SIMULTANEOUS LOCALIZATION AND MAPPING ON AN INTEGRATED MOBILE PLATFORM

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 design of 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

SIMULTANEOUS LOCALIZATION AND MAPPING ON AN INTEGRATED MOBILE PLATFORM

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 presents a study on implementing Simultaneous Localization and Mapping (SLAM) on an Integrated Mobile Platform (IMP). The IMP has been designed to have autonomous navigation capability.The modular design enables testing of a variety of navigation algorithms,control designs and SLAM. The design and construction of IMP is detailed in Chapter III of this thesis. A common challenge in most implementations of autonomous navigation is localization and mapping. Towards this end, an Extended Kalman Filter (EKF) based SLAM algorithm is introduced and implemented. The SLAM implementation is modular and designed towards easy implementation of different algorithms. The thesis includes discussion on the mathematical background as well as insights into the practical implementation of SLAM on a ground vehicle such as IMP.

Two feature detectors are implemented in the SLAM algorithm. The feature detectors extract features predicted to be in the environment using data acquired from a 2D scanning laser range finder (LIDAR). A simplistic feature extraction method to

detect point features in the environment is discussed along with its results and drawbacks. Next, linear feature extraction algorithms are discussed, both conceptually and practically. The algorithms discussed are utilized in the modular implementation of SLAM.The thesis discusses insights about the effectiveness and robustness of the feature extraction techniques and their efficacy in SLAM implementation are presented and discussed in this thesis.Experimental results for feature extraction and SLAM, in both controlled and uncontrolled indoor environments are analyzed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

Autonomous Robots are used in more and more applications everyday. Mobile robots with autonomous navigation capability find applications in search and rescue, space exploration, and even domestic use [1–3]. For this thesis, an Integrated Mobile Platform (IMP) is designed as a ground vehicle capable of autonomous navigation. In this chapter, the basic concept of Simultaneous Localization and Mapping (SLAM) and its importance in autonomous navigation is discussed as well as the various methods of sensing the environment.

1.1 Simultaneous Localization and Mapping

In most implementations of autonomous navigation, a common challenge faced is the need for the robot to have a map of the operating environment it is in and to precisely know its own location within this map. If an accurate map of the environment is available a priori, estimating the path of the robot would be a simple localization task [4]. Similarly if the true path of the robot was known, building a map of the environment is straightforward [5, 6]. However, when both the environmental map and the true path of the robot are unknown, the problem becomes challenging.

A common situation where such a problem arises is when a robot is required to autonomously navigate a previously unexplored environment, or navigate in a changing environment. Examples of applications involving such unexplored environments include space, underwater and underground explorations [3, 7, 8]. Robots capable of autonomously navigating changing environments find application in both domestic and industrial scenarios where there are humans involved in the environment. The movement of humans themselves as well as the changes they make to the environment create challenges, in which a robot's ability to map the surroundings and localize itself in that map becomes invaluable.

Simultaneous Localization and Mapping addresses the problem of a robot estimating its own position in an unknown environment, by simultaneously building a map of the environment while it achieving localization. To achieve SLAM, the robot utilizes its proprioceptive sensors to get an estimate of its current state (e.g., position, velocity), and then observe the environment via its exteroceptive sensors. It is understood that all sensor measurements are corrupted by noise. The measurements of the environment is then used to improve upon the estimated state while simultaneously improving the robot's map of the environment.

The two processes of localization and mapping need to be done simultaneously primarily due to the relationship between the localization error and mapping error being cyclic in nature. An increase in error in either one, will result in a larger increase in the other.

Most SLAM implementations use odometry to calculate the initial estimate of the robot pose. This pose estimate is refined by using measurements of the environment made by other sensors. 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 use of SLAM on smaller weight-constrained robots such as Unmanned Aerial Vehicles (UAV) to a larger extent [9] and also increases the potential of having SLAM based consumer products, e.g. autonomous vacuum cleaners, on the market. Another trend in robotics is the change in the maps that represent the environment. Most early papers on SLAM used a 2 dimensional representation of the environment. While the extension of these algorithms to full three-dimensional representation has been mathematically straightforward, it is only recently, more algorithms capable of the additional computational complexity were introduced [10–12]. The maps generated by SLAM can either be used on their own [7, 13], or as an input to path planning algorithms [14–19].

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) [20–23], an Unscented Kalman Filter (UKF) [24–27], or by Rao-Blackwellized Particle Filters (RBPF) [28–32]. 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 [33]. ICP is an algorithm that minimizes the difference between two clouds of points. In the algorithm, one point cloud, the reference or target, is kept fixed while the other one, the source, is transformed to best match the reference. The algorithm iteratively revises the transformation (combination of translation and rotation) needed to minimize the distance from the source to the reference point cloud. The inverse of this transformation gives an estimate of the relative pose of the robot.

The main advantage of ICP over the other (probabilistic) SLAM algorithms is the relatively high computational speed and the ease of implementation. However, it does 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 its simplicity and lower computational complexity.

1.2 Environment Sensing

As seen in the last section, the two major components of SLAM are proprioceptive sensing for estimation of the pose of the robot and exteroceptive sensing for sensing the environment around the robot and to create a representation of the environment that the robot can understand. While the former is a function of the motion model of the robot and the odometry of the robot, the latter depends mainly on the sensor used. A large number of exteroceptive sensors can be used to gather data about the environment based on the type of robot and the type of environment the robot is in. 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 fuse their data to obtain a better representation of the environment. In [58], data from a SONAR is combined with a single camera image in SLAM. Development of such sensor fusion algorithms is an active area of research by itself [59–61].

The two-dimensional scanning laser range finder (LIDAR),which measures the distance to objects in a plane, is a commonly used sensor for mapping the environment. There are two basic approaches to mapping with LIDARs, feature extraction and scan matching. The feature extraction approach looks for specific features that are predicted to be in the environment. These features are termed landmarks, and are used for localization of the robot. The predicted features that this approach looks for is a function of the environment. In indoor settings, lines, corners, and curves are known to be good features to expect [62–65], whereas in outdoor environments, tree detectors [66] have been extensively used [67–69]. One of the disadvantages of feature based landmark detection is the lack of a general-purpose feature detector that works well in varied environments.

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 is not dependent on the world containing straight lines, corners, or trees. However, scan matching has a major disadvantage, it tends to create dense pose graphs that significantly increase the computational cost of computing a posterior map.

The computational cost of feature based algorithms are lower than the cost of scan matching algorithms as searching over data associations is computationally less expensive than searching over the space of rigid body transformations. In scan matching algorithms, the computational cost at each iteration, is dependent on the error existing at the previous iteration. Hence, a large initialization error will result in large computational cost. Whereas, the cost of feature based matching is nearly independent of initialization error. Therefore, feature based algorithms are preferred over scan matching for applications within a homogeneous environment.

1.3 Thesis Outline

In the previous sections, it is described that Simultaneous Localization and Mappingis an essential component of autonomous navigation in unknown or dynamic environments. In Chapter II, the general concepts behind probabilistic SLAM is discussed. As an example of probabilistic SLAM techniques, a Gaussian filter, the Extended Kalman Filter, is developed along with its mathematical formulation. The Extended Kalman Filteris then presented as a solution to the SLAM problem.

Chapter III details the design process of the Integrated Mobile Platform along with the motivation and implementation of the various hardware and control components.

The software design for the current configuration of the platform is discussed followed by the motion model of the robot and how it can be used for SLAM.

Using the data from the LIDAR mounted on the platform described in Chapter III, two different kinds of features are extracted. A simplistic algorithm to extract point features is discussed in Chapter IV along with its drawbacks,thereafter modifications to overcome the drawbacks are suggested. Linear features such as walls are extracted first using RANSAC [70] and later using Hough transform [71]. The extracted features are utilized in the SLAM algorithm detailed in Chapter II and the results are presented in Chapter V.

CHAPTER II

Overview of 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 its environment taking into consideration that all measurements are corrupted by noise. The readings can be modeled as probability distribution functions. Both the pose of the robot and the location of the environmental features can also be considered to have probability distributions. Initially the variance of the robot pose is small because the initial pose is generally assured to be known to some level of accuracy. However as the robot moves around the environment, the pose estimate deteriorates due to accumulation of errors. The position of objects in the environment 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 their probability distribution. This approach to SLAM is called probabilistic SLAM.

