(b) minor similarity with one or more landmarks.

(c) likely match with another landmark.

Figure 4.12: **Range measurement likelihood**
(a) depicts the observation of a new landmark. (b) depicts the observation of a new landmark with similarity to other existing landmarks. (c) depicts the observation of an existing landmark. See text for details.

Measurement likelihood When determining the likelihood that a range measurement corresponds to a landmark there are three cases to consider all of which are illustrated in Figure 4.12. The first case involves a measurement that is completely irrelevant to all existing measurements, the likelihood of a match in this case is near zero and the measurement is treated as a new landmark. The second case deals with a measurement that is proximally similar to one or more landmarks while still signifying a newly observed landmark, this measurement is used to create a new landmark. The third and final case deals with a measurement that corresponds to an existing landmark and is used to update the landmark in the landmark update process. The particular likelihood value is calculated using the standard method used in the FastSLAM algorithms.

Data association Since only a single data point is received at a rate of 1Hz, there is no need to cull measurements for the sake of increasing computation efficiency in the data association step. This is because with only a single measurement the Hungarian algorithm reduces to a maximal likelihood search.

Robot pose update A newly hypothesized state is sampled using the method described in the visual measurement walkthrough of this algorithm.

Map update The map update process is the same as previous described; however, the time to live and sighting counters are irrelevant for these types of landmarks. Without these counters any landmarks generated from range measurements are never removed from the map.

Resampling The previous five paragraphs have detailed the steps taken for each of k particles within the Rao-Blackwellized particle filter while dealing with range measurements within this FastSLAM 2.0 variant. The resampling stage here follows the same process described in the algorithm walkthrough for visual measurements.

4.4.3 Compass sensor walkthrough

Although not used in the primary implementation of the algorithm, in the following chapter a compass will be integrated in the SLAM solution in order to investigate the potential benefit of such a device. For completeness, the process of integrating a tilt-compensated compass within FastSLAM 2.0 is included below.

Unlike standard measurement models, the measurement model of the compass can be factored out of the posterior distribution and used solely as a state predictor in the path posterior. The new factorization of the posterior distribution is shown in Eqn. 4.16.

$$p(\mathbf{x}_t, \mathbf{M}|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \tilde{\mathbf{Z}}_{0:t}, \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \tilde{\mathbf{Z}}_{0:t}, \mathbf{x}_0) \prod_n p(\mathbf{M}_n|\mathbf{x}_t, \mathbf{U}_{0:t}, \mathbf{Z}_{0:t}, \mathbf{x}_0) \tag{4.16}$$

In the interest of maintaining a common code base between all sensors, compass measurement were integrated into the algorithm via the control($\mathbf{u}_t$) variable's angular velocity. This requires a rederivation of both the covariance of the control error and the covariance of the odometry error. This is done using the

method previously described in Subsection 4.2.1. The new control variable is substituted in place of that used in the processing of both SONAR and camera measurements. This method also has the benefit of not resampling the robot's pose for each incoming compass measurement which is sampled at a high frequency.

4.5 Summary

This chapter detailed the theory and implementation details of the modifications to the original FastSLAM 2.0 algorithm used to perform SLAM using visual and SONAR measurements. Details are provided on the process of representing omnidirectional shorelines images, SONAR measurements, and compass measurements within the FastSLAM 2.0 formalism. For shoreline measurements, a SIFT-matching function is used to score potential matches over time, and a time to live process is used to prune potential matches that are not stable over a number of frames. The sonar measurements are corrected for vehicle pitch, roll and tilt and are integrated over the vehicle's motion. Finally, compass values, if available, can be integrated into the SLAM representation.

Chapter 5

Experimental validation

This chapter describes the procedure for deploying the robot, conducting data collection and evaluating the results of SLAM algorithm developed during this research.

5.1 Hardware deployment

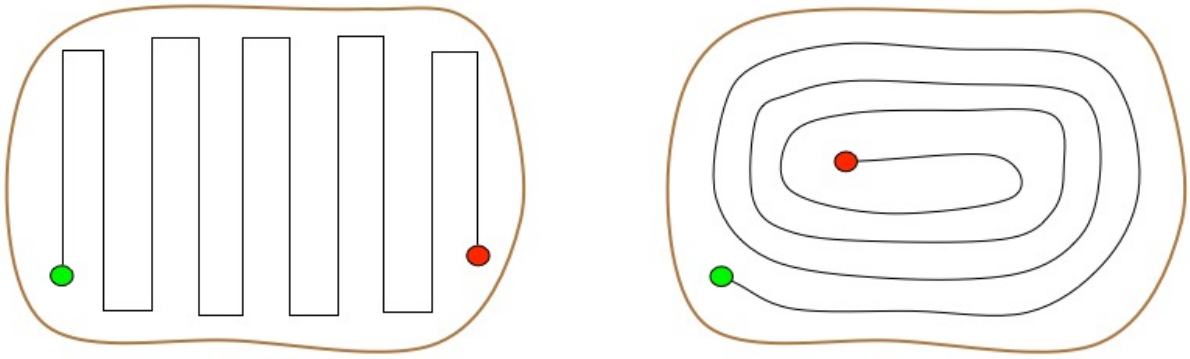
The Kingfisher M100 is a large robot that cannot be transported between bodies of water as easily as a kayak or canoe. Transportation of this aquatic vehicle is aided by a wagon with an undercarriage to store equipment and tools that are required on site. The equipment that is normally transported along with the robot includes; a mooring line to tie the robot to the dock, a video camera to record experiments from an additional advantageous viewpoint and a set of chest high waders to help retrieve the robot if it becomes stuck in mud. Once all the equipment has been transported to a location on shore where the robot can be deployed there are a number of steps that need to be followed to properly launch the robot into in its new environment. If a dock is available the robot needs to be carefully slid off the dock into water, this

Figure 5.1: **Onshore equipment**
Onshore equipment used to support the robot during field tests. Primary support equipment includes: video camera and tripod, radio basestation and landlocked GPS antenna and other miscellaneous equipment.

is easier if both left/right props are removed beforehand. After the robot is in the water both props need to be mechanically and electronically reconnected. If no dock is available then the robot must be lifted into the water sufficiently far from the shore so that the waterbed does not interfere with the rotation of the propellers. Once the robot is safely in the water it can be switched on and the ROS processes that initialize its core functionality started. The landlocked basestation governs the LAN via a wired router which uses DHCP allowing connection to the robot's local area network via a direct ethernet connection to the basestation. Any laptop connected to the basestation can remotely initiate the ROS nodes or other processes that are needed to control the robot using a standard joystick.

