STUDY ON FEATURE EXTRACTION ALGORITHMS FOR USE IN INDOOR SIMULTANEOUS LOCALIZATION AND MAPPING

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Electrical Engineering)

at the

NEW YORK UNIVERSITY
POLYTECHNIC SCHOOL OF ENGINEERING

by

Agraj Jain

May 2015

Approved:

_____
Advisor

_____
Date

_____
Department Head Signature

_____
Date

Copy No. # 1

University ID: N19443982

VITA

March 1991 ..... Born

June 2009 ..... Graduated High School from Sri Kumarans Public school, Bangalore

May 2013 ..... Bachelor of Technology in Electrical Engineering from National Institute of Technology Karnataka, Surathkal

May 2015 ..... Master of Science in Electrical Engineering from NYU Polytechic School of Engineering, New York

ACKNOWLEDGEMENTS

I would like to thank Professor Farshad Khorrami for his advisement during the course of this project, as well as his invaluable guidance throughout my masters program which enabled me to focus on control and robotics. The in depth conceptual understanding of various aspects of building a robot, has given me valuable experience and insight for which I am grateful.

I would also like to thank Dr. Krishnamurthy, for his guidance and insights about various aspects of implementing the algorithms which enabled me to move forward whenever I was stuck. I would also like to thank my colleagues Griswald Brooks and Alaina Herkelman for helping in the construction of the testing platform and designing the data acquisition process.

I would like to dedicate this work to my loving mother,father, and grandparents for their support through all my endeavors

ABSTRACT

STUDY ON FEATURE EXTRACTION ALGORITHMS FOR USE IN INDOOR SIMULTANEOUS LOCALIZATION AND MAPPING

by

Agraj Jain

Advisor: Prof. Farshad Khorrami

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (Electrical Engineering)

May 2015

This thesis provides the implementation and testing of feature extraction algorithms for Extended Kalman Filter based indoor Simultaneous Localization and Mappingon a dedicated integrated mobile platform. The thesis details the concepts behind EKF and it's application to SLAM. The features of the mobile platform and it's suitability to act as a testing platform for SLAM is discussed. A simple point feature extraction algorithm is discussed along with it's results and drawbacks. Two linear feature extraction algorithms are discussed, both conceptually and practically. The algorithms discussed are plugged into the SLAM algorithm to gain insights about their effectiveness and robustness. The testing is done in both controlled and uncontrolled indoor environments and the results are analyzed

TABLE OF CONTENTS

VITA **i**

LIST OF FIGURES **vii**

LIST OF TABLES **ix**

I. Introduction **1**

1.1 Simultaneous Localization and Mapping 1

1.2 Environment Sensing 3

1.3 Thesis Goal and Outline 4

II. Overview of Extended Kalman Filter Based Simultaneous Localization and Mapping **6**

2.1 Overview of SLAM 6

2.1.1 Pose Prediction 7

2.1.2 Observation and Landmark Extraction 8

2.1.3 Landmark Prediction and Landmark Association 8

2.1.4 Filter Update 9

2.2 Overview of the Extended Kalman Filter 10

2.2.1 Kalman Filter 11

2.2.2 Extended Kalman Filter 12

2.3 Application of Extended Kalman Filter for SLAM 13

III. Hardware Design and Development of an Integrated Mobile Platform **16**

3.1 Overview and Design Goals 16

3.2 Mechanical Design 17

3.3 Electrical Design 20

3.3.1 Device purpose and descriptions 20

3.4 Software Design for data collection 24

3.5 Encoder Based Odometry for State Estimate 24
3.5.1 Motion model and the corresponding differentials 24
3.5.2 Limitations of odometry based estimation 27

IV. 2D LIDAR Feature Extraction for SLAM 28
4.1 Feature Extraction with Point Features 28
4.1.1 Overview 28
4.1.2 Feature Extraction Algorithm 28
4.1.3 Examples 30
4.1.4 Measurement Model and the corresponding Jacobian matrices 31
4.2 Feature Extraction with Linear Features 34
4.2.1 Overview 34
4.2.2 RANSAC based Feature Extraction Algorithm 34
4.2.3 RANSAC Examples 35
4.2.4 Hough transform based Feature Extraction Algorithm 36
4.2.5 Hough Transform Examples 38
4.2.6 Measurement Model and the corresponding differentials 39

V. Implementation and Results 44
5.1 Implementation of EKF SLAM algorithm 44
5.2 Experimental Results 46
5.2.1 Using Point Features 48
5.2.2 Using linear features 50
5.2.3 Using a combination of point and linear features 53
5.2.4 Uncontrolled environment with linear features 55

VI. Conclusion 59

References 60

LIST OF FIGURES

1 Block Diagram Of Probabilistic SLAM Process 6

2 The Integrated Mobile Platform 16
3 Dimensions of Integrated Mobile Platform 18
4 System Diagram of Integrated Mobile Platform 20
5 Integrated Mobile PlatformPower Routing Diagram 21
6 Low pass filter for Encoder Input 22
7 Motion model of Differential drive platform 25

8 Simulated Arena for LIDAR 29
9 Differential based LIDAR feature detection 29
10 Real world arena 30
11 Spurious cylinders detected in data 31
12 After filtering of candidate cylinders 31
13 Example of walls detected by RANSAC 36
14 Example of spurious walls marked by RANSAC 36
15 Example of walls detected by Hough Transform 39
16 Example 1 of Hough Transform being robust to people 39
17 Example 2 of Hough Transform being robust to people 40
18 Normal form of a line 41
19 Line in robot frame and world frame. 41

20 SLAM Implementation using only LIDAR 45
21 Controlled environment and path reconstructed by an overhead camera. . 47
22 Controlled environment and path reconstructed by detecting point features in LIDAR data. Path starts from the origin. 49
23 Controlled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin. 52

24 Controlled environment and path reconstructed by detecting both point and linear features in LIDAR data. Path starts from the origin 54
25 Passages forming an uncontrolled environment 55
26 Obstacles in an uncontrolled environment 56
27 Uncontrolled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin. 57

LIST OF TABLES

1 Errors in environment reconstruction using point features. 50
2 Errors in environment reconstruction using linear features. 53
3 Error in environment reconstruction using combination of linear and point features 55

CHAPTER I

Introduction

In this thesis different algorithms to extract features from 2D scanning laser range data is studied in the light of their use for environment sensing in Simultaneous Localization and Mapping(SLAM) applications. These algorithms are tested on a custom built platform dedicated for SLAM. In this chapter the motivation for SLAM along with some of the existing research in presented first. The next section introduces the importance of feature extraction in SLAM. The last section gives an overview of the rest of the thesis.

1.1 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) addresses the problem of a robot moving through a previously unknown environment. The robot makes relative observations of it's self-perception and of the environment, both of which have some component of noise. To reconstruct a map of the environment and the path taken by the robot is the primary goal of SLAM. SLAM is a key requirement for any truly autonomous robot. There has been a lot of research about SLAM in the last decades. The continuous improvement of computational capacity of computers and reduction of their size, has made it possible for SLAM applications to be performed outside of pure test environments and even outdoors [1–3].

SLAM is extremely useful for mobile robots traveling in unknown surroundings. It lets mobile robots to help in exploration by going places that are too distant [4] or dangerous [5]. Especially robots that need to operate in extreme environments such as undersea, underground or on other planets, must be capable of SLAM. Even in simple environments such as the inside of a building accurate prior maps are difficult to find and the robot has to construct one of it's own. The maps produced by SLAM can either be used on their own like done in [5] or [6] where SLAM is used to map tunnel

systems.Or more typically serve as starting points for motion planning or autonomous navigation [7–12].

Initial papers on SLAM all considered the 2D case. The extension of these algorithms to full 3D is mathematically straightforward. However, the added complexity makes SLAM algorithms in 3D more computationally expensive. In recent years more algorithms capable of 3D SLAM were introduced [31–33].

Most algorithms use odometry to calculate the initial estimate of the robot pose. This pose estimate is refined by using measurements of the environment, made by the other sensor. The development in the robotic community has lately shifted towards the use of sensors that are cheaper, lighter, smaller and thus often less accurate. This change allows for a use of SLAM on smaller weight constrained robots such as Unmanned Aerial Vehicles to a bigger extent [13] and also increases the potential of getting SLAM based consumer products, e.g. autonomous vacuum cleaners, on the market.

If an accurate map of the environment was already available, estimating the path of the robot would be a simple localization task [14]. Similarly if the true path of the robot was known, building a map of the environment is straightforward [15] [16]. However when both are unknown the problem is challenging.

The cyclic nature of the relationship between localization and mapping is a result of how errors in the sensors readings are corrupted by error in robot's motion estimate. As the robot moves, the estimate of it's pose is corrupted by motion noise. The estimate of the environment objects are in turn corrupted by both the measurement noise and the error in the estimated pose of the robot. That is, the error in robot's path correlates errors in the map. Hence the true map cannot be obtained without also reconstructing the path of the robot *simultaneously*.

The numerous SLAM algorithms can be broadly divided into probabilistic approaches and scan-matching methods. The probabilistic approaches estimate a probability distribution that describes the joint posterior density of both the robot pose and the map. This probability distribution can be estimated by an Extended Kalman Filter (EKF) [17–20], an Unscented Kalman Filter (UKF) [21–24] or by Rao-Blackwellized Particle Filters (RBPF) [25–29]. The scan-matching methods build a map by merging consecutive scans of the environment. Optimization algorithms are used to optimally align two consecutive scans. The pose increment of the robot is found by calculating the rotation and the translation that were applied to the two scans in order to align them. Scan-matching SLAM algorithms all have the Iterative Closest Point (ICP) algorithm

at their core [30].

The main advantage of ICP over the other (probabilistic) SLAM algorithms is the relatively high computational speed and the ease of implementation. It does however, not provide any information on the likelihood of the estimated pose or the map. The disadvantage of using the Extended Kalman Filter for SLAM is that linearized versions of the motion and observation models are used, which can cause estimation errors. Neither the UKF nor the RBPF have this problem. Compared to the UKF, the RBPF is better at handling non-Gaussian noise. In spite of this, Extended Kalman Filter is still used in a large number of applications due to it's simplicity.

1.2 Environment Sensing

As seen in the last section, an important part of the SLAM algorithm depends on sensing the environment around the robot and creating a representation of the environment that the robot can understand. This is the aspect of SLAM that is the focus of this thesis. For this a large number of sensors can be used based on the type of robot and the type of environment, the robot is expected to map.Most commonly used sensors include laser scanners [34–38], monocular cameras [39–45], stereo vision systems [46–50] and 3D cameras such as Microsoft's Kinect [51–53]. Some environments such as underwater or underground require more specialized sensors such as SONAR or RADAR [54–57]. It is also common to use multiple sensors and combine their data to get a better representation of the environment for example in [58], data from a SONAR is combined with a single camera image to use in SLAM. Algorithms to do that are constantly being developed and are called sensor fusion algorithms which is an active area of research by itself [59–61].

The sensing of the environment is just the first step. To use the environment data, the information collected needs to be analysed. The sensor that is the focus of this thesis is the range finders (LIDARs). There are two basic approaches to mapping with LIDARs: feature extraction and scan matching. The first method extracts features (also called landmarks) from the LIDAR data. These features are added to the map maintained by the robot which lets it use their locations for localization. The features used often depend on the environment: in indoor settings, lines, corners and curves have been used [62–65]. Outdoors, the hand-written tree detector originally developed for the Victoria Park dataset [66], has been used almost universally [67–69]. Naturally, tree detectors work poorly in offices, and corner detectors work poorly in forests. The lack

of a general-purpose feature detector that works well in varied environments has been an impediment to robust feature-based systems. The alternative LIDAR approach, scan matching, directly matches point clouds. This approach dispenses entirely with features and leads to map constraints that directly relate two poses. Scan matching systems are much more adaptable as their performance does not depend on the world containing straight lines, corners, or trees. But scan matching has a major disadvantage, it tends to create dense pose graphs that significantly increase the computational cost of computing a posterior map.

Feature-based methods have an additional advantage, searching over data associations is computationally less expensive than searching over the space of rigid-body transformations which would be needed in Scan matching algorithms. As the prior uncertainty increases, the computational cost of scan matching grows. While the computational complexity of feature-based matching is nearly independent of initialization error. And finally, feature-based methods tend to work better outdoors, because they often reject ground strikes that result when the robot pitches or rolls. In short, feature-based methods would likely be preferable to scan matching if they were able to offer the same robustness and broad applicability to different environments.