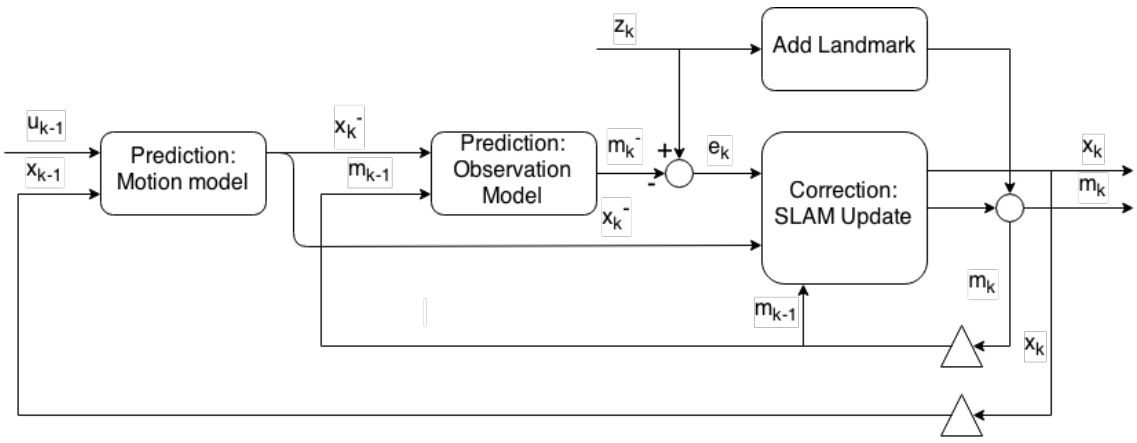


Figure 1: Block diagram of probabilistic SLAM process.

Most probabilistic SLAM methods follow the same structure. At every discrete time step k, the inputs to the SLAM algorithm are: i) the control input to the robot u_{k-1} and ii) the environmental features detected 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. 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. We assume that the position of the robot using GPS or someother global/local sensor is not available. These sensors which estimate the movement of the robot are termed *proprioceptive* sensors. The movement information, even if attained from the sensors, is referred to as the *control* inputs.

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. The error accumulates, resulting in increase of the uncertainty in the robot's position. 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: i) environmental observation and ii) 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. 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 typically used further in the algorithm. These problems can be solved by minimizing the number of bad landmarks extracted. Redefining a more suitable landmark extraction policy usually yields better performance. 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. Data association, like landmark extraction, is dependent on the type of map used.

Another important observation is that the two processes, of landmark prediction and data association, 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; therefore, reducing processing time. This approach must be implemented after due consideration depending on the kind of features being associated. For example if long linear features are being associated, 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. This step will vary based on the types of landmarks used and the type of map provided.

The conclusion of the filter update 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 family of Gaussian filters. 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.

As discussed in the previous section, 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 following form:

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

where μ is the mean 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.

Before describing how the Extended Kalman Filtercan be applied towards a SLAM algorithm, some preliminaries are provided on Kalman and Extended Kalman Filters.

2.2.1 Kalman Filter

The Kalman Filter [72, 73] consists of mathematical equations that give an efficient computational recursive solution of the least squares problem for a dynamic process corrupted by Gaussian white noise. The filter provides several advantages such as: it supports estimation of the past, present and future states even when the precise dynamic model of the system is not known.

In general the Kalman Filter equations address the problem of trying to estimate the state $x \in \Re^n$ of a discrete time process described by the following linear stochastic equation:

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

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

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

where the random variables w_k and v_k represent the process and measurement noise respectively, and are assumed to be independent of each other with a normal probability distribution functions given by

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

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

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

The equations of Kalman filter can be classified in to be in 2 groups: i) *Prediction* equations and ii) *Correction* equations. The former are responsible for projecting the current state and error probabilities forward in time and the latter are responsible for adjusting the projected estimate by an actual measurement at that time.

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 Equations (2.6) and (2.7) where A and B are from Equation (2.2) and Q is defined in Equation (2.4) [73]:

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

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

The first step in the correction step is to find a *gain* or *blending factor* that minimizes the error covariance. Which is given by Equation (2.8) [72, 74–76]. The next step is to actually measure the process to get z_k and generate a better estimate using Equations (2.9) and (2.10) [73]:

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

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

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

After each prediction and correction step, the corrected state found is used as an initial estimate for the prediction step in 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 was described as a state, $x \in \Re^n$,estimator of a discrete time process described by a set of *linear* dynamical equations. But, most applications including SLAM consists of systems governed by *non-linear* dynamics equations. Therefore resulting in the need for an EKF, The EKF utilizes the non-linear dynamics of the process, while using a linearization of the dynamics around the current state for computing the covariance matrices. 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.

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) with the measurement $z \in \Re^m$ is given by Equation (2.12):

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

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

where, 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 two sets of equations. The prediction step is given by Equations (2.13) and (2.14):

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

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

Equation (2.15) mimics the dynamics of the process except that w_k is set to zero since the noise is not known a priori. Equation (2.14) is similar to the Kalman filter equations except A_k and W_k are Jacobian matrices of partial derivatives with respect to x and w respectively, which are calculated at each time step 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) update the estimate of the state and covariance based on a measurement z at time k:

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

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

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

where the covariance matrices, H and V are again Jacobian matrices of the partial of h with respect to x and v respectively. They are also recalculated at 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 Kalman filter and Extended Kalman Filter, 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):

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

Since the pose itself consists of both location (x_R, y_R) and orientation of the robot θ_R, the state vector will have a length of $(3 + 2n)$ if there are n landmarks observed at a particular time step.

The first step towards estimating the state μ, is the prediction step as discussed in Section 2.1. The motion model is represented by a non linear function g and the prediction step is based on Equations (2.13) and (2.14). In Equation (2.13), the control input u, can either be the commands given to the robot or from proprioceptive sensors. For the purpose of this thesis, control input u is assumed to be from the proprioceptive sensors. The process noise w, is modeled as sensor noise from the proprioceptive sensor. Hence, to calculate the Jacobian matrix of the motion model with respect to the noise

w, the motion model is differentiated with respect to the control input u. The state and the covariance is given by:

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

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t \Sigma_{Control} V_t^T \tag{2.24}$$

$$G_t = \frac{\partial g}{\partial \mu} \tag{2.25}$$

$$V_t = \frac{\partial g}{\partial u}. \tag{2.26}$$

Based on data from exteroceptive sensors, features in the environment are extracted and used to update the position of the robot. A wide variety of sensors may be used such as a laser range finders and cameras depending on both the environment and the robot. The location of each of these features act as measurement z from Equation (2.12). The measurement model h relates the pose of the robot to the measurement. The same model h, is also used for predicting the measurement for landmarks already in a map. Once the landmarks 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 find 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 find the Jacobian H, the measurement model h is differentiated with respect to the full state μ, which is composed of both the robot pose and landmark positions. 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 R depends on the type of sensor being used. The SLAM update equations with these considerations can be simplified as:

$$K_t = \bar{\Sigma}_t H_t^T (H_t . \bar{\Sigma}_t . H_t^T + V R V^T)^- 1 \tag{2.27}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \tag{2.28}$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t. \tag{2.29}$$

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, modularity is emphasized in its design.

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

Mobility 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 its own position and orientation.

Environment sensing Has to have sensors to perceive the environment.

On-board computational power Has to have sufficient on-board processing power to do the computations necessary for SLAM and other navigation and obstacle detection algorithms.

Figure 2: The Integrated Mobile Platform.

Real-time controller Has to be capable of real time control implementation.

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.

Modularity The various components, both hardware and software, need to be designed in such a way that they are swappable with other components.