5.2 Exploration strategy

SLAM implementations are often volatile with respect to the path taken to explore the environment. Without an effective exploration strategy many SLAM algorithms would fail to collect the information necessary to properly map the surrounding environment. There exists a wide variety of exploration algorithms within the literature, some of which are reviewed in [44]. The development of new algorithms for the exploration



(a) The Boustrophedon search path within a simple environment.

(b) A spiral search path within a simple environment.

Figure 5.2: **Exploration strategy**
Two common exploration and sensor coverage paths that can be used to collect data within a simple environment. Both paths are effective in simple environments and exhibit different properties when extrapolated to more complex environments.

of aquatic surface environments is beyond the scope of this research. The exploration strategy employed by this SLAM implementation has be predetermined using a simple heuristic that can be described anecdotally. Travelling parallel to the boundary of the environment where there exists a large number of identifiable visual landmarks is an effective way to map the boundary of the environment. This is an effective way of reducing the uncertainty associated with visual landmarks that can be repeatably observed. After creating a reliable map of the boundary the rest of the environment is explored using an inward spiral (e.g. Fig. 5.2). The spiral exploration path slowly increases the distance away from the mapped boundary while mapping the depth of the lake bed. Actualization of this exploration process for three test environments is shown in Fig. 5.3.

5.3 Test environments

A number of data sets were collected to evaluate the SLAM algorithm presented here. Datasets were collected on Stong Pond (York University), on Okanagan Lake (Knox Mountain Park), near Kelona, British Columbia, and on Ramsey Lake (Lake Laurentian Conservation Area), near Sudbury Ontario (see Fig. 5.3).

(a) Stong Pond test run

(b) Okanagan Lake test run

(c) Ramsey Lake test run

Figure 5.3: **Test environments**
Ground truth trajectory of the robot measured using GPS overlaid on top of a map of the environment. The green dot shows the initial state of the robot, whereas the red dot shows the final state of robot (Background images appear courtesy of Google Earth).

These environments were not completely explored due to known limitations of the robot's battery capacity, radio signal reception and concerns of the safety for the robot's components. These safety concerns include damage to the propeller blades and loss of the robot. Propeller damage was minimized by keeping the robot a safe distance from the shore to prevent contact with debris lying on the lake bed. The robot's available battery power limits the distance the robot can travel away from the shore and the launch site before a return journey is necessary to prevent the robot becoming stranded. The reliability of the communication between the land operator and the robot limits also limits the exploration range of the robot.

5.3.1 Stong Pond

Stong pond is a small body of water located on the York campus in Toronto (Fig. 5.3a). Stong pond is the primary testing site for all aquatic surface vehicles from our robotics lab due to its proximity and floating dock that aids in shore based deployment allowing the robot to be slid/placed in the water at a safe depth and distance from shore.

5.3.2 Okanagan Lake

Okanagan Lake is a large lake located to the west of Kelowna, British Columbia and was the primary site for a multi terrain robot field trail in June of 2015 (Fig. 5.3b). Deployment of the robot on lake Okanagan from the available boat launch requires a submersible trailer that can used to partially submerge the vehicle and float it out to a safe distance from shore.

5.3.3 Ramsey Lake

Ramsey Lake is a large lake situated near the downtown core of Sudbury, Ontario and served as one of the sites for a multi terrain robot field trail in June of 2016 (Fig. 5.3c). The robot was deployed off a small rocky outcropping piece by piece and assembled in the shallow water.

5.4 SLAM Representations

The maps and trajectories obtained with SLAM algorithms are probabilistic. As a consequence the resulting maps shown in Figs. 5.5, 5.6, 5.7 are not visually intuitive, this is because the maps are not strictly geometric representations of the surrounding environment but a set of individual visual landmarks. Maps that were

Valid	Visual (Camera)	Audio (SONAR)	Magnetic (Compass)
✓	✓	✓	✓
✓	✓	✓	x
✓	✓	x	✓
✓	✓	x	x
x	x	✓	✓
x	x	✓	x
x	x	x	✓

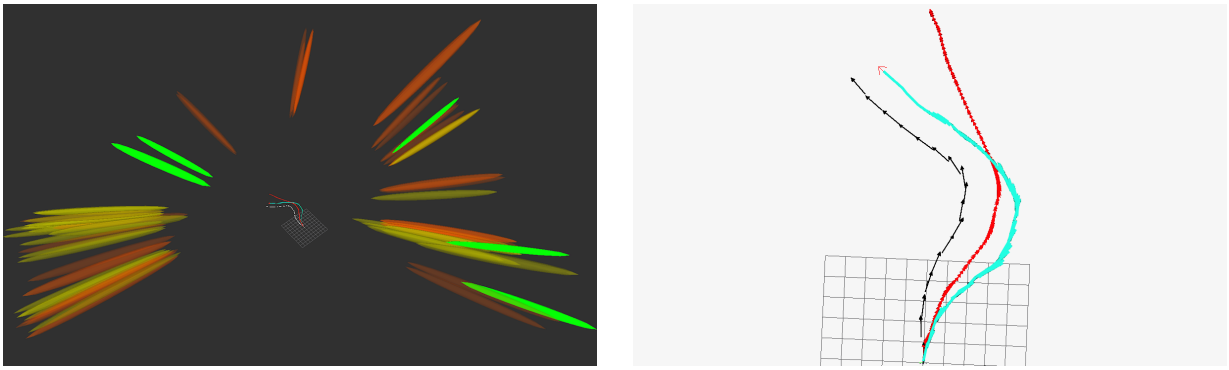
Table 5.1: **SLAM sensor combinations**

The different combinations of sensors that are used to perform SLAM in this work. Note that combinations in which visual data is unavailable do not result in a valid combination of sensors.

generated with the help of a SONAR sensor have an additional set of geometric depth features of the lakebed. All landmarks are depicted using an ellipse around the landmark's mean representing the 99% confidence region of its covariance matrix. Visual landmarks are coloured on a spectrum from red-yellow-green, to represent the number of times the landmark has been sighted. Bright green landmarks are permanent fixtures of the map, landmarks of other colours still play a minor role in localization but may be removed from the map at a later time.

5.5 Experimental validation

Experiments are grouped around answering two basic questions about the approach developed in this work (i) Particle-based SLAM algorithms are highly dependent on the number of particles used to represent the underlying distribution. How many particles are appropriate to represent this distribution? (ii) How effective are the visual, sonar and compass sensors in solving SLAM for an ASV? In particular, what can be accomplished without a compass, or to put it another way, how useful is a compass? Each of these questions are addressed in the following sections.