1.3 Thesis Goal and Outline

The primary purpose of this thesis is to develop and test a standard scalable platform to test different feature extraction algorithms. To allow this the SLAM algorithm itself is chosen to be an Extended Kalman Filter algorithm based on simplicity and it's amenability to modularize the various components of SLAM. The major advantage of this algorithm is that it is very simple to implement on an on board computer and can be used for real time SLAM applications. This algorithm along with it's mathematical formulation is explained in Chapter II.

The next part of the thesis consists of designing a robot custom built for this particular purpose. For testing a basic algorithm like EKF, a classic experimental platform is used as a starting point which is a differential drive based ground vehicle. It is also equipped with an autopilot including an Inertial Measurement Unit, an on board computer as well as exteroceptive sensors such as a 2D LIDAR and camera. Hence it can be scaled to test a large variety of Algorithms in both off-line and real time. This experimental platform is further described in Chapter III

To test the designed system, two feature extraction methods are implemented. The sensor used is a scanning laser scanner that gives us 2 dimensional information about the environment. When initially trying to use the range data from the LIDAR, a simplistic algorithm is to use rapid changes of the range as features. While this is a simple algorithm used for trying out SLAM for the first time, it has several drawbacks. This algorithm to detect point features and also modifications to improve it are explored in the first part of Chapter IV.

A number of studies have shown that for indoor SLAM, which is the target environment in this thesis, linear features such as walls and doors are found to give much better results [70]. There are a variety of methods [71] to find these linear features. To obtain a good understanding of how to extract these features a modification of the well known RANSAC algorithm [72] is implemented from scratch in python. Another common method to extract linear features is the Hough transform which is claimed to give much better performance [73]. Outside of understanding the method, for practical purposes it is usually advantageous to use optimized and parallelized implementations of the algorithms. So the LIDAR range data is rasterized into am image and the hough transform provided in the OPENCV library is used and walls are extracted from the LIDAR data. These methods of extracting linear features are discussed in the second part of Chapter IV

Both linear and point features are plugged into the feature detection of the EKF SLAM algorithm both individually and in combination and the results are presented in chapter V.

CHAPTER II

Overview of Extended Kalman Filter Based Simultaneous Localization and Mapping

2.1 Overview of SLAM

Simultaneous Localization and Mapping, commonly abbreviated as SLAM, is concerned with the problem of a mobile robot building a map of the unknown environment, while at the same time navigating the environment using the map. A common way to formulate the SLAM problem is that a robot executes controls and accumulates observations of it's environment, both corrupted by noise. The noisy readings can be modeled as probability distribution functions. Both the pose of the robot and the location of the environment objects can also be considered to be a probability distributions. Initially the variance of the robot pose is small because it knows where it is, but it increases when it moves. The objects are assumed to have a large variance the first time they are seen, but with subsequent re-observation, the assurance of their position increases reducing the variance of it's probability distribution. This approach to SLAM is called probabilistic SLAM.

Most probabilistic SLAM methods follow the same structure. At every discrete

Figure 1: Block Diagram Of Probabilistic SLAM Process

time step k, the inputs to the SLAM algorithm are: the control input to the robot (u_{k-1}) and the observation made by the sensors (z_k). The desired outputs of the SLAM algorithm are: the estimated robot pose(x_k), and the positions of landmarks or a map, represented by m_k . The general structure every probabilistic SLAM algorithm follows, is represented by the block diagram in Figure 1. First, the current pose estimate is calculated according to the motion model of the robot. This model uses the previous pose estimate and the last control input to compute the new estimated pose represented by x_k^-. Then, the current measurement is estimated by an observation model. This observation model uses the current pose estimate and the previous landmark positions to make a prediction of the current measurement represented by m_k^-. To obtain the error signal, the estimated measurement is subtracted from the real measurement z_k. This error signal,represented by e_k, together with the pose estimate and the previous landmark positions is used in the SLAM update step. The update step is used to improve the pose estimate and to update the landmark positions. This is usually done by some filter, such as Rao-Blackwellized Particle Filter (RBPF), an Extended Kalman Filter (EKF), or similar. After this step, new landmarks can be added to the map.

2.1.1 Pose Prediction

The first step of SLAM is the prediction of the robot's state. The prediction is based upon known information about the robot's own motion model. The information gathered about the robot's movement must also be modeled in order to relate it mathematically to the filter used. This information about the robot's movement can be gathered using sensors such as encoders or by using the control signals sent to the robot's localization system or a combination of both.These sensors which estimate the movement of the robot are termed *proprioceptive* sensors.

During the prediction step, the uncertainty of the robot's position is also updated. The maximum error can be approximated based on the movement of the robot and a predetermined error model. This error accumulates, and the uncertainty in the robot's position accumulates as well. Based on the SLAM implementation and the type of filter used,the computing of this uncertainty varies. The error and motion models change based on the expected environment and the robot's specific platform and hardware.

2.1.2 Observation and Landmark Extraction

The next step in the SLAM process can actually be broken into two sub-steps, environmental observation and landmark extraction. Both of these sub-steps are entirely dependent on the type of environmental sensor hardware and type of map used in the SLAM implementation. The type of filter used in the SLAM implementation does not typically affect these sub-steps, which only provide data to the next step. In Figure 1, this step provides the measurement z_k.

Environmental observation occurs by a robot observing the environment with one or more types of exteroceptive sensors. The type of sensor used can vary from range measurement to vision, and the common factor for all sensors is their ability to make useful observations of the environment suitable for landmark extraction. However, the type of data gathered can vary widely from sensor to sensor.

Landmark extraction is the process of reliably extracting landmarks from the observations for the purpose of inserting them into a map or matching them with other previously stored landmarks in a map. Landmark extraction is not only dependent on the type of sensors used but also on the type of map used in the implementation. Based on the observations gathered from the robot's sensors. Because of this, landmark extraction is vitally important to the success of SLAM. Failure of the landmark extraction step can be destructive to the success of the algorithm, even if all other steps of the process are working correctly.

2.1.3 Landmark Prediction and Landmark Association

Landmark Prediction is the process of estimating the locations of all the previously observed landmarks as per the new pose of the robot. This depends on the kind of landmarks being used and their representation as per the SLAM algorithm being used. It also depends on the movement of the sensor with respect to the robot if there is any.

Landmark association is the process of matching observed landmarks with those previously stored in the map. This process is also referred to as re-observation of landmarks. Extracted landmarks that are not matched to corresponding landmarks in the map are considered newly observed. These newly observed landmarks are sent to the map insertion process. Associated landmarks are conversely used in the filter update step.

Practical implementation of landmark association is challenging due to a number of problems that can come up. First, an observable landmark may not be associated with

its map counterpart. This problem is called a false negative and can occur either because the landmark is not extracted properly from the sensor data or because the extracted landmark cannot be easily matched to the stored landmark. If this problem occurs and can be detected, the corresponding landmark is usually not used in the algorithm. Second, a landmark may be observed once and never observed again. This is a problem mainly because, a useless landmark takes up memory space and affects the execution time of the algorithm. These landmarks should not be used further in the algorithm too. These problems can be solved by minimizing the number of bad landmarks extracted. Redefining a more suitable landmark extraction policy usually helps.Another problem that is common is a false positive, where a landmark is wrongly associated to another previously observed landmark. This problem is the most destructive to the SLAM algorithm, because the robot localizes relative to the wrong stored landmark which will affect all the future time steps. This error keeps growing. Data association, like landmark extraction, is dependent on the type of map used.

Another important observation, is that the two processes involved in this step can be interchanged depending on application and processing capabilities. That is if the landmarks detected are represented in inertial frame of reference they can first be associated to existing landmarks. That way only the positions of the associated landmarks need to be predicted reducing processing time. This approach must be implemented after due consideration depending on the kind of landmarks being associated. For some types of landmarks, such an optimization will result in an increase in the occurrences of false positives.

2.1.4 Filter Update

The filter update step is the most important step in any SLAM implementation. A filter typically is used to remove unwanted components from a signal. In SLAM, the unwanted component is the error (noise) from localizing using the proprioceptive sensors. The goal of the filter update step is to use the data from the prediction, observation, landmark extraction, and data association steps to remove the errors in both the robot's estimated pose as well as from the landmark's estimated locations. The filter update step will vary based on the type of filter used. Regardless of the type of filter used, the update step is important. Also, unassociated landmarks from the data association step are as mapped to inertial frame of the robot and then added to the map in this step. Again, this step will vary based on the types of landmarks used and the type of map

provided.

The conclusion of this step completes a single iteration of the SLAM process. While some of these steps are dependent on each other, not all of the steps will occur in a single iteration. Specifically, data association will not occur if the landmark extraction step is unable to extract any landmarks. Likewise, the filter update step will not occur if data association is unable to match any landmarks to the map. Map insertion is also dependent on data association, such that if all extracted landmarks are matched to landmarks in the map, then no new landmarks need to be added.

2.2 Overview of the Extended Kalman Filter

The filter that is implemented in this thesis is the Extended Kalman Filter. The Kalman Filter and the Extended Kalman Filters are the main representatives of the most popular Gaussian filters family. The main difference between these two filters is the type of systems that they can be applied to. The Kalman Filter deals with linear systems whereas the Extended Kalman Filter is a direct extension of the Kalman Filter to the non-linear case.

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.1}$$

It is discussed in the previous section how, in probabilistic SLAM the states and the inputs are all represented as probability density functions. Gaussian filters model the states as normal or Gaussian distributions of the form shown in Equation (2.1), where μ is the mean value or the expected value of x and σ is the standard deviation. σ^2, which is called the variance gives an extent of the uncertainty in the expected value. This is sometimes a limitation as the noise entering the process or the measurement might not be close to normally distributed. Still, Gaussian filters are the most prevalent filters for SLAM.

It is important to realize that each of the variables, in all the equations being discussed, are usually vectors or matrices and not scalars. The dimension of the vector will depend mainly on whether the SLAM process being considered in 2 dimensional or 3. The covariance matrix Σ contains the correlation between the components of the sate vector.

Before describing how the Extended Kalman Filter can be applied towards a SLAM algorithm, some preliminaries need to be provided on Kalman and Extended Kalman Filters such as the main ideas and formulas behind them.

2.2.1 Kalman Filter

The popular Kalman Filter [74, 75] consists of mathematical equations that give an efficient computational recursive solution of the least squares problem. The filter has several advantages such as: it supports estimation of the past, present and future states, even when the precise nature of the system is not known.

In general it addresses the problem of trying to estimate the state $x \in \Re^n$ of a discrete time process described by the linear stochastic equation

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.2}$$

with a measurement $z_k \in \Re^m$ which is given by,

$$z_k = Hx_k + v_k \tag{2.3}$$

The random variables w_k and v_k represent the process and measurement noise respectively. They are assumed to be independent of each other and have a normal probability distribution functions given by Equation (2.4) and Equation (2.5).

$$p(w) \approx N(0, Q), \tag{2.4}$$

$$p(v) \approx N(0, R) \tag{2.5}$$

In practice, the process and measurement noise covariances, Q and R matrices might change at every time step. They are assumed constants for now.

The equations of Kalman filter can be classified in to be in 2 groups. *Prediction* equations and *Correction* equations. The former are responsible for projecting forward in time the current state and error probabilities and the latter are responsible for adjusting the projected estimate by an actual measurement at that time. Hence the Kalman filter estimates a process by a form of feedback control.

Defining the predicted state estimate as $\hat{x}_k^- \in \Re^n$ and the corrected estimate to be $\hat{x}_k \in \Re^n$ the actual equations for the prediction are given by (2.6) and (2.7) where A and B are from Equation (2.2) and Q is from equation (2.4) [75].

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.6}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.7}$$

The first step in the correction step is to find a *gain* or *blending factor* that minimizes the error covariance. This is found by equation (2.8) [74, 76–78]. The next step is

to actually measure the process to get z_k and generate a better estimate using Equations (2.9) and Equation(2.10) [75].

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{2.8}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.9}$$

$$P_k = (I - K_k H) P_k^- \tag{2.10}$$

After each prediction and correction step, the corrected state found is used as an initial estimate for the prediction step of the next time step. This recursive nature is one of the most appealing feature of Kalman filters.