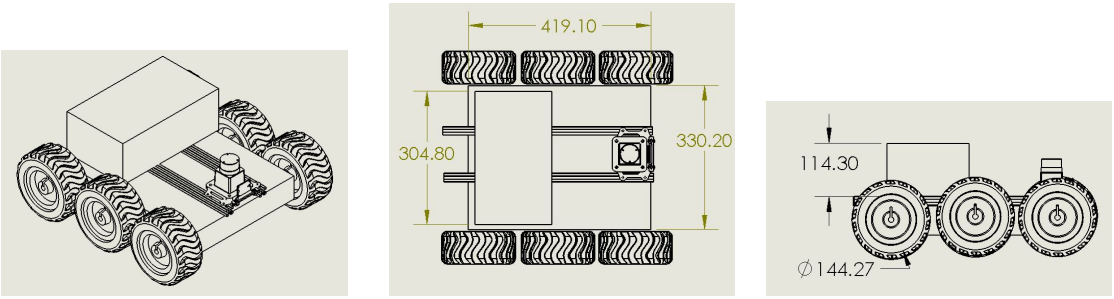
Payload Capacity The platform has to be capable of carrying sufficient payload for both sensory components and for any applications it is deployed for.

While these requirements define the bare minimum that is needed for such a platform, the design goals that guided the development of IMP were broader. 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, the IMP was designed as a six wheeled differential drive platform with a multi-processor control system, including two micro-controllers and a single-board computer. The platform was also equipped with a number of sensors: 1) a scanning laser range finder, and 2) a monocular camera. Additionally two of the platform wheels are equipped with encoders and the embedded processor module is equipped with an Inertial Measurement Unit (IMU).

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



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

Figure 3: Dimensions of Integrated Mobile Platform.

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, which is a mobile robot whose movement is based on at least two separately driven wheels placed on either side of the robot body. Therefore the IMP can change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering mechanism. If both the wheels are driven in the same direction and speed, the robot will go forward 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 platform axes. Differential drive system has several advantages over other systems. Ackerman steering, which is used in automobiles, has a lower bound on the turning radius and also requires additional control inputs for steering. In other systems such as tank treads, it is difficult to reconstruct the odometry; as the target of the robot is autonomy, this proves to be a disadvantage. A differential drive robot gives on the spot turning capability with minimal control complexity.

Once the type of robot was decided, the next important consideration was its 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. Thus, 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 lightweight but structurally sound so as to allow larger payload capability. For this purpose the material of choice is aluminum which has the structural strength required. The chassis structure itself is a hollow cuboid, but there are multiple support struts running across its breadth to give added structural

strength. This design allows the weight of the chassis to be reduced 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 an offset-V tread to have sufficient friction to traverse both indoor and outdoor environments. The wheel size was chosen based on the size of platform. The number of wheels were picked based on stability, control and sensing requirement. The four outer wheels are driven by DC geared motors generating sufficient torque the robot to have a small turning radius. Putting the encoders on the driven 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 passive wheels were added, which are used for odometry. The passive middle wheels are rotated solely due to the robot motion. Hence, measuring the rotation of the passive wheels yields a more accurate position estimate 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. Motors with higher torque capacity draw more current; this in turn demands larger and heavier batteries to achieve consistent runtime. 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. A typical Ni-Mh battery has a specific energy of 20-120 Wh/kg and the result of the heavy payload it was found that the motors were not strong enough to turn 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 its own weight but also additional sensor and computational payloads. 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 low weight allows for installation of additional 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 and on-spot turning..

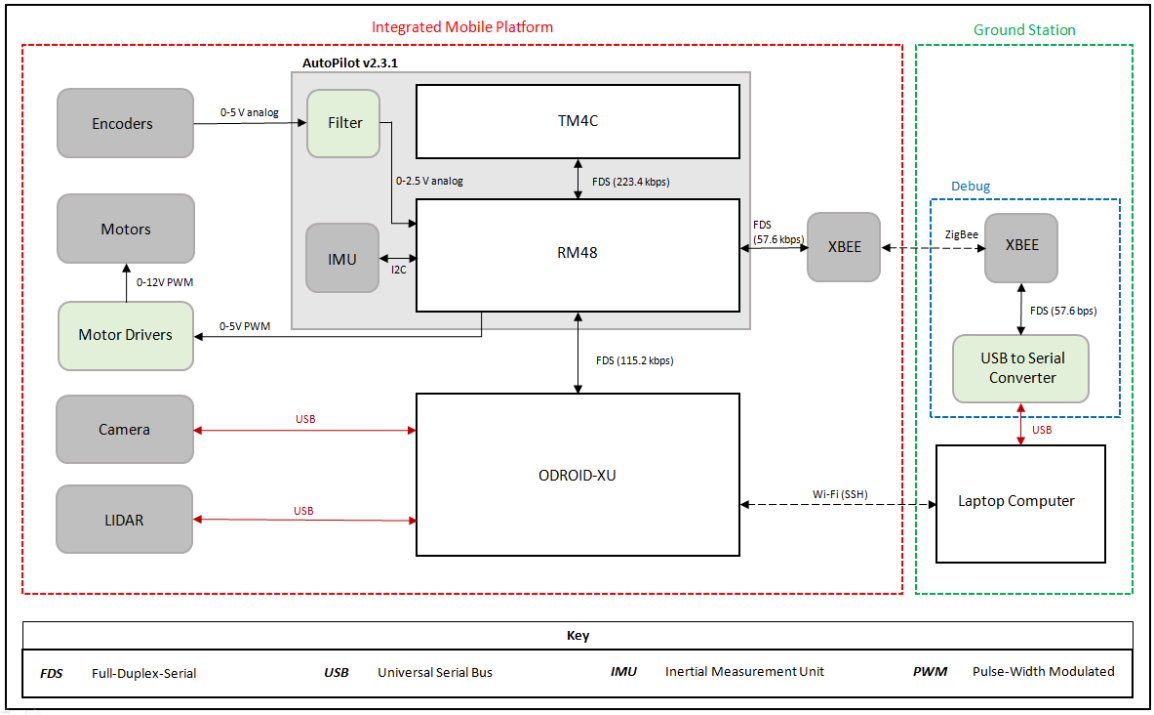


Figure 4: System diagram of Integrated Mobile Platform.

3.3 Electrical Subsystem

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 9-axis inertial measurement unit.
- ODROID-XU single-board computer.
- MA 3 miniature absolute magnetic shaft encoders.
- *Leopard Imaging* 5 MP camera.
- Hokuyo-URG-04lx scanning LIDAR .
- TP-Link Nano wireless module
- XBEE wireless radio

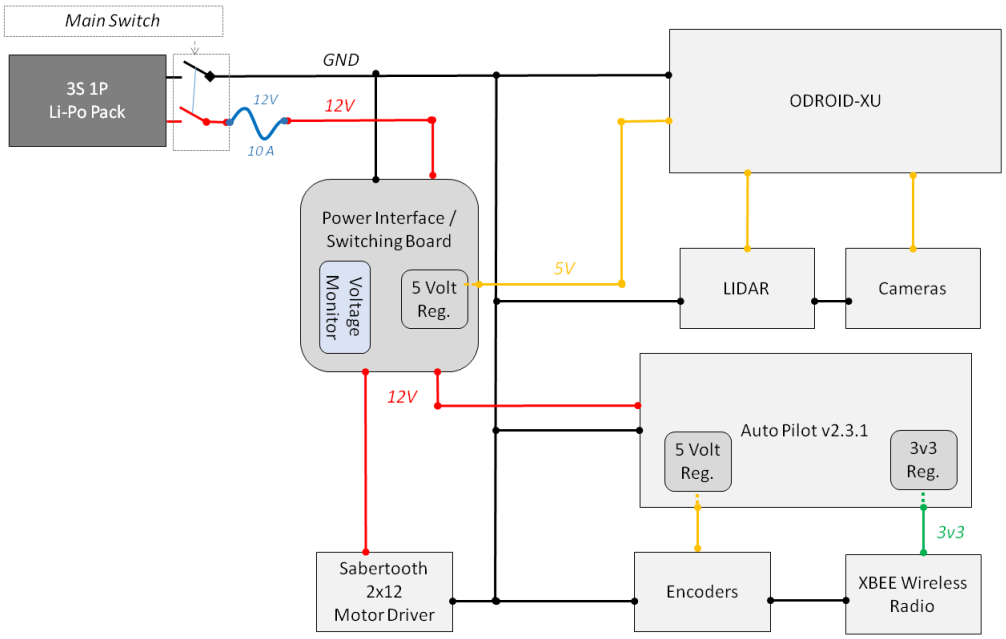


Figure 5: Integrated Mobile PlatformPower routing diagram.