(a) Map sample (b) Path sample

Figure 5.4: **Stong Pond SLAM - visual only (10 particles)**
Shows a partial run of pyFSLAM with 10 particles using only measurements from the visual sensor. (a) Shows the predicted motion in teal, ground truth in white and the robot odometry in red along with the map. Map landmarks are depicted using a 99% error ellipse determined from their associated covariance matrices. Landmarks are coloured on a spectrum from red-green based on the number of times they have been sighted. All depth landmarks are added as permanent fixtures of the map and appear as if they are green paths on the map. (b) Shows the recovered motion paths; ground truth odometry calculated using onboard GPS and Compass is shown in black, the SLAM estimate of the robot's path is shown in teal and the estimated odometry calculated using the odometry model alone is shown in red, which is the same across all sensor configurations.

5.5.1 How many particles are enough?

Empirically, 2D ground-contact SLAM is normally solvable using a reasonably small number of particles, upwards of ten (see [18][38][19]). The SLAM problem here is somewhat more complex given the lack of a simple robust plant model for the vehicle. So a key question in actually running PyFSLAM is how many particles are appropriate to represent the distribution, and does this number change when different combinations of sensors are used?

Figs. 5.5a, 5.5b show the performance of PyFSLAM with vision-only sensing for a 10 particle representation of the PDF incomparison to the data obtained from the raw odometry and GPS/Compass state estimates. As shown by Figs. 5.5a, 5.5b 10 particles in PyFSLAM are insufficient to properly represent the map-trajectory PDF sufficiently well to solve the problem. After a number of experiments it became clear

that at least 40 particles were required to properly represent the PDF.

5.5.2 How important are individual sensors and their combinations?

Theoretical results [14] have demonstrated that a compass is an extremely powerful device in terms of solving the SLAM problem in a theoretical sense. But how important is a compass to being able to solve the SLAM problem practically? In these experiments we compare PyFSLAM performance on vision alone versus vision+compass, and vision+SONAR versus vision+SONAR+compass in order to address this question. Experiments were run on three different lake surfaces, with the four sensor combinations shown in Table 5.1 Figs. 5.5-5.7 show the experimental results and Table. 5.2 summarize the findings. In all cases 40 particles were used to approximate the map-trajectory probability distribution function. Some general observations can be made from the data.

First, the plant model on its own is insufficient to solve the localization problem. The plant error is large. This is not a particularly surprising result. If the plant was sufficiently accurate then there would be no need for SLAM. The poor performance of the plant model on its own is perhaps best highlighted by performance on Lake Okanagan and Lake Ramsey, where the final RMS deviation of the robot's odometry is 293m. The experimental run in Lake Ramsey was in a particularly unsheltered region of the lake and was subject to strong wind and wave action.

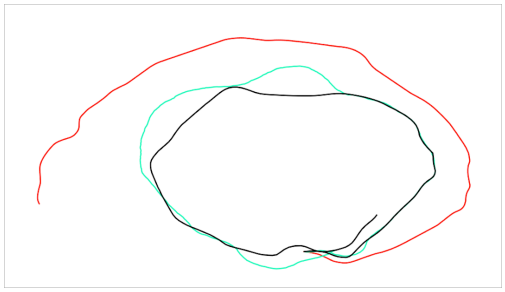
Second, more sensors are generally more helpful, but this is not always the case. Again not a particularly surprising result. Performance with vision and sonar is better than vision alone in two out of three of the environments. Performance with vision and a compass was always better than performance with vision alone, and performance with vision, sonar and a compass was aways better than performance with one or two sensors. The degree to which additional sensors improves localization depends on the environment observed by the additional sensor. Again, this is not that surprising. If one were operating the robot in a pool with a flat bottom, SONAR depth would provide no additional information. This situation also presents itself

when the depth below the robot is less than the minimum sensing depth of the sensor, as was the case in part of the Ramsey lake experiment. Although Ramsey Lake is quite deep (greater than 20m), portions of the robot's trajectory took it out over relatively shallow portions of the environment. The SONAR sensor does not return an invalid value for depth shallower than its minimum operational depth, but rather returns a minimum value. This issue is considered later in this thesis. That being said, it is rare to find a lake bed with such little variability over its entire area, and SONAR can be expected to do two things (i) SONAR data can be expected to provide data to help disambiguate one location from another if the depth of the bottom is in disagreement. (ii) SONAR data helps to prevent underestimation of the robot's motion. To see this, observe that under the stationary world assumption, if the depth to the sea bed is changing then the robot must have moved. This is a critical piece of information in the application for ASV's as the plant model is particularly poor.

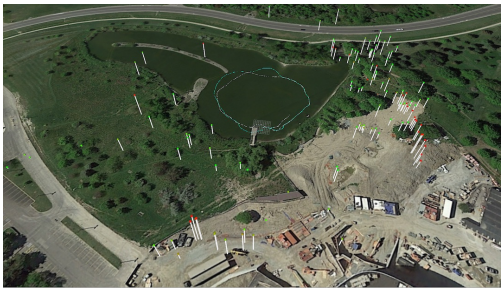
Third, a compass is a powerful thing. In the experiments performed in this work, the addition of a compass always improves the accuracy of the SLAM path relative to performance without one. For a number of the environments considered in this work the addition of compass data was particularly significant (e.g., in the Stong Pond experiments). For other cases the relative improvement associated with a compass is not as significant. For example, in the experiments on lake Ramsey the impact of compass data on localization is quite small relative to performance with the vision sensor alone. This is likely because the visual-only SLAM approach performed well in this instance on its own, rather than a lack of useful information from the compass.

5.5.3 SLAM paths and maps

This section shows the estimated paths and maps generated from different configurations and parameterizations of the PyFSLAM algorithm. The main difference between these configuration are the combination of sensors used to collect data. Obviously there are many combinations of the three sensors that cannot

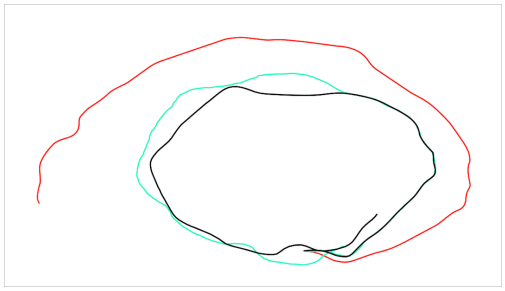


(a) Path sample

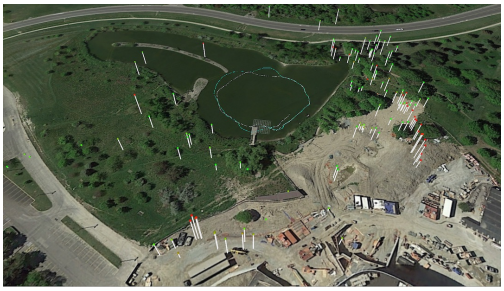


(b) Map overlay

Vision Only



(c) Path sample



(d) Map overlay

Vision and Depth

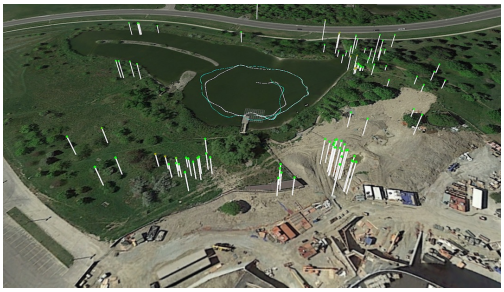
Figure 5.5: **Stong Pond SLAM (40 particles)**
This figure is continued on the next page.

produce a viable map. Combinations that might produce a valid map are shown in Table 5.1. The combinations marked as invalid can be excluded from our tests because they do not carry sufficient information to determine the position and orientation of the robot through repeated measurements. Visual measurements are needed as an aspect in all mapping attempts because of the plethora of potential landmarks and their relative accuracy.