2.2.2 Extended Kalman Filter

In Section 2.2.1, the Kalman filter gives us a set of equations to estimate the state $x \in \Re^n$ of a discrete time process described by a set of *linear* equations. But, most applications including SLAM, consists of systems governed by *non-linear* equations. The Extended Kalman filter is the technique of linearizing a non-linear dynamics around the current mean and covariance for use in a Kalman filter.

Similar to the Taylor series, the estimation around the current estimate can be linearized using the partial derivatives of the process and measurement functions. To do this the equations in Section 2.2.1 are slightly modified. The initial assumption is still that the process has a state vector $x \in \Re^n$, but it is described a *non-linear* stochastic difference equation (2.11). This equation relates the state at time step x_{k-1} and x_k. The measurement $z \in \Re^m$ is given by equation (2.12) where h is also *non-linear*.

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{2.11}$$

$$z_k = h(x_k, v_k) \tag{2.12}$$

Here, the random variables w_k and v_k are again the process and measurement noise as in equations (2.4) and (2.5). Similar to the Section 2.2.1, there are 2 groups of equations. The prediction step is given by Equation (2.13) and Equation (2.14).

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{2.13}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{2.14}$$

Here it is seen that the first equation is the same as Equation (2.11), but the w_k variable is replaced by zero as in practice it is not possible to actually know the noise and $\hat{x}_k^-$ is just an estimate. The second equation is similar to the Kalman filter equations except here A and W are Jacobian matrices of partial derivatives with respect to x and w respectively. At each time step they are recalculated according to equations (2.15) and (2.16)

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{2.15}$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{2.16}$$

As with the basic Kalman filter ,the equations for the correction step given by (2.17) to (2.19) correct the estimate of the state and covariance based on a measurement z at time k

$$K_k = P_k^- H_k^T ({}_kH P_k^- H_k^T + V_k R V_k^T)^{-1} \tag{2.17}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{2.18}$$

$$P_k = (I - K_k H_k) P_k^- \tag{2.19}$$

As in the previous case, h is from Equation (2.12) and v_k is approximated to zero for finding an estimate of state. In the *Kalman gain* and covariance equations, H and V are again Jacobian matrices of the partial of h with respect to x and v respectively. They are also recalculated each step using equations (2.20) and (2.21)

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{2.20}$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{2.21}$$

2.3 Application of Extended Kalman Filter for SLAM

Now, with the understanding of the generic Kalman filter and Extended Kalman Filter equations, it is possible to concretely define the EKF solution for SLAM. As the thesis primarily deals with 2D SLAM, the equations presented in this section are customized for that application. The state that is to be estimated consists of both the

pose of the robot, and the (x, y) locations of all the landmarks as shown in Equation (2.22). Since the pose itself consists of both location (x, y) and orientation(yaw), the state vector will have a length of $(3 + 2n)$ if there are n landmarks observed at that particular time step.

$$\mu = \begin{bmatrix} x_R \\ y_R \\ \theta_R \\ x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{2.22}$$

Going by the same procedure in Section 2.1, the first step is the prediction of the robot pose using it's motion model. This motion model is the same as the system that is being considered in Equation (2.11). The motion model is represented by a non linear function g and the prediction step is based on equations (2.13) and (2.14). The process error Q from Equation (2.14) is modeled as a measurement error in either the control or the proprioceptive sensors used and is represented by $\Sigma_{control}$. Hence to find the Jacobian of the motion model with respect to the noise, it is sufficient to differentiate it with respect to the either the control input or the readings of the dead reckoning sensors. This is represented in Equation (2.23).

$$\bar{\mu}_t = g(\mu_{t-1}, u_t) \tag{2.23a}$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t \Sigma_{Control} V_t^T \tag{2.23b}$$

$$G_t = \frac{\partial g}{\partial state} \qquad V_t = \frac{\partial g}{\partial control} \tag{2.23c}$$

Now that an estimate of the robot pose at time k is available, it needs to be improved. For this a measurement of the environment using any exteroceptive sensor on the robot is used. This measurement is got in the *Landmark Extraction* step. A wide variety of sensors can be used such as a laser range finders and cameras depending on both the environment and the robot. Whatever sensor is used, the feature or landmark

which is to be used as measurement is extracted and it's relation to the pose of the robot is represented by a measurement model represented by h. The same measurement model can be used to predict the measurement of existing landmarks. Once the land marks are predicted and associated the error or the *innovation* is calculated by the difference $z_k - h(\hat{x}_k^-, 0)$ from equation (2.18).

To calculate the Kalman gain as per (2.17), the Jacobian matrices H and V are required. As well as R which is the noise covariance. To find H, the measurement model is first differentiated with respect to the robot pose to get the first part of it. For the rest of the Jacobian the measurement model is differentiated with respect to the rest of the state vector which contains some representation of the environment already observed by the robot. Hence the measurement model is differentiated with respect to the landmark locations. The Jacobian V which represents how the measurement changes with respect to other landmarks can be assumed to be an identity as each landmark is assumed to be independent of the other. The noise covariance error represented by Q_k depends on the type of sensor being used. The SLAM update equations with these considerations can be simplified as shown in Equation (2.24).

$$K_t = \bar{\Sigma}_t H_t^T (H_t.\bar{\Sigma}_t.H_t^T + Q)^-1 \tag{2.24a}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \tag{2.24b}$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{2.24c}$$

Using these equations, both the pose of the robot and the locations of all the objects can be updated simultaneously.

CHAPTER III

Hardware Design and Development of an Integrated Mobile Platform

3.1 Overview and Design Goals

The Integrated Mobile Platform (IMP) is designed as a robust, scalable platform for Simultaneous Localization and Mapping in a variety of complex environments along with capability for autonomous navigation and decision making. IMP is designed as a platform that can be used both for deployment as well as research. Hence, along with robustness, modularity is also emphasized in the design. It is a mobile robot which can be used either as an independent agent or as a long range mobilization unit in a cooperative team, requiring long range capability and considerable payload capacity.

For the primary purpose of SLAM, the major requirements of such a platform are:

Movement Has to be able to move with a variety of speeds and relatively small turning radii as it is to be used indoors.

Self-position acknowledgment Has to have proprioceptive sensors which give an estimate of it's own position and orientation.

Environment sensing Has to have sensors to understand the environment.

On-board computer Has to have sufficient on-board processing power to do the computations necessary for SLAM.

Figure 2: The Integrated Mobile Platform

Real time controller Has to have a capability to implement real time control.

Communication Has to have robust communication between the different components and also with the ground station.

Memory Has to have sufficient on-board memory to collect data to enable testing of SLAM algorithms off line.

Flexibility The various components both hardware and software need to be designed in such a way that it is easy to switch them around.

While these requirements define the bare minimum that is needed for such a platform, the design goals that guided the development of IMP were more broad. The design goals were as follows:

- Have a hardware design capable of supporting large payloads and which can be scaled for any desired run time.
- Have enough computational power for implementing real time algorithms for autonomy.
- Design modular hardware and software interfaces for the sensors to allow testing of different sensors and sensor fusion algorithms.

With these goals in mind, IMP was designed as a six wheeled differential drive platform with a multi processor control system, including two micro-controllers and an on-board computer and equipped with a scanning laser range finder and monocular camera.

The following section discusses the design process involved and the resulting features of IMP, first with respect to the hardware components. The next section discusses the major sensory, computational and power handling components of IMP. The last section discusses the communication and information flow in the current implementation of using IMP for data collection for SLAM. The motion model for EKF SLAM using this platform is discussed in the last section.

3.2 Mechanical Design

The first design decision was the basic configuration of the robot. The primary motivation being simplicity, the type of robot was decided to be a differential drive robot,

(a) Isometric view (b) Top view (c) Side view

Figure 3: Dimensions of Integrated Mobile Platform

is a mobile robot whose movement is based on at least two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion. If both the wheels are driven in the same direction and speed, the robot will go in a straight line. If both wheels are turned with equal speed in opposite directions, the robot will rotate about the central point of the axis. It has several advantages over other systems. Ackerman steering which is used in automobiles, have a lower bound on the turning radius and also require additional control inputs for steering. In other systems such as tank treads, it is difficult to reconstruct the odometry, and as the target of the robot is autonomy, this proves a disadvantage. A differential drive robot gives on the spot turning capability with minimal control.

Once the type of robot was decided, the next important consideration was the size. This mainly depends on the budget and the application the robot is being built for. As per the design principles stated, the primary objective is to keep the robot scalable, and equipped to carry larger payloads. So the slightly larger size of the robot will allow payloads to be attached in the future. The dimensions of the robot are shown in Figure 3. The chassis is designed to be light weight but structurally sound so as to allow larger payload capability. For this purpose the material of choice is aluminum. It has the lightness offered by plastics and none of the structural weaknesses exhibited by it. The chassis structure itself is an hollow cuboid, but there are multiple support struts running across it's breadth to give added strength. This design allows us to minimize the weight of the chassis to about 2.2 kg.

The guiding principle behind the design of the robot was versatility. Therefore, the wheels were chosen to be made of rubber with a offset-V tread to have sufficient friction

to traverse both indoor and outdoor environments. The wheel size was decided based on the size of platform. The number of wheels were chosen based on both stability, control and sensing requirement. The four outer wheels were decided to be powered using a DC geared motor so that sufficient torque can be generated to allow the robot to have a small turning radius. Putting the encoders on the same wheels will give errors in odometry when there is slipping of the wheels because of friction loss which is a common occurrence on gravel in outdoor environments and on smooth floors indoors. Due to this the two middle wheels were added, which are used for odometry. Since it is only the movement of the robot which will cause the middle wheels to turn, it gives a better estimate of the path followed by the robot which gives a huge benefit, for autonomous applications.

Once the basic platform is designed, the choice of motors and battery is a combined decision, the torque capacity of the motors depend mainly on the weight of the payload that is to be carried by the robot. More the torque capacity the current drawn is more hence for the same run time a larger battery is needed which in turn increases the weight of the payload. For example, in a previous version, the same platform had a 24V nickel-metal hydride(Ni-MH) battery and DC geared motors with a rated torque of 1.4 kgf-cm. Since a typical Ni-Mh battery has a specific energy of 20-120 Wh/kg, a battery which had an energy rating of 4500 mAh, weighed around 1.5 kg. With this configuration, it was found that the motors were not strong enough to turn just the platform itself with a reasonable turn radius. This was due to a combination of both the large weight of the battery and the low torque of the motor.

Both the batteries and the motors were then upgraded, to enable the platform to carry not only it's own weight but also additional sensor and computational payload. Ni-MH batteries were replaced by high specific energy lithium polymer(LiPo) batteries. These batteries typically have a specific energy of 100-265 Wh/kg. The battery used on IMP with 2500mAh, weighs around 0.4 kg. Even though the battery capacity has reduced by a factor of 2, the weight reducing by a factor of 3 compensated for it. Also, the low weight allows further increase of the battery capacity without having to change the motors if needed. The motors currently used on the Integrated Mobile Platformhave a rated load of 7.3 kgf-cm, which makes the IMP ideal for larger payloads.

Figure 4: System Diagram of Integrated Mobile Platform

3.3 Electrical Design

The electrical design is guided by the same principles of Scalability for autonomy, and flexibility. The major computational and sensory payloads on the IMP are:

- Dual processor AutoPilot unit with a 12-axis inertial measurement unit
- ODROID-XU Computer
- MA 3 Miniature Absolute Magnetic Shaft Encoders
- Leopard Imaging 5 MP camera
- Hokuyo-URG
- TP-Link Nano Wireless Module
- XBEE Wireless radio

3.3.1 Device purpose and descriptions

AutoPilot

The AutoPilot is a dual processor unit that is equipped for low level control algorithms as well to act as an input/output interface for the robot. The AutoPilot consists

Figure 5: Integrated Mobile PlatformPower Routing Diagram

of two processing units: a TM4C and RM48 micro-controller (MCU), which operate at 80 MHz and 220 MHz, respectively. These processors can communicate with each other either over UART or Controller Area Network(CAN). One UART of the RM48 MCU is also connected to an on-board computer, an ODROID-XU through a level translator circuit. The AutoPilot is powered via an external 12 V supply. This AutoPilot unit includes a 12-axis inertial measurement unit (IMU) which consists of two, 3-axis accelerometers, one 3-axis rate gyro, and a 3-axis magnetometer unit. Additionally, the AutoPilot includes humidity and temperature sensors and receives GPS signals.