3.3.1 Device Purpose and Descriptions

AutoPilot

The AutoPilot is a dual processor unit that is equipped for low level real-time control algorithms as well to act as a data acquisition platform and an input/output interface for the robot. The AutoPilot consists of two processing units: a Texas Instruments TM4C and a RM48 ARM 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 9-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 can be easily interfaced to a GPS unit.

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 real-time control algorithms such as PID can be implemented.

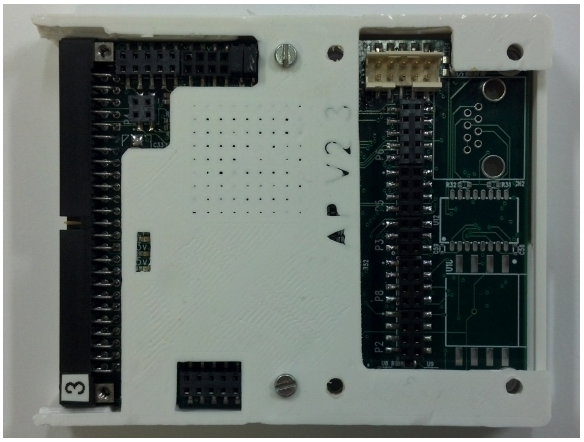


Figure 6: Autopilot on the Integrated Mobile Platform.

The autopilot is currently implemented as an I/o processing and sensor data acquisition unit and is interfaced with the encoders, the IMU and the motor driver.

ODROID XU

The ODROID is the on-board computer and provides significant computational power especially when the IMP is used for real time autonomous navigation. Equipped with Samsung Exynos5422 Cortex-A15 2.0Ghz quad core and Cortex-A7 quad core CPUs the odroid has 2Gbyte LPDDR3 RAM capable of operating at 933MHz and several USB 2 and USB 3 ports for interfacing. With these features, considerably powerful and computationally intensive algorithms for Simultaneous Localization and Mapping along with path planning and decision making can be run. The on-board computer is implemented as a higher level controller of the robot and is powered by a separate 5V regulator as shown in Figure 5.

Encoders

The magnetic shaft encoders measure the absolute position of the wheel, and provide a 0-5V output and can hence, be modeled as a continuous rotation potentiometer. So while the wheel is continuously turning, the value rises from 0 to 5 volts and then instantaneously drops to 0 at the end of every rotation. Therefore, to calculate rotational speed, the encoder signal has to be first stored in an accumulator and then differentiated. While the signal is reasonably free of noise and performs well for dead reckoning, there exists some component of high frequency noise that needs to be filtered out. So the analog signal is first passed through a hardware based active low pass filter of 950

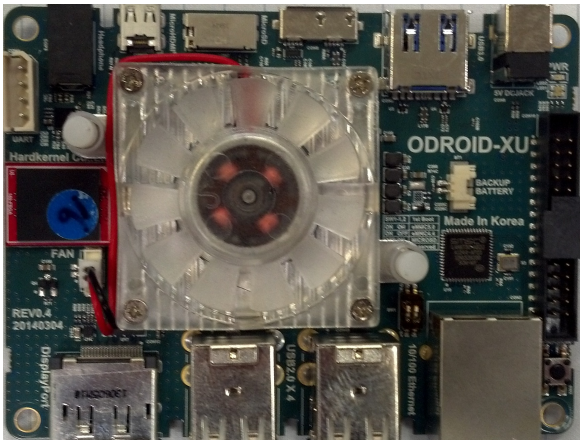


Figure 7: Odroid on the Integrated Mobile Platform.

Hz which is shown in Figure 8. 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 are built into the AutoPilot.

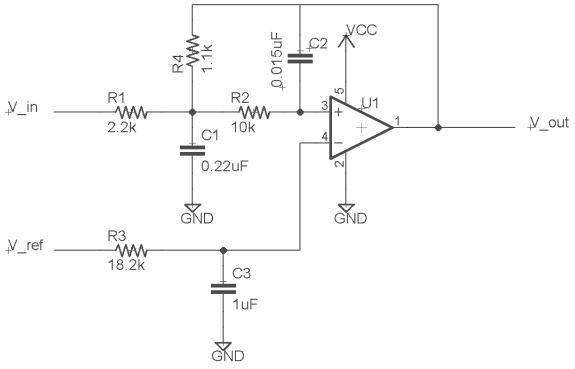


Figure 8: Low pass filter for encoder input.

Camera

The camera can be used for sensing features in the environment, or for visual odometry which gives an estimate of the robot's pose and hence, can act both as an exteroceptive sensor as well as a proprioceptive sensor. This gives a major advantage in situations where the number of sensors neeed to be minimized.

The *Leopard Imaging* camera, captures 5 megapixel HD images at speeds up to 30 frames per second. The camera has a 72° horizontal field of view, and 30° vertical. The ODROID is used to read in the camera data through an USB interface.

LIDAR

When picking a LIDAR, the choice needs to be made between using 2D or 3D LIDARs. 3D LIDARs have numerous advantages, but the high cost in addition to its weight limits the applications. 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 mount a single range finder 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 the range measurements slightly inaccurate.

The current LIDAR used is the Hokuyo URG-04LX Scanning Laser Range Finder which has a low scanning period of 100 ms/scan, and is accurate up to 2 mm. With a 240° field of view and capability to detect distances from 20 mm to 5600 mm, it provides the robot the capability of sensing the environment to a largely accurate extent.

Communication Modules

The primary channel of communication is the Wi-Fi module, a TP-Link Nano with a speed of 150 Mbps and is configured in *AP Router* mode. This lets IMP host its own Wi-Fi which is advantageous when IMP is to be used in an environment where there are no existing Wi-Fi networks available.

Another wireless communication interface is the XBee connected directly to the AutoPilot through a 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 Xbee 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 has a precise Real Time Interrupt module in the RM48, and therefore maintains a timebase for the whole process. Its interrupts are set in a half millisecond cycles. Every millisecond the encoders are read, and updates an accumulator. The IMU outputs are read every 5 ms. Both the encoder and the IMU data is 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 are 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 provide 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, the left and right wheel movement is first 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 are considered separately. This decision is made by observing the distance traveled by the left and right wheels in the same time step. If they are exactly the same, the robot is assumed to be going straight and the motion model is given by Equation (3.1).

If $r = l$,

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

Otherwise, the robot is turning, and Equations (3.2) and (3.3) are used.

If $r \neq l$,

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

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

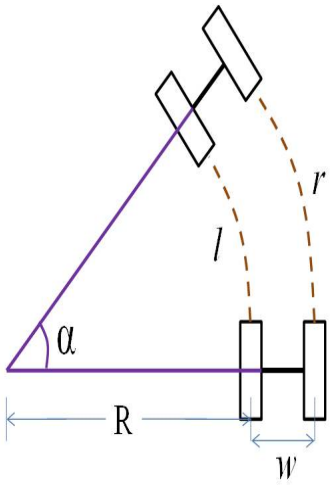


Figure 9: Motion model of differential drive platform.

where R, α and w are as shown in Figure 9.

In general, the motion model given by Equations (3.1) to (3.3), is compactly represented as:

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

Once the motion model is known, it is necessary to find its Jacobian with respect to the state. Since both the motion model has 3 dimensions the Jacobian will be a 3×3 matrix given by

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

Since the motion model is piecewise depending on the left and right wheel velocities, its Jacobian is also calculated in two parts by differentiating the respective Equations (3.1) and (3.3) as:

If $r = l$,

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

and 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 9.

Next the motion model is differentiated with respect to the noise. The process noise is modeled as due to the noise in encoder measurement and is assumed to be 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 in a piecewise fashion.