5.5.4 When SLAM fails

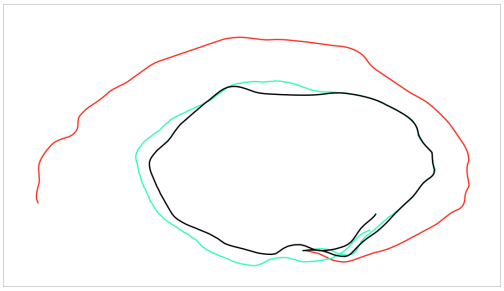
Although the results given in Table 5.2 suggests that the SLAM algorithms were more or less successful under all conditions, there are clear deviations in the recovered robot paths, especially when only two sensors are

(e) Path sample

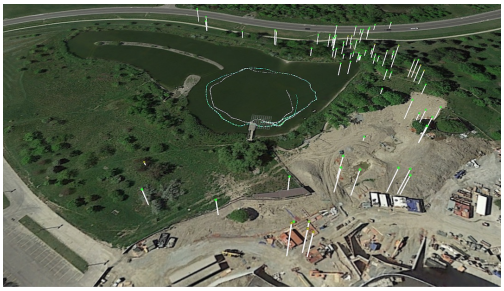


(f) Map overlay

Vision and Compass



(g) Path sample



(h) Map overlay

Vision, Compass and Depth

Figure 5.5: **Stong Pond SLAM (40 particles) (cont.)**
The final stage of the SLAM run for all sensor combinations shown in Table I. (a,c,e,g) Shows the recovered motion paths; ground truth odometry calculated using onboard GPS and Compass is shown in black, the SLAM estimate of the robots path is shown in teal and the estimated odometry calculated using the odometry model is shown in red, which is the same across all sensor configurations. (b,d,f,h) Shows the generated map (landmark means+height) and ground truth and SLAM estimated paths overlaid on a satellite view of the environment (image courtesy of Google Earth). For the map overlay display ground truth is shown in white and recovered trajectory is in teal. Landmarks are displayed using a coloured sphere around their estimated mean on top a white line depicting their height above the robot's ground plane. Permanent landmarks are coloured green and temporary landmarks, red. The perspective view of all map images are slightly different to better display all landmarks. Note: Terrain images shown for each map overlay are satellite images courtesy of Google Earth and do not necessarily represent the terrain at the time of data collection.

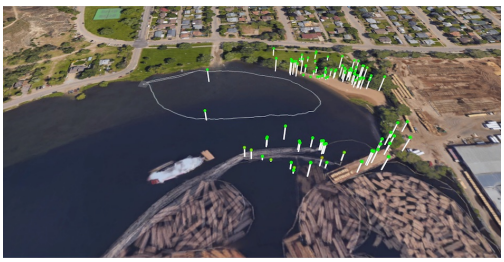
used. Exploring what effects lead to these deviations from ground truth provides insights into the weaknesses of various sensors and how one might go about adapting to them.

(a) Path sample

(b) Map overlay

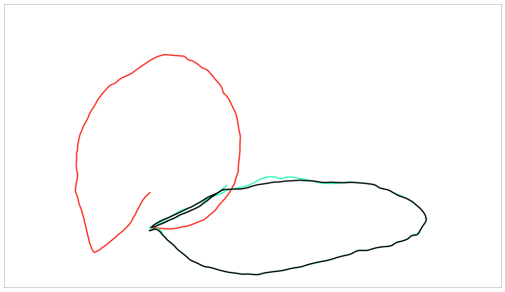
Vision Only

(c) Path sample



(d) Map overlay

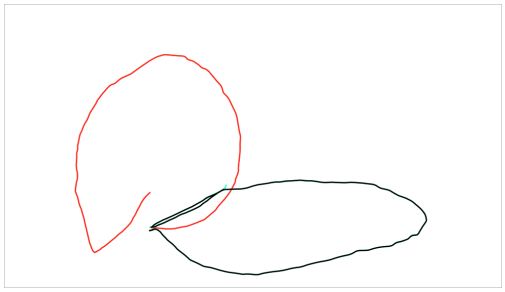
Vision and Depth



(e) Path sample

(f) Map overlay

Vision and Compass

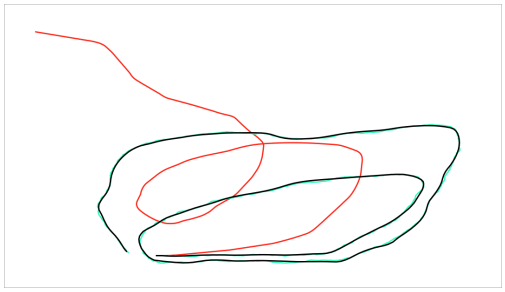


(g) Path sample

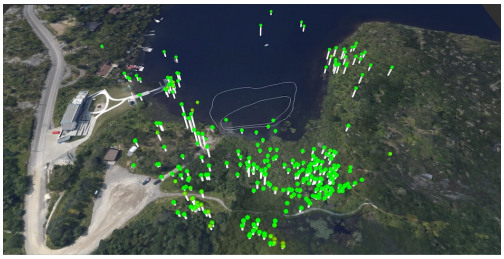
(h) Map overlay

Vision, Compass and Depth

Figure 5.6: **Okanagan Lake SLAM (40 particles)**
Shows the final stage of the pyFSLAM for all sensor combinations. See Figure 5.5 for a description of the elements within each image in this figure.