The AutoPilot being a versatile device can be used for a large number of purposes depending on the implementation. A part of the autonomy algorithm can be off loaded to the AutoPilot, or low level control algorithms such as PID can be implemented on it. It is currently implemented as an I/o processing unit. It interfaces with the encoders, the motor driver while acquiring measurements from the IMU.

ODROID XU

The ODROID is the on-board computer. It plays a major part if the IMP is used for real time autonomous navigation. It is equipped with Samsung Exynos5422 Cortex-A15 2.0Ghz quad core and Cortex-A7 quad core CPUs. It has 2Gbyte LPDDR3 RAM capable of operating at 933MHz and several USB 2 and USB 3 ports for interfacing. With these features it can run considerably powerful algorithms for Simultaneous Localization and Mappingand also path planning and decision making. It is the higher level controller of the robot. It is powered by a seperate 5V regulator as shown in Figure 5.

Encoders

Figure 6: Low pass filter for Encoder Input

The magnetic shaft encoders give the absolute position of the wheel. From the perspective of a device interfacing with it, it can be modeled as a potentiometer. It gives 0-5V analog output corresponding to the angle the wheel is at currently. So while the wheel is continuously turning, the value rises from 0 to 5 volts and then suddenly drops to 0. Therefore, to calculate movement the encoder signal has to be first stored in an accumulator and then differentiated. The signal is considerably free of noise and performs reasonably well for dead reckoning, it has some high frequency noise that needs to be filtered out. So the analog signal is first passed through an active low pass filter of 950 Hz which is shown in Figure 6. A Digital to Analog Converter(DAC) is used to be able to set the reference voltage without having to change the hardware. Both the filter circuit and the DAC is inbuilt into the AutoPilot.

Camera

The major advantage of an on-board camera is that it can act both as an exteroceptive sensor as well as a proprioceptive sensor. The camera can be used for sensing

features in the environment. It can also be used for visual odometry which gives an estimate of the robot's pose.

The Leopard imaging Camera, captures 5 mega pixel HD images at speeds up to 30 frames per second. It can has a 72° horizontal field of view, and 30° vertical. It has a simple USB interface and it's data is read into the ODROID.

LIDAR

When picking a LIDAR, first the choice needs to be made between using 2D or 3D LIDARs. 3D LIDARs while having numerous advantages, it's high cost limits the application of the robot. It is more practical to have a 2D LIDAR initially with scope for further expansion into 3D. A really simple way to implement 2D LIDAR is to have a single range finder and mount it on a rotating platform. That way the cost can be reduced but the resolution and scanning speed are adversely affected. An earlier iteration of IMP, used a Piccolo Laser Scanner, which had an inexpensive laser range finder being rotated at around 15 Hz. The range finder itself, was based on beam deviation rather than *Time of Flight* which made it slightly inaccurate.

The current LIDAR used is the Hokuyo URG-04LX Scanning Laser Range Finder which has a low scanning of 100 ms/scan, and is accurate up to 2 mm. It has a 240° field of view and can detect from 20 mm to 5600 mm. It is read through a USB port into the ODROID. With this LIDAR, the robot has the capability of sensing the environment to a largely accurate extant.

Communication Modules

The primary channel of communication is the Wi-Fi module, TP-Link Nano. It has a speed of 150 Mbps and is configured in *AP Router* mode. This is done so that IMP can host its own Wi-Fi to which any ground station that needs to interact with it can connect to it directly. This is advantageous when IMP is to be used outdoors and there is no existing Wi-Fi networks available.

Another interface is the XBee connected directly to the AutoPilot through an UART port. This can be used if the AutoPilot is to be commanded directly for tele-operation without using the ODROID. In the current configuration it is used as a Debug port so that the status of the AutoPilot can be checked whenever needed.

3.4 Software Design for data collection

The Autopilot having a precise Real Time Interrupt module in the RM48, maintains timebase for the whole process. It's interrupts are set in a half millisecond cycles. Every millisecond it reads the encoders and updates an accumulator. The IMU are read every 5 ms. Both the encoders and the IMUs are logged on the SD card at a rate of 80Hz. Every 10 ms, the latest accumulator and IMU readings are sent to the ODROID.

The ODROID, parses the data sent by the AutoPilot. Based on the standard time base, at the rate of 5 Hz, the on-board camera and the LIDAR data is acquired. All the collected data is stored in the file system of the Odroid. These files are then transfered through SSH to the ground station. The ODROID program itself is started and controlled over Wi-Fi, through which the user commands are transmitted.

3.5 Encoder Based Odometry for State Estimate

The encoders attached on the two passive wheels give an estimate of the robot's movement in each time step. The function used to calculate the robot's movement from the encoder data is the motion model of the robot and is used for equation (2.11) in section 2.2.2. Since this robot has only 3 degrees of freedom, the state vector, $x \in \Re^3$ and is defined as $x = [x, y, \theta]^T$ where x and y give the position of the robot from an arbitrary fixed point in inertial frame of reference.

3.5.1 Motion model and the corresponding differentials

To estimate the motion model, first left and right wheel movement is converted into distances traveled through a linear mapping using the known radius of the wheels. These values are represented by $u = [l, r]^T$. The equations are better implemented as a piecewise function. The two cases of when the robot is estimated to be going straight or to be turning is considered separately. This decision is made by observing the distance traveled by the left and right wheels. If they are exactly the same, the robot is going straight and the motion model is given by equations (3.1).

If $r = l$:

$$\begin{bmatrix} \hat{x}^- \\ \hat{y}^- \\ \hat{\theta}^- \end{bmatrix}_k = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{k-1} + \begin{bmatrix} l.\cos(\hat{\theta}_{k-1}) \\ l.\sin(\hat{\theta}_{k-1}) \\ 0 \end{bmatrix} \tag{3.1}$$

Figure 7: Motion model of Differential drive platform

If not, then the robot is turning, and equations (3.2) and (3.3) are used. Where R, α and w are as shown in Figure 7

If $r \neq l$:

$$\alpha = \frac{r-l}{w} \qquad R = \frac{l}{\alpha} \tag{3.2}$$

$$\begin{bmatrix} \hat{x}^- \\ \hat{y}^- \\ \hat{\theta}^- \end{bmatrix}_k = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{k-1} + \begin{bmatrix} \left(R + \frac{w}{2}\right)\left(\sin(\hat{\theta}_{k-1} + \alpha) - \sin(\hat{\theta}_{k-1})\right) \\ \left(R + \frac{w}{2}\right)\left(-\cos(\hat{\theta}_{k-1} + \alpha) - \cos(\hat{\theta}_{k-1})\right) \\ \alpha \end{bmatrix} \tag{3.3}$$

In general for ease of representation this motion model is referred to only by f.

$$[h!]\hat{x}_k^- = f(\hat{x}_{k-1}, u_k) \tag{3.4}$$

Once the motion model is known, it is necessary to find it's Jacobian with respect to the state. Since both the motion model f and the state have 3 dimensions the Jacobian will be a 3×3 matrix given by (3.5).

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} \tag{3.5}$$

Since the motion model is piecewise, it's Jacobian also calculated in two parts by differentiating the respective equations (3.1) and (3.3).

If $r = l$:

$$A = \begin{bmatrix} 1 & 0 & -l\sin\theta \\ 0 & 1 & -l\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

If $r \neq l$

$$A = \begin{bmatrix} 1 & 0 & (R+\frac{w}{2})(\cos(\theta+\alpha)-\cos\theta) \\ 0 & 1 & (R+\frac{w}{2})(\sin(\theta+\alpha)-\sin\theta) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Where R, w and α are according to equation (3.2) and Figure 7

Next the motion model is differentiated with respect to the noise. Here the process noise is assumed to be essentially due to the noise in encoder measurement and it is additive in nature. Hence it is possible to know the variance of the movement with respect to noise by differentiating the motion model with respect to the encoder measurements l and r. Since this is of dimension 2, the noise covariance matrix W will be of dimension 3×2 given by equation (3.8).

$$W = \frac{\partial f}{\partial(u+w)} = \frac{\partial f}{\partial(u)} = \begin{bmatrix} \frac{\partial f_1}{\partial l} & \frac{\partial f_1}{\partial r} \\ \frac{\partial f_2}{\partial l} & \frac{\partial f_2}{\partial r} \\ \frac{\partial f_3}{\partial l} & \frac{\partial f_3}{\partial r} \end{bmatrix} \tag{3.8}$$

Each of the individual terms are then calculated independently. If $r = l$:

$$\frac{\partial f_1}{\partial l} = \frac{1}{2}(\cos\theta + \frac{l}{w}\sin\theta) \tag{3.9a}$$

$$\frac{\partial f_2}{\partial l} = \frac{1}{2}(\sin\theta - \frac{l}{w}\cos\theta) \tag{3.9b}$$

$$\frac{\partial f_1}{\partial r} = \frac{1}{2}(-\frac{l}{w}\sin\theta + \cos\theta) \tag{3.9c}$$

$$\frac{\partial f_2}{\partial r} = \frac{1}{2}(\frac{l}{w}\cos\theta + \sin\theta) \tag{3.9d}$$

$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{3.9e}$$

If $r \neq l$:

$$\frac{\partial f_1}{\partial l} = \frac{wr}{(r-l)^2}(\sin\theta' - \sin\theta) - \frac{r+l}{2(r-l)}\cos\theta' \tag{3.10)a}$$

$$\frac{\partial f_2}{\partial l} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{3.10)b}$$

$$\frac{\partial f_1}{\partial r} = \frac{-wr}{(r-l)^2}(\sin\theta' - \sin\theta) + \frac{r+l}{2(r-l)}\cos\theta' \tag{3.10)c}$$

$$\frac{\partial f_2}{\partial r} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{3.10)d}$$

$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{3.10)e}$$

Where, $\theta' = \theta + \alpha$ and R, w and α are as per Figure 7. Once the Jacobian is calculated, the last component needed for the estimation according to equation (2.14) is the Process noise covariance Q. This has to contain some information about the amount the noise in each time step. Since all noise is assumed to be only sensor measurement noise, a diagonal matrix with the error in each encoder is chosen as the covariance as in equation (3.11).

$$Q = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \tag{3.11}$$

3.5.2 Limitations of odometry based estimation

One of the primary limitation of estimating a robot's position solely on odometry is that, over a long time, the error accumulates and results in the estimate being very far from the actual position of the robot. Also every real world encoder has a component of noise which accumulates over time. In indoor environments, there is a good chance of wheel slippage especially during turns making it hard to accurately reconstruct a turn. Another drawback is that if the encoder wheels are not fully lubricated and free to move, any wheel might stop moving with the robot and skid instead. This will result in the reconstruction seeing a turn, whereas the robot itself might have been moving straight. To correct these, is the major motivation in any SLAM algorithm.

CHAPTER IV

2D LIDAR Feature Extraction for SLAM

To improve the estimate of the state and to get an idea of our environment, the input of the exteroceptive sensors has to be analyzed. As seen in chapter III, the robot is equipped with both a Camera and a 2D scanning LIDAR.

4.1 Feature Extraction with Point Features

4.1.1 Overview

In this section a simple algorithm is used to detect fixed features in the environment. The algorithm is mainly based on large jumps in the range measurement of the LIDAR. The algorithm is then improved using preexisting knowledge of the environment.

4.1.2 Feature Extraction Algorithm

Consider an extremely clean environment. One which has only fixed features and they are all away from the wall. And the entire environment is within the range of the LIDAR. In such an environment the distance readings gradually increase and decrease all along the walls except when they encounter and obstacle or a *feature*. This algorithm essentially looks for large jumps in the distance readings. One such arena is shown in figure 8. The gradual nature of the distance readings are more clearly seen when the scan values are plotted as a function of the angle of the scan beam with respect to the robot as shown in figure 9a.

For the robot to detect the large jumps the distance is differentiated with respect to angles. As seen in figure 9b this will give a specific pattern each time a cylinder is present. A condition can then be designed in the following way. Each time the derivative is larger than a fixed threshold and is a negative number, a cylinder's beginning is found, and when a large positive value is encountered, it's end. Finding the mid point of these two readings the location of the cylinder is found as in figure 9b.