If $r = l$,

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

$$\frac{\partial f_2}{\partial l} = \frac{1}{2}(\sin\theta - \frac{l}{w}\cos\theta) \tag{3.10}$$

$$\frac{\partial f_1}{\partial r} = \frac{1}{2}(-\frac{l}{w}\sin\theta + \cos\theta) \tag{3.11}$$

$$\frac{\partial f_2}{\partial r} = \frac{1}{2}(\frac{l}{w}\cos\theta + \sin\theta) \tag{3.12}$$

$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{3.13}$$

and if $r \neq l$,

$$\frac{\partial f_1}{\partial l} = \frac{wr}{(r-l)^2}(\sin\theta' - \sin\theta) - \frac{r+l}{2(r-l)}\cos\theta' \tag{3.14}$$

$$\frac{\partial f_2}{\partial l} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{3.15}$$

$$\frac{\partial f_1}{\partial r} = \frac{-wr}{(r-l)^2}(\sin\theta' - \sin\theta) + \frac{r+l}{2(r-l)}\cos\theta' \tag{3.16}$$

$$\frac{\partial f_2}{\partial r} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{3.17}$$

$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{3.18}$$

where, $\theta' = \theta + \alpha$.

Once the Jacobian is calculated, the last component needed for the estimation according to Equation (2.14) is the process noise covariance Q, which has to contain

some information about the amount the noise in each time step. Since all noise is assumed to be due to sensor measurement noise, a diagonal matrix with the error in each encoder is chosen as the covariance as in Equation (3.19):

$$Q = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}. \tag{3.19}$$

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. Additionally, every real world encoder has a component of noise which accumulates when integrated 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 wheels with the encoders attached are not fully lubricated and free to move, that 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 the afore mentioned sources of errors in pure encoder odometry, is a major motivation for SLAM algorithms.

CHAPTER IV

2D LIDAR Feature Extraction for SLAM

The exteroceptive sensors on the Integrated Mobile Platform, described in Chapter III are the Hokuyo Laser range finder and the 5 MP camera. In this thesis, the Hokuyo LIDAR output (i.e., range data) is only utilized for feature extraction in SLAM. The feature detection algorithms extract either point features or linear features.

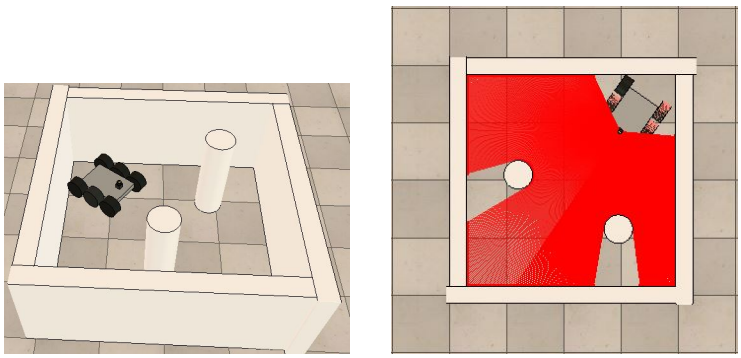
4.1 Feature Extraction with Point Features

4.1.1 Overview

A point feature can be completely defined by its position in the inertial frame. In this section, a simplistic algorithm is described that could be used to detect point features in the environment. The algorithm is mainly based on large jumps in the range measurements of the LIDAR. The drawbacks of such an algorithm are discussed and improvements using preexisting knowledge of the environment are suggested.

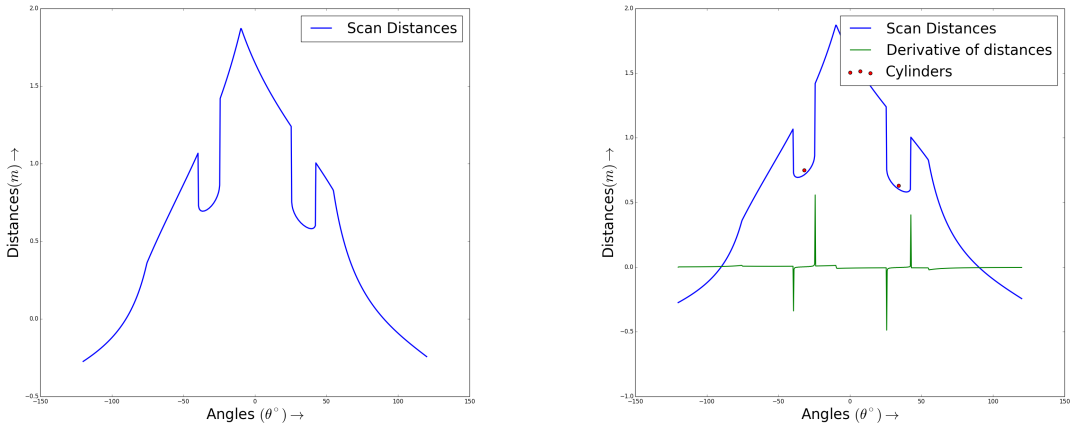
4.1.2 Feature Extraction Algorithm

To test the algorithm, a graphical simulation is built using V-REP [77]. A simple environment is set up with four walls and a few cylinders to represent point features. The environment within the four walls is referred to as the arena and is shown in Figure 10a. In the simulated arena of Figure 10a the important aspects to observe are that, the entire arena is within the range of the LIDAR, and also the cylinders placed to represent point features are all relatively away from the wall. In such an environment, the distance readings gradually increase and decrease all along the walls except when they encounter an obstacle or a *feature* resulting in the larger jumps in the distance readings that our algorithm looks for. The gradual change in 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 11a.



(a) A simulated arena. (b) Top view.

Figure 10: Simulated arena for LIDAR.



(a) LIDAR scan. (b) Detected cylinders.

Figure 11: Differential based LIDAR feature detection.

For the robot to detect the large jumps, the distance is differentiated with respect to angles. As seen in Figure 11b, this will give a specific pattern each time an obstacle (cylindrical in shape) is present. As outlined above,the feature detection may be achieved by considering the derivative of the range data. Namely, each time the derivative is larger than a fixed threshold and is a negative number, n object's beginning is found, and its end is found when a large positive value is encountered. Finding the midpoint of these two readings the location of the cylinder is found as in Figure 11b.

It is apparent that this method has a large number of drawbacks. For example, if the arena is larger than the range of the LIDAR. At the points that the distance to any surface is greater than the LIDAR range, a zero value is returned. Leaving these values as zero, will introduce breaks in the range data which will have a detrimental affect on

extracting features using the method of differentials outlined above. Another drawback is when there is an obstacle placed very close to a wall. The change in differentials corresponding to the obstacle are much smaller than the change corresponding to other landmarks. Reducing the threshold of differential jump will result in any small features of the arena such as doors being classified as obstacles. Hence it is seen, the choice of threshold for the change in derivative is a very sensitive tuning factor.

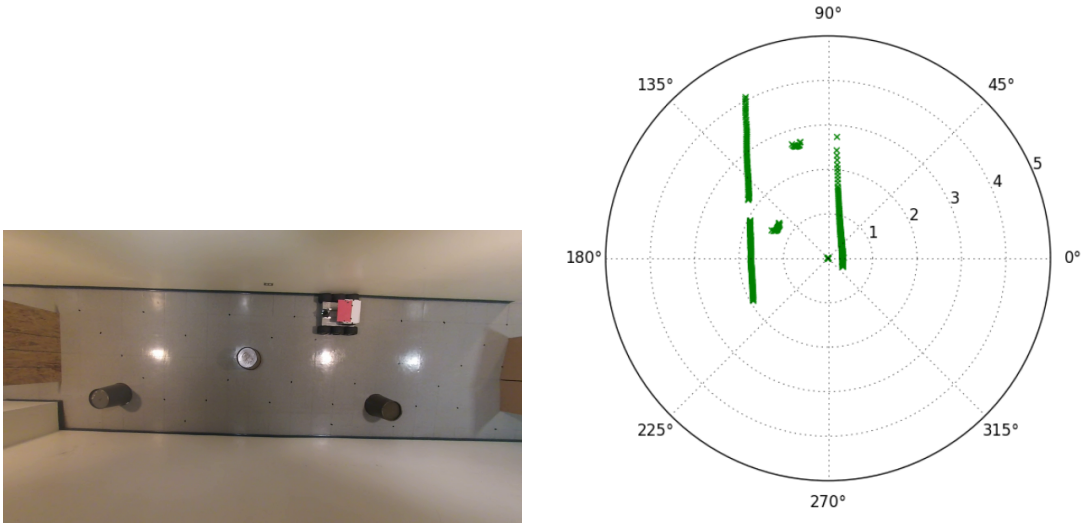
To overcome the above shortcomings, a filter can be designed based on preexisting knowledge of the nature of features. A low threshold is set for the derivative of the range, which results in a large number of obstacles being detected. This set of obstacles is then thinned down by eliminating unlikely detections. Fist filter, is that all obstacles need to have a minimum number of non-zero LIDAR readings that lie on them. Based on the a priori known information, that the obstacle is in the shape of a cylinder with a known approximate radius, the approximate angular width that a prospective obstacle should occupy is predicted. The difference in the width of a candidate obstacle and the predicted width, can be used to remove a few prospective obstacles. There is still a chance of a segment of wall having the same angular width as the predicted width. This will result in spurious landmarks, hence each of the prospective obstacles are compared to the least square fit line for the same points. If the points all fit largely along the line, then it is a segment of the walls and can be removed from the list of possible candidates. 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 FilterSLAM.