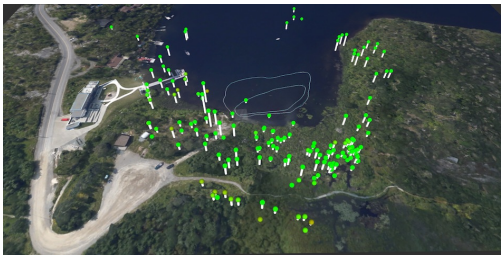
(a) Path sample



(b) Map overlay

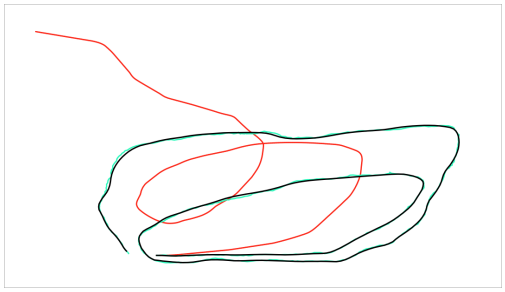
Vision Only

(c) Path sample

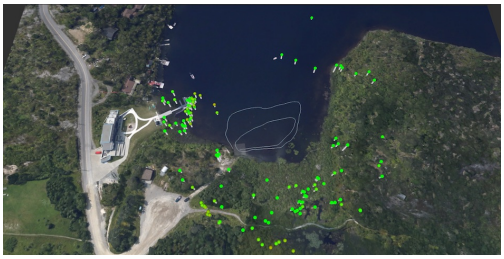


(d) Map overlay

Vision and Depth

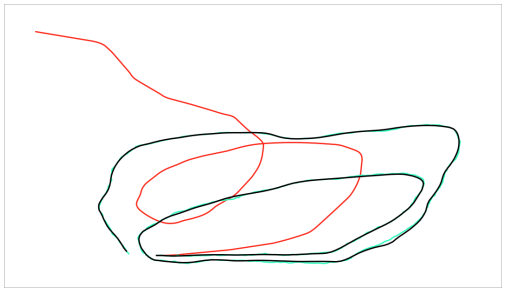


(e) Path sample

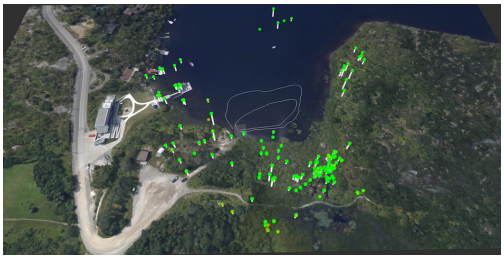


(f) Map overlay

Vision and Compass



(g) Path sample



(h) Map overlay

Vision, Compass and Depth

Figure 5.7: **Ramsey Lake SLAM (40 particles)**
Shows the final stage of the pyFSLAM for all sensor combinations. See Figure 5.5 for a description of the elements within each image in this figure.

	Stong Pond			Okanagan Lake			Ramsey Lake		
	RMS	**MAX**	**RMS%**	**RMS**	**MAX**	**RMS%**	**RMS**	**MAX**	**RMS%**
Odometry	35.408	68.778		139.696	239.040		60.518	148.093	
Visual	9.484	15.005	26.7%	10.144	26.700	7.2%	1.132	3.522	1.8%
Visual + SONAR	6.648	10.707	18.7%	7.804	21.293	5.5%	1.618	4.056	2.6%
Visual + Compass	4.984	8.133	14.0%	2.718	9.379	1.94%	0.835	2.085	1.3%
Visual + SONAR + Compass	3.213	5.979	9.0%	0.649	3.307	0.4%	0.724	1.868	1.1%

Table 5.2: **SLAM error over the entire path.**
Shows both the root mean squared error (RMS) and maximum deviation from ground truth for each SLAM configuration's (including Odometry) of the robot's estimated path for each recovered data set. Data gathered from the onboard DGPS sensor was taken as ground truth. RMS percentage(RMS%) is reported as the percentage of RMS error relative to odometry.

(a) path (b) map (c) image

Figure 5.8: **SLAM visual failure**
Series of image outlining the context behind a major failure in the visual aspects of the SLAM algorithm. (a) SLAM path with section of deviation highlighted. (b) Google Map with area of deviation highlighted. (c) Image with SIFT features sample from data during path deviation, areas of the closest shore perpendicular and off-perpendicular to motion are highlighted.

Vision Sensor

Small deviations of the SLAM path from ground truth can be easily explained as a consequence of issues related to the sensor (e..g, sensor noise) or issues related to failures in the particle-based representation of the underlying PDF's (e.g., proposal sampling error), or failures in the ground truth estimation process (e.g.,

(a) path

(b) map

Figure 5.9: **SLAM depth failure**
Images outlining the context behind a major failure in the SONAR(depth) aspects of the SLAM algorithm. (a) SLAM path with area of depth deviation highlighted in yellow and path deviation highlighted in red. (b) Google Map with depth deviation highlighted in yellow and path deviation highlighted in red.

GPS failure). Larger deviations from ground truth represent failures in the SLAM algorithm that can be caused by assumptions made about the environment that are violated, at least for short periods of time. The largest deviation encountered between GPS-based ground truth and recovered SLAM trajectory was seen during the Stong pond experiment (shown in Fig. 5.8). The section of the path associated with the SLAM error (shown in Fig. 5.8a) occurs in a portion of the map (shown in Fig. 5.8b) where there are very few strong SIFT features on the closest shore perpendicular to the robot's motion (highlighted orange in Fig. 5.8c). The off-perpendicular sections (highlighted yellow in Fig. 5.8c) would be the "next best" places for strong SIFT features to be detected; however, these sections are also devoid of features. The algorithm relies on SIFT features to produce measurable changes in bearing when moving perpendicular to them. In practice this means that shore regions with few features are problematic.

SONAR(depth) sensor

This algorithm can also fail under circumstances involving the use of the depth sensor. In the case of the Ramsey lake data set (see Fig. 5.7) the inclusion of the depth sensor without a compass actually degraded

the accuracy of the recovered motion. The majority of the readings from the depth sensor along the path sections highlighted yellow in Fig. 5.9 were shallower than the sensor's minimum operating depth. The readings defaulted to the sensor's minimum value. Without useful depth information the possibility of underestimating the robot's motion increases. Failure can be seen within the path sections highlighted red in Fig. 5.9. The recovered motion (teal) is misidentified as intersecting the first loop within the environment. This particular failure can be explained by three contributing factors; the motion underestimation in the previous path segment, the relative proximity between the highlighted path segment, and the inability to distinguish between similar depth readings. This problem could have be avoided by carefully planning a path that did not traverse so closely to itself.

Compass

Compass failures were not encountered within these experiments. A high precision, tilt-compensated compass proved to be an exceptionally reliable sensor in the experiments reported here.

5.6 Summary

The SLAM algorithm developed in the research is able to use visual measurements to drastically improve location estimates despite an inaccurate motion model and a volatile environment whilst using a simplified (2 dimensions) state space. Estimates of the robot's location begins to fail in segments where there are no sensed landmarks that are positioned approximately perpendicular to the robots trajectory. The addition of depth measurements results in an improved path estimate relative to the GPS estimate (RMS 9.484 vs RMS 6.648) with maximum deviation from the GPS path of 15.005 m for vision only and 10.707 m for vision+SONAR sensors. Depth measurements help to prevent underestimation of the robot's motion as long as there is sufficient variability in the topology of the lakebed.