(a) A simulated arena (b) Top view

Figure 8: Simulated Arena for LIDAR

(a) LIDAR scan (b) Detected cylinders

Figure 9: Differential based LIDAR feature detection

It is apparent this method has a large number of drawbacks. The arena has to be fully within the range of the sensor, if not there will be breaks in the distance curve that will generate spurious landmarks. This drawback can be easily overcome by creating a zero order hold for those regions. Also any disturbances in the boundaries will also generate spurious landmarks. The threshold for the derivative is a very sensitive tuning parameter. If it is set too high there is an increased chance of missing landmarks placed closed to walls and if it is set low, there are a large number of spurious landmarks.

To overcome this, based on preexisting knowledge of the nature of features a filter of sorts can be designed. A low threshold is chosen and a large number of prospective features are enumerated. All those that lack a minimum number of points on them are immediately eliminated. Next, based on the fact that the radius of the cylinders that are

the landmarks is known, the angular width of any feature detected can be approximated. From the large set, only the ones within a range around this approximate angular width are retained. Then, the least squares fit is computed to check if the features are indeed cylinders or segments of the wall. Only the features having a large enough curvature are retained. By this method a small amount of robustness is added to a simplistic algorithm of feature detection. The results of such a filter when applied with EKF can be seen in section 4.1.3.

Once the features are detected, the measurement model from the robot to these features need to be calculated to be used in Extended Kalman Filter SLAM.

4.1.3 Examples

In a clean simulated arena the results of the first approach is seen in section 4.1.2. But in a real world environment, a large number of variations will occur. To demonstrate this a relatively clean real world environment is considered as shown in figure 10 The zero values are caused by the entire arena not being within the range of the LIDAR.

(a) Overhead view (b) Polar plot of laser scan

Figure 10: Real world arena

With such an arena if just jumps in distance are used to determine the cylinder position a large number of spurious landmarks are detected as shown in figure 11. These cannot be avoided by simply changing the threshold for the change in derivative. It is possible to avoid some by interpolating the data whenever the distance is zero, but

this introduces additional problems when the robot is too close to an obstacle and if a landmark is too close to a wall.

Figure 11: Spurious cylinders detected in data

Figure 12: After filtering of candidate cylinders

To refine this conditions similar to those discussed in section 4.1.2 are implemented. This is seen to give much better results. as seen in figure 12.

4.1.4 Measurement Model and the corresponding Jacobian matrices

Once the cylinder coordinates are found in the LIDAR data, it's position has to be stored in the map. Since an estimate of the robot's position in the inertial frame is known, the cylinder coordinates are mapped from robot to inertial frame using a

homogeneous transformation as in equation (4.1). Where (x_w, y_w) are coordinates in world frame. (x_r, y_r, θ_r) are estimated pose of the robot and (x_1, y_1)

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & x_r \\ \cos(\theta_r) & \sin(\theta_r) & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{4.1}$$

Once the cylinder is in inertial frame the next step is to check if it corresponds to any preexisting cylinder in the map. For this the euclidean distance between the detected cylinder and the existing cylinder positions is measured and if it is less than a particular threshold then the new cylinder is associated with the existing one.

For the correction of the robot pose as explained in section 2.3, the measurement model of the point features is needed. That is a mapping that gives the laser reading or the distance r and angle α from the robot to the cylinder. For this first the coordinates of the cylinder in the robots frame of reference is found using the inverse of the homogeneous transformation in equation (4.1). But the robot pose used now is the new position at this particular time step. Once (x_1, y_1) is found in the robot frame, the measurement h can be found using equation (4.2). The same equations are used for every the corresponding cylinder that is detected to give z which is used to calculate the *innovation* in equation (2.18).

$$h = \begin{bmatrix} r \\ \alpha \end{bmatrix} = \begin{bmatrix} \sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2} \\ \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) - \theta_r \end{bmatrix} \tag{4.2}$$

The measurement model gives the innovation but not the Kalman Gain. For this, according to equation (2.17) the derivatives of the model with respect to the state vector $x \in \Re^n$ is used. The state contains both the robot pose as well as all the landmarks already existing in the map at that particular time step. Since it is assumed each landmark is independent of the other, most part of the Jacobian H contains zeros except for the part corresponding to the robot pose and if the measurement has been associated with an existing landmark, then it will depend on that landmark's position. Hence for the Jacobian H, the measurement model h is differentiated as per equation (4.3).

$$H = \begin{bmatrix} \frac{\partial r}{\partial x_r} & \frac{\partial r}{\partial y_r} & \frac{\partial r}{\partial \theta_r} & \cdots & \frac{\partial r}{\partial x_1} & \frac{\partial r}{\partial y_1} & \cdots \\ \frac{\partial \alpha}{\partial x_r} & \frac{\partial \alpha}{\partial y_r} & \frac{\partial \alpha}{\partial \theta_r} & \cdots & \frac{\partial \alpha}{\partial x_1} & \frac{\partial \alpha}{\partial y_1} & \cdots \end{bmatrix} \tag{4.3}$$

Each of the terms are derived separately as per equation (4.4).

$$\frac{\partial r}{\partial x_r} = \frac{(x_r - x_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial x_r} = \frac{(y_1 - y_r)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.4)a}$$

$$\frac{\partial r}{\partial y_r} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_r} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.4)b}$$

$$\frac{\partial r}{\partial \theta_r} = 0 \qquad \frac{\partial \alpha}{\partial \theta_r} = -1 \tag{4.4)c}$$

$$\frac{\partial r}{\partial x_1} = \frac{(x_r - x_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_1 - y_r)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.4)d}$$

$$\frac{\partial r}{\partial x_1} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.4)e}$$

The only other information needed for Kalman Gain calculation is the observer error $V^T RV$ with V being the derivative of the measurement model with respect to noise. If all landmarks are assumed to be uniformly affected by noise, it is reasonable to approximate V to be an identity matrix. R is the measurement noise. It is a diagonal matrix with one of the eigenvalues representing the error in distance measurement of the LIDAR and the other the angle measurement as in equation (4.5).

$$R = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\alpha \end{bmatrix} \tag{4.5}$$

Using all this information, as in it is possible to correct the estimate of the robot pose while simultaneously correcting the position of the landmarks. The result of such a Simultaneous Localization and Mappingis seen in section...

4.2 Feature Extraction with Linear Features

4.2.1 Overview

One of the most ubiquitous features of any indoor environment are walls. While it maybe necessary to find other obstacles and landmarks, for path planning and mapping, knowing the locations of walls always advantageous. At the very least it can be used to remove the laser readings close to it reducing the data set to be further analyzed by methods such as clustering. In relatively clean environments, such as corridors that are not too crowded, walls will usually suffice for Simultaneous Localization and Mapping. Also algorithms that find linear features are more robust to humans moving around in the environment than those finding cylinders or generic features in the data. Also since the feature we are trying to extract is a wall, it is reasonable to model it as a line of infinite length which reduces the dimensions of state required to represent it. Similar to section 4.1.4 once the walls have been detected, a measurement model is required to calculate the innovation. It is obvious that the measurement model and it's derivatives will not be the same as in point features as the wall moves very differently as the robot moves. For example, when the robot is moving parallel to the wall, the LIDAR scan from one time step to another will be identical, but unlike in the case of point features that need not imply the robot has not moved. So different measurement models are developed and their Jacobian matrices calculated for linear features.

4.2.2 RANSAC based Feature Extraction Algorithm

The fundamental concept in extracting linear features is to fit a line or multiple lines onto a set of points. There are a large number of methods to do that such as least squares, Split-merge and RANSAC [71]. Looking into a few implementations of RANSAC in common libraries such as PCL and OpenCV, we see that it tries to robustly fit a line to a given set of points. In regular environments, since more than one wall can be seen at a time, it becomes necessary to first segment the data before using the algorithm. Instead another approach is to randomly sample points and try to fit multiple lines simultaneously. This approach is explained in algorithm 1 [80].

Algorithm 1 Multiple line fitting with RANSAC

while

- there are still unassociated laser readings,
- and the number of readings is larger than the consensus,
- and we have done less than N trials.

do

- Select a random laser data reading.
- Randomly sample S data readings within D degrees of this laser data reading
- Using these S samples and the original reading calculate a least squares best fit line.
- Determine how many laser data readings lie within X distance from this best fit line
- If the number of laser data readings on the line is above some consensus C do the following:
  - calculate new least squares best fit line based on all the laser readings determined to lie on the old best fit line.
  - Add this best fit line to the lines we have extracted.
  - Remove the number of readings lying on the line from the total set of unassociated readings.

end while

4.2.3 RANSAC Examples

Considering the same setup as in figure 10, we can try the RANSAC algorithm on the data collected from the LIDAR. It is seen to give good results initially as seen in figure 13. As we move forward though the result is not always maintained. Looking at algorithm 1 it is possible that since the first laser reading is chosen at random, it may not be on a wall at all. S readings in it's neighborhood could generate a line that is at a small angle to both walls in the environment. Since it is close enough to the walls, a large number of points will wrongly be associated to this, which will result in a spurious wall being detected.This will have a detrimental effect in all the further time steps. An example of this is seen in figure 14.

(a) Polar plot of laser scan (b) Detected lines

Figure 13: Example of walls detected by RANSAC

(a) Detected lines (b) Detected lines

Figure 14: Example of spurious walls marked by RANSAC

4.2.4 Hough transform based Feature Extraction Algorithm

In the fields of image analysis and computer vision, hough transform is a very popular feature extraction technique [81]. While it was originally designed to find lines in an image can be used to detect any shape as long as that shape can be represented in a mathematical form [82]. It can detect the shape in spite of distortions making it much more robust than RANSAC [73]. In the current application the aim is to find walls

in the range data of the LIDAR. Hence the first step is to rasterize the points into an image so that we can use the OpenCV implementation of Hough Transform. The image is then converted to a binary image and then passed to the hough transform function. The algorithm itself is described below.

In image processing, a common way of representing lines is the normal form of a line equation,

$$[h!]r = x\cos\theta + y\sin\theta \tag{4.6}$$

In this form a line is completely defined by 2 variables r and θ. The first step of finding lines in an image is that for any point (x_0, y_0) in the image, a family of lines passing through that point is expressed as,

$$[h!]r_\theta = x_0\cos\theta + y_0\sin\theta \tag{4.7}$$

Since all lines are assumed to be infinite in both directions, θ can only take unique values between 0° and 180°. This range is then discretized to give a fixed number of possible line angles. The range r is also discretized based on the resolution required. For single pixel resolution the possible ranges will be all numbers from 0 to the length of the diagonal of the image. Since this will result in a large number of possible r values, a lower resolution will considerably speed up the algorithm. Now that there is a fixed number for possible r and θ, a 2D matrix of zeros is created in which each row corresponds to a possible value of θ and each column to r as in equation (4.8). This is called the accumulator.

$$A = \begin{array}{c} \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{array} \begin{array}{c} \begin{array}{cccc} r_1 & r_2 & \dots & r_n \end{array} \\ \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \end{array} \tag{4.8}$$

Once this initial setup is complete, a non zero point on the binary image (x_0, y_0) is chosen at random. For this point the family of lines passing through it will be given by equation (4.7). For each possible θ the corresponding r is calculated from the same equation. For each (r, θ) pair found, the corresponding entry in the accumulator A is incremented. Once all the possible pairs are calculated, a different image point is chosen till all the non zero points on the image are explored.

The accumulator now represents all possible lines in the image and the number of points that lie on each of the lines. The accumulator entries that are greater than a certain preset threshold represent the lines in the image. The (r, θ) values corresponding to these lines are returned.

An important aspect of using the Hough transform from OpenCV is the conversion of LIDAR data to image and the detected lines back to the robot frame of reference. The former is straight forward. Since the range of the LIDAR is known, the image size can be fixed based on the resolution required. For, example the LIDAR described in chapter **??** has a maximum range of $5m$. So for a resolution of $1cm$, a blank image of 500×500 pixels is created. When each LIDAR point is converted to Cartesian coordinates (x_l, y_l), they are with respect to an origin at the LIDAR itself. OpenCV images on the other hand, follow a coordinate system that has the origin in the top right corner of the image. Hence each point is mapped to the image coordinate frame using a homogeneous transformation,

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 500 \\ 1 & 0 & 500 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} \tag{4.9}$$

The image points are then rounded to the nearest centimeter and the pixel corresponding to that value is set to white. This gives an image with black background and all the Laser readings shown in white which can be passed to the hough transform function. It may sometimes be required to erode and dilate the image with different kernels to make the points more visible.

The lines returned by the hough transform, are in units of pixels and are with reference the image origin. To convert it to the robot frame, 2 points are chosen on the line and each of these points are mapped to the robot frame using the inverse transformation of equation (4.9) and the normal form of the equation of the line passing through these 2 points is found.