4.1.3 Examples and Validation on IMP

In a clean simulated arena, the efficacy of the first approach are 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 12.

With such an arena, the simplistic approach of changes in derivatives being obstacles is tried with multiple threshold values.Even in the best case, out of the two obstacles within the range of the LIDAR, only one obstacle gets identified correctly. The other three obstacles detected are incorrect as seen in Figure 13.



(a) Overhead view.

(b) Polar plot of laser scan.

Figure 12: Real world arena.

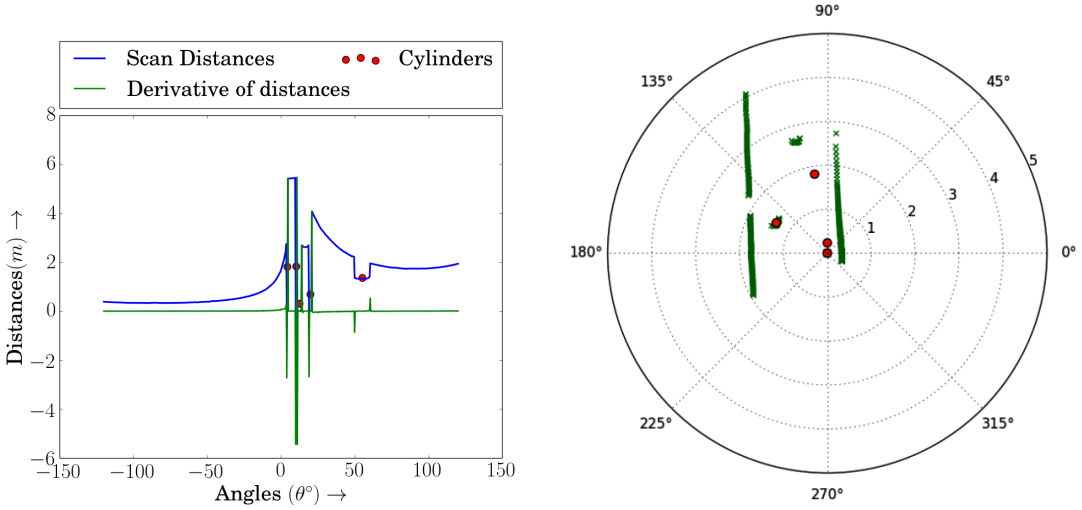


Figure 13: Spurious cylinders detected in data.

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 14.

4.1.4 Measurement Model and the Corresponding Jacobian Matrices

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

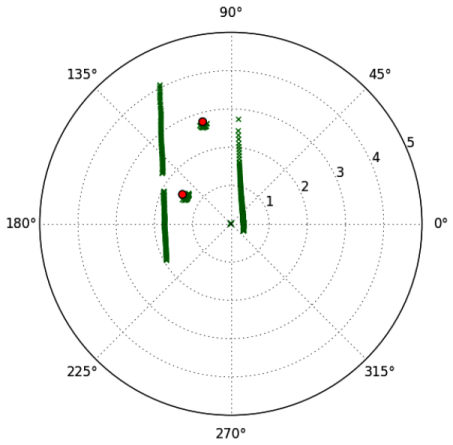


Figure 14: After filtering of candidate cylinders.

geneous transformation in 2 dimensions as shown in Equation (4.1):

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

where, (x_w, y_w) are coordinates in the world frame,while (x_r, y_r, θ_r) are the estimated pose of the robot and (x_1, y_1) are the object coordinates in the LIDAR frame.

For, data association, i.e. to check if the detected object is being seen for the first time or is being reobserved, the Euclidean distance is used. The euclidean distance between the detected object and the existing object positions is measured and if it is less than a particular threshold then the new object 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. This is a mapping that gives the laser reading (i.e. range r and bearing α) from the robot to the object given by Equation (4.2):

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

Using Equation (4.2) for both the existing landmarks and detected landmarks which have been expressed in the inertial frame, the *innovation* from Equation (2.18) can be calculated.

For the Kalman gain, according to Equation (2.17) the derivatives of the model with respect to the state vector $x \in \Re^n$ are used. The state contains both the robot pose and 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; if the measurement has been associated with an existing landmark, then it will also depend on that landmark's position. Hence for the Jacobian H, the measurement model h is differentiated as follows:

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

Each of the terms in H are derived separately as:

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

$$\frac{\partial r}{\partial y_r} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_r} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.5}$$

$$\frac{\partial r}{\partial \theta_r} = 0 \qquad \frac{\partial \alpha}{\partial \theta_r} = -1 \tag{4.6}$$

$$\frac{\partial r}{\partial x_1} = \frac{(x_r - x_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_1 - y_r)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{4.7}$$

$$\frac{\partial r}{\partial x_1} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2}. \tag{4.8}$$

The only other information needed for Kalman gain calculation is the observer error $V^T R V$ 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, the covariance of measurement noise, is a diagonal matrix with one of the eigenvalues representing the error in distance measurement of the LIDAR and the other eigenvalue being the angle measurement.R is given by,

$$R = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\alpha \end{bmatrix}. \tag{4.9}$$

Using all the measurement model and Jacobian matrices, 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 V.

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 may be necessary to find other obstacles and landmarks for path planning and mapping, knowing the locations of walls tends to be advantageous. At the very least, it can be used to remove all the laser readings that are close to a wall, therefore, 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 of interest 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 its derivatives will not be the same as in point features, as the apparent relative motion of a wall as the robot moves is very different than for point features. For example, when the robot is moving parallel to the wall, the LIDAR scans from one time step to another will be identical, but unlike in the case of point features the similarity of LIDAR scans over successive time steps does not imply that the robot has not moved.

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 this such as least squares, Split-merge and RANSAC [78]. The typical RANSAC-based line extraction in libraries such as OPENCV and PCL, attempt to robustly fit a line to a given set of points. In typical environments, since more than one wall can be seen at a time, it becomes necessary to first segment the data before using the algorithm for demo. Instead, another approach is to randomly sample points and try to fit multiple lines simultaneously.RANSAC finds these line landmarks by randomly taking a sample of the laser readings and then using a least squares approximation to find the best fit line that runs through these readings. Once this is done RANSAC checks how many laser readings lie close to this best fit line. If the number is above some threshold we can

safely assume that we have seen a line (and thus seen a wall segment). This threshold is called the consensus. This approach is explained in algorithm 1 [79].