Chapter 6

Summary and future work

6.1 Summary

Simultaneous localization and mapping is a fundamental problem in autonomous robotics. Although considered solved for sufficiently well behaved versions of the problem, complex outdoor environments still present challenges for SLAM algorithms. This work considered SLAM on the surface of lakes in GPS and compass denied environments. In particular it investigated how visual omnidirectional bearing data could be combined with local lake depth information in order to solve SLAM and the benefits of a compass under such conditions.

This research has shown how the simultaneous measurement extension of the original FastSLAM 2.0 algorithm[38] described in [59] can be further extended to incorporate measurements from both a SONAR sensor and omnidirectional camera. This extension includes a number of features that allow the algorithm to be tuned in a variety of ways. The most notable difference between this implementation and its predecessor

is the inclusion of a time to live counter associated with each landmark that is used as the primary method for removing inconsequential landmarks from the map. This accounts for the assumption adopted in the previous implementation that landmarks are visible throughout the environment. In this research landmarks are considered weak until they have been observed at at least $\beta_{\mathbf{z}_t,\psi_j}$ times and have increased in certainty by at least 50%. This puts less pressure on a landmark's continual observance and instead puts more weight on the amount of certainty associated with the landmark. This method helps to increase the number of weak landmarks that are removed from the map over time. Another major change introduces a weighting function augmenting the proximal likelihood of an association between a landmark and measurement. This weighting function helps to lend more influence to the visual likelihood of a measurement association when a landmark has a high uncertainty. This improves the effectiveness of the data association process. This weighting function is also used to prioritize landmarks in the particle weighting process based on their certainty.

The SLAM approach presented in this work has shown its merits in mapping static marine environments, especially when all presented sensors are used concurrently. Unfortunately the relative instability of SIFT features under changes in environmental lighting conditions may preclude generated maps from being used for navigation at drastically different times of the day. Low lighting conditions are likely to prevent this approach from being used as an effective mapping tool altogether. However, the pixpro[51] omnidirectional camera used in this research is a phenomenal sensor that provides panoramic visual information and a wide FOV. A single visual sensor that can provide panoramic images is much more desirable than a set of cameras that must be meticulously calibrated and stitched together to provide a similar panorama at a higher resolution.

When the compass on board the Kingfisher is integrated within the SLAM process it always resulted in an improved estimate of the robot's motion within the environment. The navigational information provided by this sensor is extremely valuable as long as electro-magnetic interference does not disrupt the output from the compass. A compass is also a valuable addition to the repertoire of any robot due to its ability to estimate the orientation of the robot on a global scale.

The depth sensor installed on the Kingfisher and utilized by this research was instrumental in properly determining the robot's linear motion in data sets presented in this thesis. Given the relatively low cost of the sensor, the expense can absolutely be justified given its benefits. This given the shortcomings of the sensor are properly taken in account. The primary shortcomings of the sensor are its inability to produce raw data and wide beam angle. In addition to the sensor's benefit to SLAM it can also provide information useful for a variety of other applications such as navigating away from the lakebed features that may pose a risk to the robot.

6.2 Future work

Future work will seek to further quantify the relative significance of added sensors in improving SLAM results. Critical here is repeated experiments in the same environment using different exploration strategies, under differing weather/seasonal conditions to account for due lighting and other seasonal changes. To determine weather the inclusion of a sensor into the SLAM process provides an improvement that is statistically significant, requires large sampling of RMS errors for each combination of sensors under varying conditions. As a result of the stochastic nature of FastSLAM2.0 multiple runs of the algorithm using the same data set can increase the sample size without the need for excessive amounts of data collection.

6.2.1 Applications

The ability to map aquatic environments without the express need for a GPS sensor or compass opens up a wide variety of applications. Three applications of particular interest are (i) mapping the volume of tailing ponds where materials may interfere with the accuracy of an onboard compass (ii) underground or ground water caverns where GPS sensors are rendered inert (iii) non-terrestrial environments that lack a stable magnetic field or orbital position satellites. These are described in detail below.

(a) Aerial view of a tailing pond in Elko, Nevada.

(b) Terrestrial view of s tailing pond in Elko, Nevada.

Figure 6.1: **Tailing ponds**
Tailing ponds contain materials leftover after the mining process. A portion of these material can adversely effect the performance of a compass.

Tailing ponds

Tailing ponds are remnants of the process of mining. In many countries tailings ponds and their upkeep are subject to strict environmental regulations. Upkeep of these tailing ponds requires accurately maintaining the ratio of tailings to water. Tailings by nature are more dense than water and therefor settle at the bottom of the pond. Since tailing ponds are manually constructed their dimensions are known although shifts in the ground and the pressure caused by the material stored in the pond can lead to substantive changes in pond geometry. A SONAR sensor that can measure the depth of the water column below the sensor can used to measure the volume water in the pond. The total volume of materials in the pond minus the volume of water can be used to calculate the volume of tailings in the ponds. The current method to perform this calculation requires commercial divers to manually measure the depth of the water in a number of positions within the pond. The current process is inexact, expensive and dangerous as it subjects divers to chemical runoff from the mining process. Automated solutions are clearly desired. In such a situations it likely is possible to place beacons on the shore to aide in navigation at the expense of numerous beacons for each site and cost attributed to their maintenance.

Aquatic terrestrial caverns

Terrestrial ground water and underground aquatic caverns are two environments where GPS sensors are unable to provide global positioning data. Some of these environments may also interfere with the accuracy of compass data due to deposits of ferrous metals. As long as there is enough ambient lighting to detect visual features in the environment and the lakebed environment has a enough variability the SLAM algorithm developed here can be of use to mapping these environments. In the event where there is not enough ambient light to properly detect visual features a high powered diffuse light source may be useful to detect visual features.

Non-terrestrial aquatic environments

Non-terrestrial aquatic environments fit a unique category where there is no stable magnetic field to determine absolute orientation and the infrastructure for a global positioning system does not exsist. Titan[11] is the only known celestial object with bodies of surface liquid[7] other than Earth in our own solar system. If the ethane and methane lakes of Titan[10] are not frozen below a certain depth and the lakebed topology can be measured using a SONAR sensors this algorithm can be instrumental in mapping these types of environments.