4.2.5 Hough Transform Examples

In the same arena as in figure 10, the hough transform is found to be really robust in finding the walls as seen in figure 15.

Unlike the RANSAC algorithm, the hough transform considers all possible lines before deciding on any line, it is robust to noise in the surroundings. This is especially useful when there are people walking around in the environment.

(a) Polar plot of laser scan (b) Detected lines

Figure 15: Example of walls detected by Hough Transform

(a) Overhead view of the arena (b) Detected lines

Figure 16: Example 1 of Hough Transform being robust to people

4.2.6 Measurement Model and the corresponding differentials

Once the lines in the data are recovered, we need an effective representation to store them. The common, slope-point form has a major disadvantage when trying to represent perfectly vertical lines and the two point form will expand the size of the measurement vector z increasing the calculation required for Extended Kalman Filter . So it is preferable to represent it in the normal form where, just the coordinates of the

(a) Overhead view of the arena

(b) Detected lines

Figure 17: Example 2 of Hough Transform being robust to people

point of intersection of the normal from the origin to the line is stored. For example, in figure 18 the line can be represented solely by the point D.

Once we have a line, we need to convert it to the inertial frame of reference as in section 4.1.4. Unlike point features we are not trying to find the same point in a different frame but the same line. In figure 19 we have point P_1 in robot frame and we need to find point P_2 in inertial given coordinates of the robot in inertial frame as P_r. For which we use Algorithm 2.

Once we have the landmark in the world frame we need to see if it corresponds to any existing landmark in the map. We check this by comparing both the perpendicular distances between the walls and the angle between the walls. This allows us to have 2 tuning parameters so that the association can be weighted as desired. Usually since most walls in an indoor environments are at right angles, the variation allowed in angle for the association is larger compared to distance.

Algorithm 2 is also called the inverse measurement model as in subsequent time steps for the measurement model we do the exact opposite process. That is given a line using it's normal form in inertial frame, we need to get the *measurement* from the robot. That is, we need the perpendicular distance from the robot to the wall and the angle of the laser beam which would hit the wall normally. For this given coordinate pf point P_2 as (x_2, y_2) and the robot pose as (x_r, y_r, θ_r) we get P_1 using equation (4.10).

Figure 18: Normal form of a line

Figure 19: Line in robot frame and world frame.

Algorithm 2 To convert linear features from robot frame to inertial frame

1. Convert point P_1 from robot frame of reference to inertial frame as in section 4.1.4
2. Given points P_r and P_1 in inertial frame, the equation of line l_1 is found using two point form of a line
3. Since $l1 \perp l2$ the slope of l_2 is the negative reciprocal of the slope of l_1.
4. Using the slope and the point P_1 the equation of line l_2 is found using slope-point form of a line.
5. The equation of line l3 is found similarly using slope of l_1 and point O.
6. Solving equations for l_3 and l_2 we get the coordinates of point P_2.

$$x_1 = x_2 - \frac{y_2(x_2 y_r - y_2 x_r)}{(x_2^2 + y_2^2)} \qquad y_1 = y_2 + \frac{x_2(x_2 y_r - y_2 x_r)}{(x_2^2 + y_2^2)} \tag{4.10}$$

$$r = \sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2} \qquad \alpha = \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) - \theta_r \tag{4.11}$$

Once we have P_1, we can find the perpendicular distance r and angle α using equation (4.11) giving measurement model h as per equation (4.12). This measurement can be used to calculate the *innovation* as per equation (2.18).

$$h = \begin{bmatrix} r \\ \alpha \end{bmatrix} \tag{4.12}$$

For calculating the *Kalman Gain*, we need the Jacobian matrix H. As each wall is independent of the other, this has the same structure as explained in section 4.1.4 and is given by equation (4.3). We derive each of the terms separately as per equation (4.13) and (4.14).

$$\beta = \frac{x_r x_1 - x_1^2 - y_1^2 + y_r y_1}{x_1^2 + y_1^2} \tag{4.13a}$$

$$\frac{\partial r}{\partial x_r} = 2x_1\beta \quad \frac{\partial r}{\partial x_r} = 2y_1\beta \quad \frac{\partial r}{\partial \theta_r} = 0 \tag{4.13b}$$

$$\frac{\partial \alpha}{\partial x_r} = 0 \quad \frac{\partial \alpha}{\partial y_r} = 0 \quad \frac{\partial \alpha}{\partial \theta_r} = -1 \tag{4.13c}$$

$$\gamma = \frac{x_r x_1 - x_1^2 - y_1^2 + y_r y_1}{x_1^2 + y_1^2} \tag{4.14)a}$$

$$\frac{\partial r}{\partial x_1} = -2x_1\gamma^2 - 2\gamma(2x_1 - x_r) \tag{4.14)b}$$

$$\frac{\partial r}{\partial y_1} = -2y_1\gamma^2 - 2\gamma(2y_1 - y_r) \tag{4.14)c}$$

$$\frac{\partial \alpha}{\partial x_1} = \frac{-y_1}{x_1^2 + y_1^2} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{x_1}{x_1^2 + y_1^2} \tag{4.14)d}$$

Having the Jacobian matrices we can use the same observer error given by equation (4.5) to correct the position estimate using equations (2.17) to (2.19).

CHAPTER V

Implementation and Results

5.1 Implementation of EKF SLAM algorithm

The SLAM algorithm, as discussed in chapter II, consists of subparts each of which present challenges of its own. The primary purpose of this thesis is to develop a flexible implementation that can accommodate different algorithms for the subparts,hence, the guiding principle of the implementation is modularity. Hence an object oriented language such as Python is the platform of choice. This allows us to create individual objects for each part of the algorithm which can be substituted with ease to utilize different combinations of algorithms for feature extraction, association and filtering.

During the initial setup, all the objects are created and initialized. A robot object is instantiated with its initial position and covariance representing the uncertainty in its position. This object holds the robot's position and uncertainty all through the runtime. It also contains objects for all the sensors and components that the actual robot contains. For example, an object of ENCODER class is instantiated with all its properties such as its noise factors, the diameter of the wheels it is attached on, and the separation between the wheels. A LIDAR object is also instantiated with the minimum and maximum range of the Hokuyo, the offset of the LIDAR from the center of the platform, and the measurement noise factors. These sensor objects are attached to the Robot object into proprioceptive and exteroceptive sensor lists, respectively. The proprioceptive sensors contain methods to calculate the motion model of the robot and also its Jacobian matrices. To the exteroceptive sensors, feature extractor objects are attached. Each feature extraction algorithm is implemented as its own class. All such classes have to have a particular structure based on the sensor whose data the extractor uses. For LIDAR based extractors, there has to be a function called *get_landmarks()* which takes in angles and distances and returns a list of landmark locations. They also need a member variable named *landmarkType* which contains the type of landmark that the extractor is trying to find.

Figure 20: SLAM Implementation using only LIDAR

The exteroceptive sensors are designed to take the list of landmark positions from each feature extractor attached to them and create objects of LANDMARK class for them. A combined list of landmark objects from all the feature extractors is returned by the exteroceptive sensor class. These landmark objects are objects of the LANDMARK class. This is a factory class which, for example, returns objects of either WALL or CYLINDER based on the *landmarkType* returned by *get_landmarks*. Each of these classes contain methods to calculate their measurement models and their Jacobian matrices. The main part of SLAM is carried out by an object of EKF class. This object contains state and covariance members which represent the whole environment. In SLAM along with the robot's position being updated, the environment also needs to be mapped. So a representation of the environment called the map is held by an object of EKF class. It is essentially a list of all robot and landmark objects created.

Once initial setup is made, the robot object is added to the EKF map. Then for each time step, the *run()* method of the EKF object is called. In this function the first step is prediction wherein for each robot in the map, the *predict()* method of the

robot object is called. That in turn calls the *update()* method of all the proprioceptive sensors attached to that robot. This gives the estimated position and uncertainty using equation (3.3).

Then the *observe()* method of the robot is called which in turn calls the exteroceptive sensors which generate landmark objects for all the landmarks seen at that time step. Each of these landmark objects are then checked to see if they are associated with a landmark preexisting in the map. For each re-observed landmark, the innovation and Kalman gain are calculated and the whole state is corrected. The map objects are then updated with the corrected positions.

5.2 Experimental Results

To test the impact of the two feature extraction algorithms discussed in Chapter IV on Simultaneous Localization and Mapping, the Integrated Mobile Platformwas driven around in first a controlled and then an uncontrolled indoor environment while continuously logging data. The Encoder and IMU data was logged at high speed at 100 Hz, and the LIDAR and Camera was logged at 5 Hz. The logged data was then analyzed using an EKF SLAM algorithm implemented as described in Section 5.1.

In the controlled environment, the two ends of a regular corridor are sealed off resulting in a closed space with no other features except walls and cylinders placed intentionally. This enables us to test the point feature extraction algorithm and see its effect on SLAM both individually and in combination with Hough transform based linear feature extraction. The controlled environment along with the path taken by IMP is seen in Figure 21. The closed-off space is still large enough such that it is not entirely within the range of the LIDAR. Hence, the problems and modifications discussed in Chapter IV are still relevant even in a clean environment. To look at the effectiveness of the SLAM algorithm, some knowledge of the actual path taken by the robot is required for comparison. This is reconstructed using an overhead camera and the movement of the robot in the video is recorded. While this may not give the actual path or the ground truth accurately, it gives a reasonable estimate of it to within a few inches of deviation. A sample reconstructed path is shown in Figure 21. The camera based reconstructed path is also shown in all the plots of SLAM reconstructions for comparison. In the run, the robot starts at the bottom right corner and ends at the position shown.

Since the map is constructed relative to the starting location, the robot is always assumed to be starting at the origin in all the reconstructions.

Figure 21: Controlled environment and path reconstructed by an overhead camera.

5.2.1 Using Point Features

In Figure 21 the cylinders around which the robot travels are the point features that are to be detected. The algorithm used is the same as described in Section 4.1.2. Using just the jump in distance as a landmark results in a large number of spurious landmarks. As discussed in Section 4.1.2, it is therefore necessary to filter through the candidate landmarks using preexisting knowledge as previously described. The minimum number of points that need to be on the candidate landmark is chosen to be 11 for this particular arrangement. This is a tuning parameter which is decided after a few trials for optimized performance. The other tuning parameters include the thresholding for the range that the angular width can be in based on the radius of the cylinders which is 0.1524 m and the threshold for the curvature of the candidate landmark which is chosen to be 9%. These tuning parameters are chosen only once for a particular environment.

Once the feature extractor is tuned, the SLAM algorithm itself needs to be tuned based on the estimated noise factors for the various sensors. The SLAM algorithm is more robust to these parameters than the feature extractor and hence, a range of values yield good performance. Another part of the algorithm that needs to be tuned carefully is the landmark association. For point features in the map, the Euclidean distance between the detected and the preexisting cylinder is used. Hence, the maximum allowable distance for association needs to be set. That distance has to be large enough to accommodate for a particular landmark not being seen for a few steps and then being detected. It also should not be too large since this will result in wrong associations which will correct the path of the robot in a completely different way than the correct associations. Hence this parameter is also a function of the order of distances in the arena and is chosen to be $2m$, in these experiments.

With these parameters, the path of the robot is reconstructed as seen in Figure 22. We can see that it tries to correct the path towards the ground truth compared to the odometry based estimate, but it overcompensates frequently. This demonstrates the disadvantage of the conservative nature of the filters used. Since the focus is to not allow any spurious landmarks, the actual landmarks are also missed in a number of time steps. This effects the reconstruction mainly when there is only one landmark in the field of view of the LIDAR.

In the part of the path that is shown in the upper right quadrant in Figure 22, as soon as the robot has lost the first cylinder from its field of view, only the middle

Figure 22: Controlled environment and path reconstructed by detecting point features in LIDAR data. Path starts from the origin.

Table 1: Errors in environment reconstruction using point features.