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 less than N trials have been done,

do

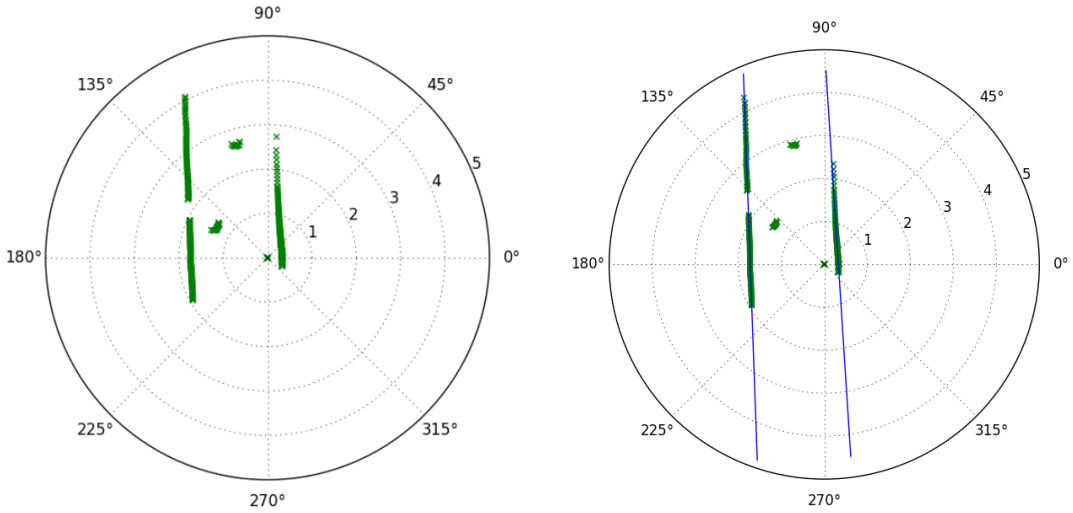
- Select a random laser data reading.
- Randomly sample S data readings within D degrees of this laser data reading
- Using these S samples and the original reading calculate a least squares best fit line L.
- Determine how many laser data readings lie within X distance from this best fit line
- If the 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 close to the old best fit line.
  - Add this best fit line to the lines that have been extracted.
  - Remove the number of readings lying close to the line from the total set of unassociated readings.

end while

This algorithm can thus be tuned based on the following parameters:

- N Max number of times to attempt to find lines.
- S Number of samples to compute initial line.
- D Degrees from initial reading to sample from.
- X Max distance a reading may be from line to get associated to line.
- C Number of points that must lie on a line for itto be taken as a line

4.2.3 RANSAC Examples



(a) Polar plot of laser scan.

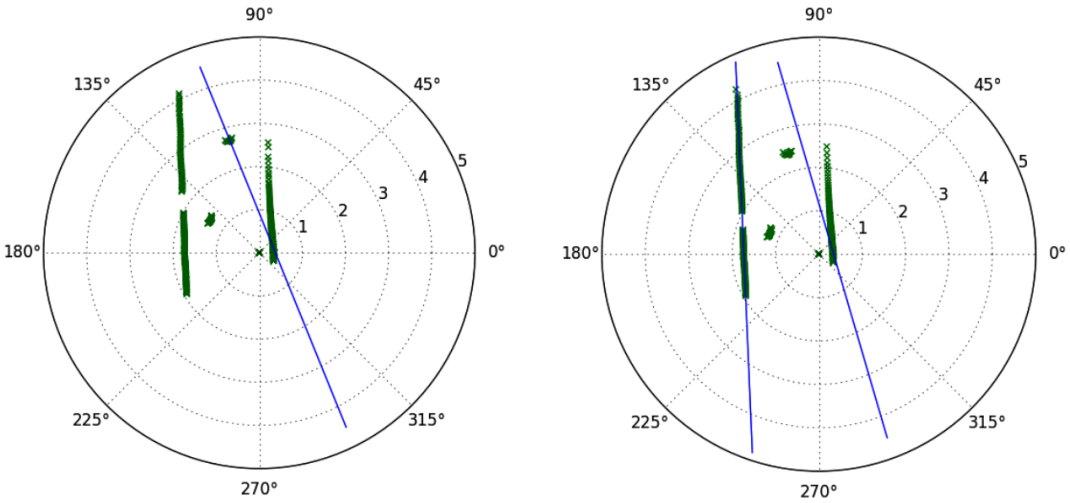
(b) Detected lines.

Figure 15: Example of walls detected by RANSAC.

Considering the same setup as in Figure 12, we can try the RANSAC algorithm on the data collected from the LIDAR. It is initially seen to give good results as seen in Figure 15. However, the robustness of the algorithm is not consistant. As we move forward though, the robustnes 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 its 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 16.

4.2.4 Hough Transform Based Feature Extraction Algorithm

In the fields of image analysis and computer vision, the Hough transform is a very popular feature extraction technique [80]. While it was originally designed to find lines in an image, it can also be used to detect any shape as long as that shape can be represented in a parametric form [81]. This algorithm can detect a shape in spite of distortions making it much more robust than the RANSAC based algorithm shown in Algorithm 1 [71]. 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



(a) Detected lines. (b) Detected lines.

Figure 16: Example of spurious walls marked by RANSAC.

can use the OpenCV implementation of Hough transform. The image is then converted to a binary image and 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,

$$r = x\cos\theta + y\sin\theta. \tag{4.10}$$

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

$$r_\theta = x_0\cos\theta + y_0\sin\theta. \tag{4.11}$$

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 (4.12). This is called the accumulator:

$$A = \begin{array}{c} \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{array} \begin{array}{c} \begin{array}{cccc} r_1 & r_2 & \dots & r_n \end{array} \\ \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \end{array}. \tag{4.12}$$

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 (4.11). 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 threshold represent the lines in the image. The (r, θ) values corresponding to these lines are returned.

An important aspect of using the ransform from OpenCV is the conversion of LIDAR data to an image and the detected lines back to the robot frame of reference. The former is straightforward. 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 III has a maximum range of 5 m. So for a resolution of 1 cm, 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 given as

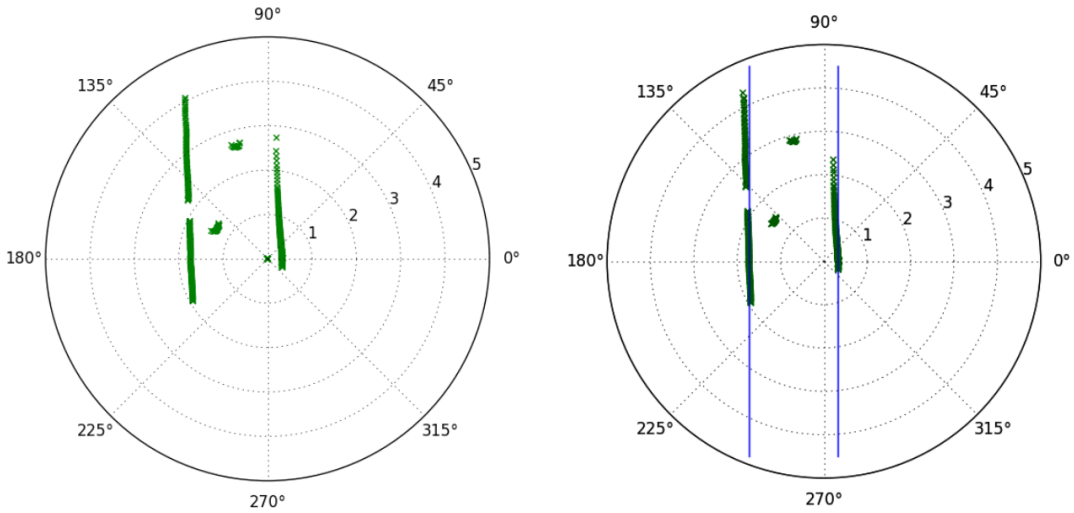
$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 500 \\ 1 & 0 & 500 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix}. \tag{4.13}$$

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.

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 (4.13) and the normal form of the equation of a line passing through these 2 points is found.

4.2.5 Hough Transform Examples

In the same arena as in Figure 12, the Hough transform is found to be really robust in finding the walls as seen in Figure 17.



(a) Polar plot of laser scan.

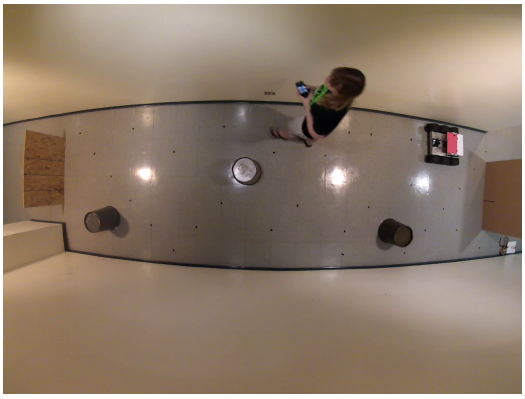
(b) Detected lines.