This research has addressed how the inclusion of the depth the water column can be used to improve localization and mapping using an aquatic surface vehicle. The most obvious extension to this research is to upgrade the single-beam SONAR sensor in favour of either a multi-beam or side-scanning SONAR sensor. These sensors provide measurements of the lakebed at much higher resolution and wider field of view. Integrating these sensors into a appropriate SLAM algorithm would allow for higher fidelity mapping of the lakebed and the ability perform scan matching between scans and/or the environment. There are additional sensors that can be added to the robot to provide increased situational awareness. LiDAR sensors

can be effective for detecting environmental landmarks at moderate distances. RADAR sensors are effective for moderate to long range detection of other vehicles, terrain and even weather patterns. The addition of these sensor types would help offset mapping performance under adverse conditions such as poor lighting conditions and fog, both of which prevent visual sensors obtaining accurate measurements. LiDAR can still operate effectively without ambient light and RADAR remains unaffected by all but the most severe weather.

There are also many other avenues that can be explored from an algorithmic standpoint. One such adjustment that could be made changes the assumption made about the sparsity of depth measurements. In the current formulation the environment is assumed to be sparsely populated by landmarks above and below the surface of the water. The visual landmarks within the scene can be modelled as sparse so long as they are differentiable by a means other than their location. However SONAR measurements can be obtained from every location within the environment and may be more effective modelled as densely packed landmarks. This would require a reformulation of the SLAM algorithm introducing increased complexity with the benefit a better model of the lakebed.

The un-delayed initialization of bearing-only landmarks adopted in this implementation is sufficient given the employed exploration strategy. However, delayed initialization approaches offer a much more robust way of calculating the position of new landmarks at the expense of increased algorithmic complexity. The delayed initialization approach used in [59] requires that landmarks can be differentiated solely via some sort of descriptor which can introduce errors from visually similar landmarks from different locations within the environment. The most promising delayed initialization approach[63] employs a Gaussian sum filter to represent the initial uncertainty of a new measurement using multiple hypotheses, after numerous measurements one estimate within the filter eventually wins out over the others.

Modern SLAM algorithms such as PTAM[23] and DTAM[35] are very effective at mapping visual environments while estimating the 6DOF pose of the mapping agent. These algorithms would be very effective mapping the shore but would become less effective further from shore. The caveat with these algorithms

is they can only process images, investigating whether high resolution SONAR scans can be interpreted as images to conform to requirements of either PTAM or DTAM is of particular interest. If a SONAR scan can be interpreted as an image and integrated into either algorithm this would provide valuable information at locations in the environment where visual features are sparse.

This research can be used in a variety of applications some of which have been previously discussed. Terrestrial applications of this work include the exploration of aqueous caverns of water and mapping tailing pond with deposits of ferrous material. Non-terrestrial applications include the exploration of lakes on different celestial objects, currently the only known celestial body with lakes on its surface is Saturn's moon Titan. Each of these applications include limiting environmental features that limit the reliability of either a compass or GPS sensor. Underground or surface accessible caverns of water restrict communication with overhead satellites rending GPS sensors useless and the possibility of ferrous material in the surrounding environment limits the accuracy of a compass. Tailing ponds are placed in ideal locations for acquiring accurate GPS data; however, the slurry of materials settling at the basin might disrupt compass readings. Celestial bodies with surface accessible lakes are devoid of the infrastructure required global positioning and a strong and stable magnetic field is not always present. Using the method presented here can be used to map such environments despite such sensor limitations.

Bibliography

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping (SLAM): Part I," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 2, pp. 99–110, 2006.

[2] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, no. 3, pp. 195–207, 1998.

[3] U. Government. (2015) Official u.s. government information about the global positioning system (gps) and related topics. Accessed 25-September-2015. [Online]. Available: http://www.gps.gov

[4] Google. (2015) About - google maps. Accessed 25-September-2015. [Online]. Available: https://www.google.ca/maps/about/

[5] O. Foundation. (2015) Openstreetmap. Accessed 25-September-2015. [Online]. Available: https://www.openstreetmap.org

[6] OpenSeaMap. (2015) Openseamap: Mainpage. Accessed 25-September-2015. [Online]. Available: http://www.openseamap.org

[7] A. Hayes, R. Michaelides, E. Turtle, J. Barnes, J. Soderblom, M. Masrtogiuseppe, R. Lorenz, R. Kirk, and J. Lunine, "The distribution and volume of titan's hydrocarbon lakes and seas," in *Lunar and Planetary Science Conference*, vol. 45, 2014, p. 2341.

[8] NASA. (2013) Hubble sees evidence of water vapor at jupiter moon. Accessed 25-September-2015. [Online]. Available: http://www.jpl.nasa.gov/news/news.php?release=2013-363

[9] ——. (2014) Nasa space assets detect ocean inside saturn moon. Accessed 25-September-2015. [Online]. Available: http://www.jpl.nasa.gov/news/news.php?release=2013-363

[10] C. Neish, R. Lorenz, D. O Brien, C. R. Team *et al.*, "The potential for prebiotic chemistry in the possible cryovolcanic dome Ganesa Macula on Titan," *International Journal of Astrobiology*, vol. 5, no. 1, p. 57, 2006.

[11] S. Ledvina, J. Luhmann, S. Brecht, and T. Cravens, "Titan's induced magnetosphere," *Advances in Space Research*, vol. 33, no. 11, pp. 2092–2102, 2004.

[12] G. Davis, *Theseus and the Minotaur*. Osprey Publishing, 2014, vol. 12.

[13] J. G. Sindy McKay, Wilhelm Grimm, *Hansel and Gretel*. Treasure Bay, Inc., 1999.

[14] H. Wang, M. Jenkin, and P. Dymond, "Enhancing exploration in topological worlds with a directional immovable marker," in *Proc. 2013 International Conference on Computer and Robot Vision (CRV)*. IEEE, 2013, pp. 348–355.

[15] O. Aharonson, "Time: Titan mare explorer," 2010, [Online; accessed 20-September-2015]. [Online]. Available: http://www.kiss.caltech.edu/workshops/titan2010/presentations/aharonson.pdf

[16] H. Durrant-Whyte, D. Rye, and E. Nebot, *Robotics Research: The Seventh International Symposium*. London: Springer London, 1996, ch. Localization of Autonomous Guided Vehicles, pp. 613–625. [Online]. Available: http://dx.doi.org/10.1007/978-1-4471-1021-7_69

[17] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proc. 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2432–2437.

[19] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Oct 2003, pp. 206–211 vol.1.

[20] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proc. 2003 ACM International Joint Conference on Artificial Intelligence (IJCAI)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 1135–1142. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630822

[21] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *Proc. 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1691–1696.

[22] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM - 3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[23] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. 2009 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Oct 2009, pp. 83–86.

[24] A. Harmat, I. Sharf, and M. Trentini, "Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle," in *Intelligent Robotics and Applications*, ser. Lecture Notes in Computer Science, C.-Y. Su, S. Rakheja, and H. Liu, Eds. Springer Berlin Heidelberg, 2012, vol. 7506, pp. 421–432.