	Actual Position (x,y)	Detected Position (x,y)	Error(m)
Cylinder 1	(-1.2192,1.524)	(-1.2525,1.4851)	0.002622
Cylinder 2	(-0.6096,3.3528)	(-0.7522,3.3942)	0.022048
Cylinder 3	(-1.2192,5.1816)	(-1.5308,5.2055)	0.097665
End Position	(-0.4459,2.3715)	(-0.5917,2.4141)	0.023072

cylinder is seen for sometime until the robot turns and the third cylinder comes close enough to be resolved. This is the time interval when the SLAM reconstructed path deviates from the actual path as the heading was being previously corrected using the cylinders detected, but when the landmark is missed, the pure odometry keeps moving it in the same direction. A similar effect is seen in the upper right corner when the only cylinder that can be seen is the uppermost one. This compounds the error previously accumulated. It is only when the robot is able to see both the lower and the middle cylinders simultaneously, that the path is corrected. This overcompensation also results in a corresponding error in the reconstruction of the environment. This error as well as error in the end position of the robot is seen in Table 1.

5.2.2 Using linear features

In the controlled environment of Figure 21, the linear features are the 4 walls enclosing the space. For wall detection, two algorithms are discussed in Section 4.2. The RANSAC based algorithm, when implemented in the way discussed is seen to be not robust in Figure 14. When used in SLAM it generates a number of spurious landmarks which result in wrong data association and subsequent correction. This results in the reconstruction being highly erroneous.

Hence the Hough transform based algorithm is used. As described in Section 4.2.4, the only tuning parameters for that algorithm are the pixel and angular resolution and threshold for line detection. Since these are broad parameters any reasonable value may be chosen. Since we use an image of 1000 $\times$ 1000 pixels to represent 10 m $\times$ 10 m space in the real world, the physical significance of the pixel resolution is the minimum distance in centimeters that walls need to be far from each other for them to be considered as 2 individual walls instead of one. The angular resolution is useful as it avoids finding multiple walls with small angles between them. This is especially useful while

reconstructing indoor environments as most walls tend to intersect at right angles hence corresponding to a very low probability that the actual line features intersect at a small angle. In this arena, an angular resolution of 10° and a distance resolution of 2 cm is chosen. This allows a good reconstruction while speeding up the Hough transform itself as the size of the accumulator from equation (4.8) is reduced.

The noise covariances for the encoder and LIDAR are kept the same as in Section 4.1.3 since the same instruments are used. The only part that needs to be different from Section 5.2.1 is the data association. There are a large number of ways that a linear feature can be associated. The simplest method is the Euclidean distance. This is feasible as the lines are represented in the normal form as shown in Figure 18. Hence, the coordinates of the line are actually the point of intersection of the normal from the origin. If there is a difference in either angle or distance, the point of intersection will move. Hence the Euclidean distance between the detected and preexisting lines gives a good measure for association.

However, using solely the Euclidean distance is seen to have a drawback. It can lead to a wrong association when the lines are sufficiently close to the origin, such as in the bottom right corner of Figure 21. Then, the Euclidean distance between the points of intersection of the normals is lower than the threshold causing the 2 perpendicular walls to be associated as the same wall. Hence, it is necessary to add another heuristic to avoid this. The heuristic added in this implementation was based on the angle of intersection between the walls. The additional condition is that, even if Euclidean distance is small, if the angle of intersection is not close to 0 or 180 degrees, then the features are not associated with each other. With this addition, the path of the robot was reconstructed as shown in Figure 23.

It is seen that the SLAM reconstruction is able to track the actual position for the most part except during the turn at the top and in a part close to the left wall. In the former case, the deviation from the path is mainly due to the odometry. This occurs as the robot has turned sufficiently to lose sight of the right wall and there are only small sections of the back and left wall in its field of view. This causes the walls to be not detected and emphasis is therefore, given to the odometry. In the part close to the left wall, the robot is moving slightly towards the wall in the ground truth whereas according to the odometry it is almost parallel. Since the only wall that the LIDAR can see at this point is the left wall and since the left wall has only recently been observed there is a greater tendency to move the estimated wall than the robot. This is also the

Figure 23: Controlled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin.

Table 2: Errors in environment reconstruction using linear features.

	Actual Position (x,y)	Detected Position (x,y)	Error(m)
Wall 1	(0,-0.3048)	(-0.0066 -0.4371)	0.017547
Wall 2	(0.3048,0)	(0.3018,-0.0018)	0.000012
Wall 3	(0,6.096)	(-0.1784,6.3731)	0.108611
Wall 4	(-1.6764,0)	(-1.7358,-0.0447)	0.005526
End Position	(-0.4459,2.3715)	(-0.5362,2.3477)	0.008721

reason in Figure 23, for the left wall to be seen at an angle. Once the robot close enough to see the back wall, the SLAM reconstruction does a large correction resulting in the jump seen. The errors in the environment reconstruction as well as the path are shown in Table 2.

5.2.3 Using a combination of point and linear features

As seen in Section 5.1, the implementation of SLAM algorithm is designed to use both kinds of features simultaneously. With all the implementation details as in Sections 4.1.3 and 5.2.2, the reconstructed path is shown in Figure 24.

From the path, it is seen that a few of the errors from the previous two reconstructions have been corrected. For example, in Figure 22, there is a large error in the top left part of the plot due to the cylinder not being detected robustly. In this path, the lack of the cylinder feature is compensated for by using the walls. Since it is a corner region the algorithm is able to correct both position and orientation. In Figure 23, there is a jump in the path near the left wall, and the left wall is reconstructed at an angle, as discussed in Section 5.2.2. When the robot is traveling parallel to the left wall, it has 2 cylinders in its field of view, and hence, this jump is avoided.

There are still deviations from the path near the left wall as both the linear features based and the point features based reconstructions had both deviated at that region, a combination of the two cannot yield an accurate reconstruction, though it can be better than either one individually. The environment reconstruction errors are tabulated in Table 3

To compare the reconstructions, we can compare the landmark error values from the tables. In Table 3, the total of all the cylinder errors is 0.0231 m. The corresponding errors in Table 1 add up to 0.1223 m. Hence, in terms of finding the location of the

Figure 24: Controlled environment and path reconstructed by detecting both point and linear features in LIDAR data. Path starts from the origin

Table 3: Error in environment reconstruction using combination of linear and point features

	Actual Position (x,y)	Detected Position (x,y)	Error(m)
Cylinder 1	(-1.2192,1.524)	(-1.2856,1.5376)	0.004602
Cylinder 2	(-0.6096,3.3528)	(-0.5768,3.3911)	0.002543
Cylinder 3	(-1.2192,5.1816)	(-1.1308,5.2717)	0.015932
Wall 1	(0,-0.3048)	(-0.0177,-0.3476)	0.002145
Wall 2	(0.3048,0)	(0.2805,-0.0080)	0.000654
Wall 3	(0,6.096)	(-0.3663,6.1225)	0.134878
Wall 4	(-1.6764,0)	(-1.7436,0.0152)	0.004747
End Position	(-0.4459,2.3715)	(-0.5013,2.4103)	0.00457

Figure 25: Passages forming an uncontrolled environment

cylinders using the combination of both kinds of features reduces the error by 81.1%. Similarly The total error in walls, for the combination of both is 0.142 and with just linear features is 0.132. Hence, using the combination of features is seen to increase the error, but only by 8%. If we consider the end position of the robot, we see that compared to the point features based reconstruction, there is a 80% reduction of error in the reconstruction using both types of features and a 47.5% reduction compared to using only linear features.

5.2.4 Uncontrolled environment with linear features

The primary purpose of implementing EKF SLAM is to run it in real time. For this, it is essential that the tuning parameters previously discussed should not need to

Figure 26: Obstacles in an uncontrolled environment

be tuned for each specific arena configuration. Hence, a longer run in an uncontrolled environment with the same tuning parameters is analyzed.

From Section 5.2.3, it is clear that the linear features based SLAM is relatively accurate compared to point features based SLAM. Although the combination of both walls and cylinders for SLAM gave the best results, the feature extractor for cylinder is seen to be conservative and not entirely robust. In Section 4.2.4 it was shown that the Hough transform based linear feature extractor is robust to humans in its field of view. Hence for a long range uncontrolled indoor environment, the linear feature based SLAM is an ideal candidate.

To test this, the robot is driven around a long passage which has both open and closed doors on either side as seen in Figure 25 There are also people walking around and entering and leaving through those doors. As seen in Figure 26, there are also other unintentional point features in the environment.

This gives a challenge for both EKF based SLAM and for the Hough feature extractor. The correction should ideally be made based only on walls and not other lines such as door ways etc. For SLAM itself, since the distance traveled is much larger and the velocity with which the robot moves is also larger, the encoders drift by a considerable amount as seen in Figure 27. Since the run is in a larger environment, the ground truth is approximated manually, without any overhead camera data. It can be seen that linear feature based SLAM does a good job in estimating the path traveled compared to the odometry. It does not completely reconstruct the ground truth due to the large number of disturbances throughout it's path such as doors. Towards the end of the run, a number of estimated wall are seen due to the shape of the passages in that region.

Hence, it is seen that in an uncontrolled environment, the Hough transform based

Figure 27: Uncontrolled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin.

linear feature detector performs well at reconstructing the path of the robot, even though the reconstruction of the environment is not accurate. It is also seen that the tuning parameters chosen in Section 4.1.3 and 5.2.2, are robust and do not need to be changed for similar environments.

CHAPTER VI

Conclusion

This thesis has provided a summary of simple feature detection algorithms which can be used for Extended Kalman Filter based Simultaneous Localization and Mapping. A general overview of EKF and it's application to SLAM was given and a simple differential drive platform with capabilities of real time implementation of SLAm was described. The motion models for the platform and how they are used in EKF was discussed. The feature detectors discussed were simplistic and easy to implement and provide a first level of understanding of the way SLAM works. The different feature detection algorithms were plugged into the SLAM algorithm and the results were analyzed to gain input about the impact of robustness and accuracy of feature extraction on reconstruction of robot path and the environment. The various tuning parameters for each of the algorithms were discussed along with the reasoning on their impact on the results.

Further work can include more robust feature extraction for point features by utilizing the optimized functions of OpenCV. Robust feature extractors such as sift or surf can be experimented with. Also in this thesis the on-board camera available on the platform was not utilized. Further work on analyzing the images acquired by the on-board camera can result in replacing encoder based odometry by visual odometry. That making the algorithm scalable to other non-wheeled robots. Since the platform is equipped with a real time arm processor, implementing the current SLAM algorithm in real time will provide more insight about the different feature extractors discussed in terms of their speed and processing power requirements.

References

[1] M. Miettinen, M. Ohman, A. Visala, and P. Forsman, "Simultaneous localization and mapping for forest harvesters," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 517–522.

[2] D. Schleicher, L. Bergasa, M. Ocana, R. Barea, and M. Lopez, "Real-time hierarchical outdoor slam based on stereovision and gps fusion," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 440–452, Sept 2009.

[3] P. Newman, D. Cole, and K. Ho, "Outdoor slam using visual appearance and laser ranging," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1180–1187.

[4] M. P. Golombek, R. A. Cook, T. Economou, W. M. Folkner, A. F. C. Haldemann, P. H. Kallemeyn, J. M. Knudsen, R. M. Manning, H. J. Moore, T. J. Parker, R. Rieder, J. T. Schofield, P. H. Smith, and R. M. Vaughan, "Overview of the mars pathfinder mission and assessment of landing site predictions," *Science*, vol. 278, no. 5344, pp. 1743–1748, 1997. [Online]. Available: http://www.sciencemag.org/content/278/5344/1743.abstract

[5] S. Thrun, D. Hahnel, D. Ferguson, D. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, "A system for volumetric robotic mapping of abandoned mines," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, Sept 2003, pp. 4270–4275 vol.3.

[6] G. Arbeiter, J. Fischer, and A. Verl, "3-d-environment reconstruction for mobile robots using fast- slam and feature extraction," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, June 2010, pp. 1–5.

[7] J. Goyal and K. Nagla, "A new approach of path planning for mobile robots," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, Sept 2014, pp. 863–867.

[8] M. Sutoh, M. Otsuki, S. Wakabayashi, T. Hoshino, and T. Hashimoto, "The right path: Comprehensive path planning for lunar exploration rovers," *Robotics Automation Magazine, IEEE*, vol. 22, no. 1, pp. 22–33, March 2015.

[9] L. Guangshun and S. Hongbo, "Study of technology on path planning for mobile robots," in *Control and Decision Conference, 2008. CCDC 2008. Chinese*, July 2008, pp. 3295–3300.

[10] N. Ying, L. Eicher, W. Xunzhang, G. Seet, and M. Lau, "Real-time 3d path planning for sensor-based underwater robotics vehicles in unknown environment," in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 3, 2000, pp. 2051–2058 vol.3.