[25] A. Harmat, M. Trentini, and I. Sharf, "Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 291–317, 2015.

[26] J. M. Sáez, A. Hogue, F. Escolano, and M. Jenkin, "Underwater 3d slam through entropy minimization," in *Proc. 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3562–3567.

[27] A. Chambers, S. Achar, S. Nuske, J. Rehder, B. Kitt, L. Chamberlain, J. Haines, S. Scherer, and S. Singh, "Perception for a river mapping robot," in *Proc. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2011, pp. 227–234.

[28] C. Bibby and I. Reid, "A hybrid SLAM representation for dynamic marine environments," in *Proc. 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 257–264.

[29] W. Fink, M. A. Tarbell, R. Furfaro, L. Powers, J. S. Kargel, V. R. Baker, and J. Lunine, "Robotic test bed for autonomous surface exploration of titan, mars, and other planetary bodies," in *Proc. 2011 IEEE International Aerospace Conference*. IEEE Computer Society, pp. 1–11.

[30] W. Fink, M. Tuller, A. Jacobs, R. Kulkarni, M. A. Tarbell, R. Furfaro, and V. R. Baker, "Robotic lake lander test bed for autonomous surface and subsurface exploration of titan lakes," in *Proc. 2012 IEEE International Aerospace Conference*, March 2012, pp. 1–12.

[31] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.

[32] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[33] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proc. 2002 National Conference on Artificial*

Intelligence (AAAI), ser. AAAI'02. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 593–598. [Online]. Available: http://dl.acm.org/citation.cfm?id=777092.777184

[34] D. Blackwell, "Conditional expectation and unbiased sequential estimation," *The Annals of Mathematical Statistics*, pp. 105–110, 1947.

[35] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proc. 2011 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2320–2327.

[36] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proc. 2011 IEEE International Symposium on Mixed and Augmented reality (ISMAR)*. IEEE, 2011, pp. 127–136.

[37] B. Steux and O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines of c-language program," in *Proc. 2010 International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE, 2010, pp. 1975–1979.

[38] 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 *Proc. 2003 International Joint Conference on Artificial Intelligence (IJCAI)*, ser. IJCAI'03. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1151–1156. [Online]. Available: http://dl.acm.org/citation.cfm?id=1630659.1630824

[39] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[40] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. New York, NY, USA: Cambridge University Press, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=331954

[41] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, and G. A. Demetriou, "Control architectures for autonomous underwater vehicles," *Control Systems, IEEE*, vol. 17, no. 6, pp. 48–64, 1997.

[42] G. K. Batchelor, *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000, Cambridge Books Online. [Online]. Available: http://dx.doi.org/10.1017/CBO9780511800955

[43] C. Robotics. (2011) Kingfisher M100 unmanned surface vehicle. Accessed 12-August-2016. [Online]. Available: https://oceanai.mit.edu/svn/moos-ivp-kfish/trunk/docs/kfish-m100-manual.pdf

[44] P. M. Forooshani, "Coordinated sensor-based area coverage and cooperative localization of a heterogeneous fleet of autonomous surface vessels (ASVs)," 2015.

[45] R. Codd-Downey, M. Jenkin, and A. Speers, "Building a ROS node for a NMEA depth and temperature sensor," in *Proc. 2014 International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 02, Sept 2014, pp. 506–512.

[46] CruzPro. (1997) CruzPro ATU120AT. Accessed 25-September-2015. [Online]. Available: http://www.cruzpro.co.nz/active.html

[47] E. Upton. (2012) Model b schematics. Accessed 25-September-2015. [Online]. Available: https://www.raspberrypi.org/blog/model-b-schematics/

[48] A. Devices. (2015) ADXL345 3-axis, 2g/4g/8g/16g digital accelerometer. Accessed 25-September-2015. [Online]. Available: http://www.analog.com/en/products/mems/mems-accelerometers/adxl345.html\#product-overview

[49] Wikipedia. (2015) Sonar — wikipedia, the free encyclopedia. Accessed 25-September-2015. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sonar\&oldid=681938766

[50] J. I. Ltd, "Kodak PIXPRO digital cameras: SP360 action camera."

[51] Kodak. (2015) Kodak pixpro digital cameras SP360 action camera. Accessed 25-September-2015. [Online]. Available: http://kodakpixpro.com/docs/specsheets/actioncamera/sp360/sp360-specsheet-en.pdf

[52] D. C. Brown, "Decentering distortion of lenses," *Photometric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[53] G. Bradski *et al.*, "The OpenCV library," *Doctor Dobbs Journal*, vol. 25, no. 11, pp. 120–126, 2000.

[54] M. Rufli, D. Scaramuzza, and R. Siegwart, "Automatic detection of checkerboards on blurred and distorted images," in *Proc. 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008, pp. 3121–3126.

[55] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Proc. 2006 IEEE International Conference on Computer Vision Systems (ICVS)*. IEEE, 2006, pp. 45–45.

[56] ——, "A toolbox for easily calibrating omnidirectional cameras," in *Proc. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2006, pp. 5695–5701.

[57] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[58] T. Foote, "tf: The transform library," in *Proc. 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, ser. Open-Source Software workshop, April 2013, pp. 1–6.

[59] C. Gamallo, M. Mucientes, and C. V Regueiro, "A fastSLAM-based algorithm for omnidirectional cameras," *Journal of Physical Agents*, vol. 7, no. 1, pp. 12–21, 2013.

[60] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[61] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. 2003 IEEE International Conference on Computer Vision (ICCV)*, Oct 2003, pp. 1403–1410 vol.2.

[62] S. Thompson and A. Zelinsky, "Accurate local positioning using visual landmarks from a panoramic sensor," in *Proc. 2002 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 2656–2661.

[63] N. M. Kwok, G. Dissanayake, and Q. P. Ha, "Bearing-only SLAM using a SPRT based Gaussian sum filter," in *Proc. 2005 IEEE International Conference on Robotics and Automation (ICRA)*, April 2005, pp. 1109–1114.

[64] T. Lemaire, S. Lacroix, and J. Sola, "A practical 3d bearing-only SLAM algorithm," in *Proc. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2449–2454.

[65] K. E. Bekris, M. Click, and E. Kavraki, "Evaluation of algorithms for bearing-only SLAM," in *Proc. 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1937–1943.

[66] L. Juan and O. Gwun, "A comparison of SIFT, PCA-SIFT and SURF," *International Journal of Image Processing (IJIP)*, vol. 3, no. 4, pp. 143–152, 2009.

[67] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proc. 2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 2564–2571.

[68] W. E. Hoover and M. Rockville, *Algorithms for Confidence Circles and Ellipses.* U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, National Ocean Service, 1984.

[69] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692