[11] R. Panda and B. Choudhury, "An effective path planning of mobile robot using genetic algorithm," in *Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on*, Feb 2015, pp. 287–291.

[12] D. Bodhale, N. Afzulpurkar, and N. Thanh, "Path planning for a mobile robot in a dynamic environment," in *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, Feb 2009, pp. 2115–2120.

[13] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard, "Visual slam for flying vehicles," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1088–1093, Oct 2008.

[14] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1322–1328 vol.2.

[15] P. Tripathi, H. Singh, K. Nagla, and S. Mahajan, "Occupancy grid mapping for mobile robot using sensor fusion," in *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on*, Feb 2014, pp. 47–51.

[16] G. Lakemeyer and B. Nebel, *Exploring Artificial Intelligence in the New Millennium*, ser. The Morgan Kaufmann Series in Artificial Intel-

ligence Series. Morgan Kaufmann Publishers, 2003. [Online]. Available: https://books.google.com/books?id=vnlQAAAAMAAJ

[17] L. D'Alfonso, A. Griffo, P. Muraca, and P. Pugliese, "A slam algorithm for indoor mobile robot localization using an extended kalman filter and a segment based environment mapping," in *Advanced Robotics (ICAR), 2013 16th International Conference on*, Nov 2013, pp. 1–6.

[18] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *Robotics, IEEE Transactions on*, vol. 23, no. 5, pp. 1036–1049, Oct 2007.

[19] ——, "Convergence analysis for extended kalman filter based slam," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 412–417.

[20] F. Chanier, P. Checchin, C. Blanc, and L. Trassoudaine, "Slam process using polynomial extended kalman filter: Experimental assessment," in *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, Dec 2008, pp. 365–370.

[21] J. Cheng, J. Kim, Z. Jiang, and X. Yang, "Compressed unscented kalman filter-based slam," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, Dec 2014, pp. 1602–1607.

[22] S. Kuifeng, D. Zhidong, and H. Zhen, "Novel slam algorithm for ugvs based on unscented kalman filtering," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, vol. 3, May 2012, pp. 63–67.

[23] N. Sunderhauf, S. Lange, and P. Protzel, "Using the unscented kalman filter in mono-slam with inverse depth parametrization for autonomous airship control," in *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, Sept 2007, pp. 1–6.

[24] R. Havangi, M. Nekoui, H. Taghirad, and M. Teshnehlab, "Slam based on intelligent unscented kalman filter," in *Control, Instrumentation and Automation (ICCIA), 2011 2nd International Conference on*, Dec 2011, pp. 877–882.

[25] Y. Zang, K. Yuan, W. Zou, and H. Hu, "A two-step particle filter for slam of corridor environment," in *Information Acquisition, 2006 IEEE International Conference on*, Aug 2006, pp. 370–375.

[26] B. Gouveia, D. Portugal, and L. Marques, "Speeding up rao-blackwellized particle filter slam with a multithreaded architecture," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 1583–1588.

[27] N. Fairfield, G. Kantor, and D. Wettergreen, "Towards particle filter slam with three dimensional evidence grids in a flooded subterranean environment," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 3575–3580.

[28] H. S. Lee and K. M. Lee, "Multiswarm particle filter for vision based slam," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct 2009, pp. 924–929.

[29] A. Koenig, J. Kessler, and H.-M. Gross, "A graph matching technique for an appearance-based, visual slam-approach using rao-blackwellized particle filters," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 1576–1581.

[30] H.-C. Lee, S.-H. Lee, S.-H. Lee, T.-S. Lee, D.-J. Kim, K.-S. Park, K.-W. Lee, and B.-H. Lee, "Comparison and analysis of scan matching techniques for cooperative-slam," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, Nov 2011, pp. 165–168.

[31] D. Cole and P. Newman, "Using laser range data for 3d slam in outdoor environments," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1556–1563.

[32] J. Weingarten and R. Siegwart, "Ekf-based 3d slam for structured environment reconstruction," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 3834–3839.

[33] O. Stasse, A. Davison, R. Sellaouti, and K. Yokoi, "Real-time 3d slam for humanoid robot considering pattern generator information," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 348–355.

[34] R. Li, J. Liu, L. Zhang, and Y. Hang, "Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments," in *Inertial Sensors and Systems Symposium (ISS), 2014 DGON*, Sept 2014, pp. 1–15.

[35] D. Wang, H. Liang, T. Mei, H. Zhu, J. Fu, and X. Tao, "Lidar scan matching ekf-slam using the differential model of vehicle motion," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, June 2013, pp. 908–912.

[36] H. Alismail, L. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 6096–6101.

[37] M.-J. Jung, H. Myung, S.-G. Hong, D. Park, H.-K. Lee, and S. Bang, "Structured light 2d range finder for simultaneous localization and map-building (slam) in home environments," in *Micro-Nanomechatronics and Human Science, 2004 and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society, 2004. Proceedings of the 2004 International Symposium on*, Oct 2004, pp. 371–376.

[38] K. Narayana, F. Goulette, and B. Steux, "Planar landmark detection using a specific arrangement of lidar scanners," in *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, May 2010, pp. 1057–1069.

[39] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[40] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Oct 2003, pp. 1403–1410 vol.2.

[41] T. Vidal-Calleja, A. Davison, J. Andrade-Cetto, and D. Murray, "Active control for single camera slam," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1930–1936.

[42] H. Lim and Y. S. Lee, "Real-time single camera slam using fiducial markers," in *ICCAS-SICE, 2009*, Aug 2009, pp. 177–182.

[43] M. Milford and G. Wyeth, "Single camera vision-only slam on a suburban road network," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 3684–3689.

[44] Z. Yang, "Large scale visual slam with single fisheye camera," in *Audio, Language and Image Processing (ICALIP), 2014 International Conference on*, July 2014, pp. 138–142.

[45] M. Milford, F. Schill, P. Corke, R. Mahony, and G. Wyeth, "Aerial slam with a single camera using visual expectation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2506–2512.

[46] K. Al-Mutib, E. Mattar, M. Alsulaiman, and H. Ramdane, "Stereo vision slam based indoor autonomous mobile robot navigation," in *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, Dec 2014, pp. 1584–1589.

[47] C. Han, Z. Xiang, J. Liu, and E. Wu, "Stereo vision based slam in outdoor environments," in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, Dec 2007, pp. 1653–1658.

[48] A. Kalay and I. Ulusoy, "Vision-based simultaneous localization and map building: Stereo and mono slam," in *Signal Processing and Communications Applications Conference, 2009. SIU 2009. IEEE 17th*, April 2009, pp. 125–128.

[49] S. Takezawa, D. Herath, and G. Dissanayake, "Slam in indoor environments with stereo vision," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2, Sept 2004, pp. 1866–1871 vol.2.

[50] J.-H. Kim and M. J. Chung, "Slam with omni-directional stereo vision sensor," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, Oct 2003, pp. 442–447 vol.1.

[51] J. Hartmann, D. Forouher, M. Litza, J. H. Kluessendorff, and E. Maehle, "Real-time visual slam using fastslam and the microsoft kinect camera," in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, May 2012, pp. 1–6.

[52] M. Ghani, K. Sahari, and L. C. Kiong, "Improvement of the 2d slam system using kinect sensor for indoor mapping," in *Soft Computing and Intelligent Systems*

(SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on, Dec 2014, pp. 776–781.

[53] M. Gansari and C. Buiu, "Building a slam capable heterogeneous multi-robot system with a kinect sensor," in *Electronics, Computers and Artificial Intelligence (ECAI), 2014 6th International Conference on*, Oct 2014, pp. 85–89.

[54] J. Marck, A. Mohamoud, E. v.d.Houwen, and R. van Heijster, "Indoor radar slam a radar application for vision and gps denied environments," in *Microwave Conference (EuMC), 2013 European*, Oct 2013, pp. 1783–1786.

[55] E. Jose and M. Adams, "Multiple line-of-sight predicted observations with millimetre wave radar for outdoor slam," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 1, Dec 2004, pp. 155–160 Vol. 1.

[56] D. Ribas, P. Ridao, J. Neira, and J. Tardos, "Slam using an imaging sonar for partially structured underwater environments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 5040–5045.

[57] M. Fallon, J. Folkesson, H. McClelland, and J. Leonard, "Relocating underwater features autonomously using sonar-based slam," *Oceanic Engineering, IEEE Journal of*, vol. 38, no. 3, pp. 500–513, July 2013.

[58] S. Jo, H. Choi, and E. Kim, "Ceiling vision based slam approach using sensor fusion of sonar sensor and monocular camera," in *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, Oct 2012, pp. 1461–1464.

[59] G. Leivas, S. Botelho, P. Drews, M. Figueiredo, and C. Haffele, "Sensor fusion based on multi-self-organizing maps for slam," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, Sept 2010, pp. 139–143.

[60] F. Zhang, C. Shen, and X. Ren, "The application of multi-sensor information fusion by improved trust degree on slam," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on*, vol. 1, Aug 2013, pp. 360–364.

[61] C. Vianchada, P. Escamilla, M. Ibarra, J. Ramirez, and P. Gomez, "Attitude estimation using fusion of monocular slam and inertial sensors," *Latin America Trans-*

actions, IEEE (Revista IEEE America Latina), vol. 12, no. 6, pp. 977–984, Sept 2014.

[62] E. B. Olson, S. Teller, and J. Leonard, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2008.

[63] A. Diosi, L. Kleeman, A. Diosi, and L. Kleeman, "Uncertainty of line segments extracted from static sick pls laser scans," in *SICK PLS laser. In Australiasian Conference on Robotics and Automation*, 2003.

[64] G. A. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *J. Intell. Robotics Syst.*, vol. 40, no. 3, pp. 267–297, Jul. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:JINT.0000038945.55712.65

[65] P. Nunez, R. Vazquez-Martin, J. del Toro, A. Bandera, and F. Sandoval, "Feature extraction from laser scan data based on curvature estimation for mobile robotics," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, May 2006, pp. 1167–1172.

[66] J. E. Guivant, F. R. Masson, and E. M. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, vol. 40, no. 23, pp. 79 – 90, 2002, intelligent Autonomous Systems - {IAS} -6. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889002002336

[67] M. Montemerlo, "Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[68] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based slam," *Int. J. Rob. Res.*, vol. 26, no. 4, pp. 335–359, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1177/0278364906075026

[69] Y. Liu and S. Thrun, "Results for outdoor-slam using sparse extended information filters," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 1, Sept 2003, pp. 1227–1233 vol.1.

[70] A. Garulli, A. Giannitrapani, A. Rossi, and A. Vicino, "Mobile robot slam for line-based environment representation," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, Dec 2005, pp. 2041–2046.

[71] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, Aug 2005, pp. 1929–1934.

[72] 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

[73] Z. Hu, Y. Yang, and H. Tsui, "In defense of the hough transform," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, Aug 1998, pp. 24–26 vol.1.

[74] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[75] G. Welch and G. Bishop, "An introduction to the kalman filter," Chapel Hill, NC, USA, Tech. Rep., 1995.

[76] *Stochastic Models: Estimation and Control: v. 1: Estimation and Control:*, ser. Mathematics in Science and Engineering. Elsevier Science, 1979, no. v. 1. [Online]. Available: http://books.google.com/books?id=FkkZBqQG36gC

[77] O. Jacobs, *Introduction to control theory*, ser. Oxford sciences publications. Oxford University Press, Incorporated, 1993. [Online]. Available: http://books.google.com/books?id=df8pAQAAMAAJ

[78] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises, 4th Edition.* Wiley Global Education, 2012. [Online]. Available: https://books.google.com/books?id=PeQbAAAAQBAJ

[79] P. C. Mahalanobis, "On the generalised distance in statistics," in *Proceedings National Institute of Science, India*, vol. 2, no. 1, Apr. 1936, pp. 49–55. [Online]. Available: http://ir.isical.ac.in/dspace/handle/1/1268

[80] S. Riisgaard and M. R. Blas, "Slam for dummies," *A Tutorial Approach to Simultaneous Localization and Mapping*, vol. 22, no. 1-127, p. 126, 2003.

[81] G. Stockman and L. G. Shapiro, *Computer Vision*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[82] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972. [Online]. Available: http://doi.acm.org/10.1145/361237.361242