Figure 17: Example of walls detected by Hough transform.

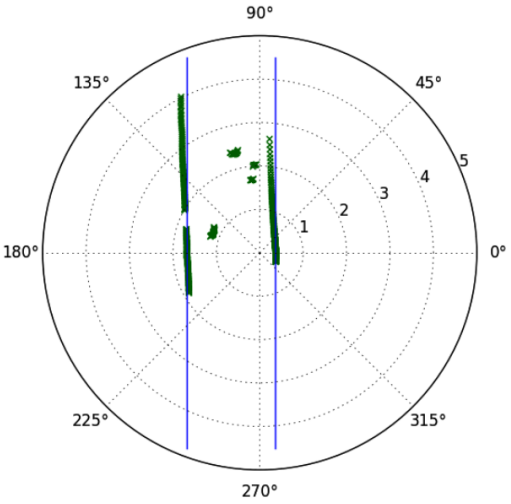
Unlike the RANSAC algorithm, the Hough transform considers all possible lines before deciding on any line and is robust to noise in the surroundings. This is especially useful when there are people walking around in the environment.

4.2.6 Measurement Model and the Corresponding Differentials

Once the lines in the data are recovered, the lines need to be expressed in an effective format 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

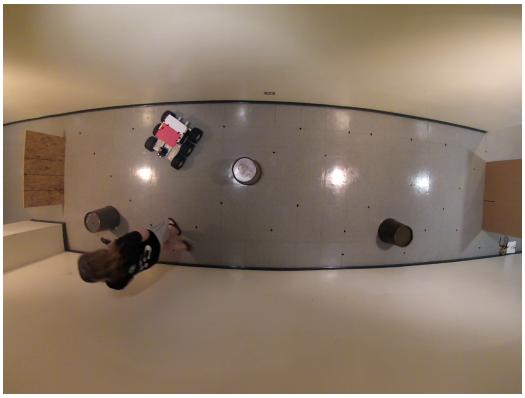


(a) Overhead view of the arena.

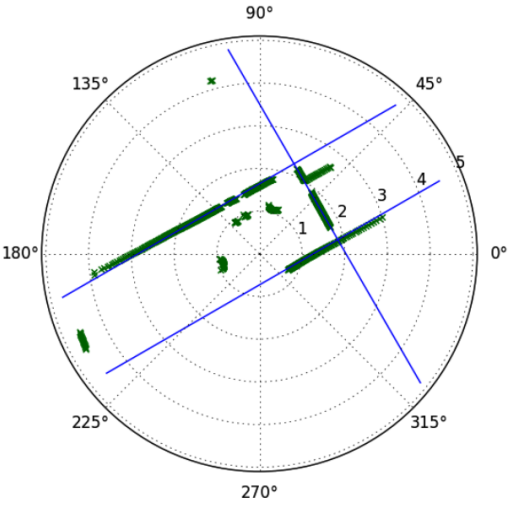


(b) Detected lines

Figure 18: Example 1 of Hough transform being robust to additional objects (e.g., people) in the environment.



(a) Overhead view of the arena



(b) Detected lines

Figure 19: Example 1 of Hough transform being robust to additional objects (e.g., people) in the environment.

the point of intersection of the normal from the origin to the line is stored. For example, in Figure 20, the line can be represented solely by the point D.

As in Section 4.1.4, the line coordinates have to be converted to the inertial frame of reference.For Example, in Figure 21, the point P_1 in robot frame is known and point P_2 has to be found in the inertial frame given coordinates of the robot in inertial frame

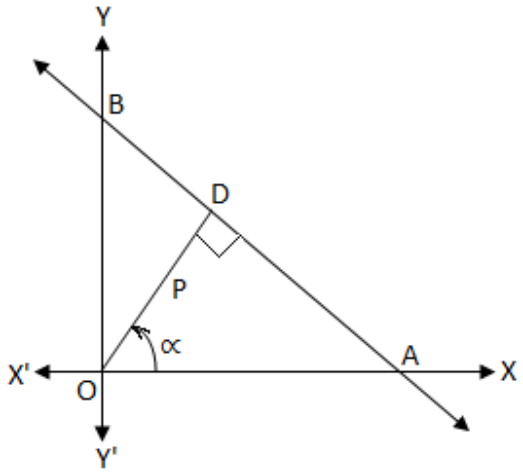


Figure 20: Normal form of a line.

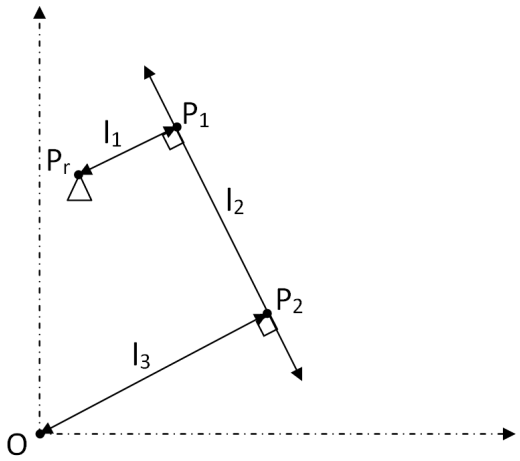


Figure 21: Line in robot frame and world frame.

as P_r(refer Algorithm 2).

Once the landmark is in the world frame,its corresponds to any existing landmark in the map can be checked. This can be checked 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 since in subsequent time steps for the measurement model is obtained by the exact opposite process: That is given a line using its normal form in inertial frame, get the *measurement* from the robot. That is, we need the perpendicular distance from the robot to the wall and the angle of the

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

1. Convert point P_1 from robot frame 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 can be foundby connecting few points.
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. The coordinates of point P_2 are obtained by solving equations for l_3 and l_2.

laser beam which would hit the wall perpendicularly. For this given coordinate pf point P_2 as (x_2, y_2) and the robot pose as (x_r, y_r, θ_r), the Cartesian coordinates P_1 is calculated as:

$$x_1 = x_2 - \frac{y_2(x_2 y_r - y_2 x_r)}{(x_2^2 + y_2^2)} \qquad y_1 = y_2 + \frac{x_2(x_2 y_r - y_2 x_r)}{(x_2^2 + y_2^2)}. \tag{4.14}$$

Once P_1 is known,the perpendicular distance r and angle α are calculated using Equation (4.15):

$$r = \sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2} \qquad \alpha = \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) - \theta_r, \tag{4.15}$$

giving measurement model h as per Equation (4.16)

$$h = \begin{bmatrix} r \\ \alpha \end{bmatrix}. \tag{4.16}$$

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). Each of the terms on H is derived individually. The A factor common to all the components of the Jacobian is given by:

$$\beta = \frac{x_r x_1 - x_1^2 - y_1^2 + y_r y_1}{x_1^2 + y_1^2}. \tag{4.17}$$

The part of the Jacobian with respect to the position of the robot is:

$$\frac{\partial r}{\partial x_r} = 2x_1\beta \quad \frac{\partial r}{\partial y_r} = 2y_1\beta \quad \frac{\partial r}{\partial \theta_r} = 0 \tag{4.18}$$

$$\frac{\partial \alpha}{\partial x_r} = 0 \quad \frac{\partial \alpha}{\partial y_r} = 0 \quad \frac{\partial \alpha}{\partial \theta_r} = -1. \tag{4.19}$$

In these equations, it is seen that, α is independent of the location of the robot and is inversely proportional to the orientation. Since r and α are in the robot frame of reference. For a given wall, as long as the robot is facing the same direction, the angle at which the wall is seen remains the same. Similarly the perpendicular distance r between a robot center and a wall will remain the same irrespective of the orientation of the robot.

Differentiating with the obstacle positions gives us the following:

$$\frac{\partial r}{\partial x_1} = -2x_1\beta^2 - 2\beta(2x_1 - x_r) \tag{4.20}$$

$$\frac{\partial r}{\partial y_1} = -2y_1\beta^2 - 2\beta(2y_1 - y_r) \tag{4.21}$$

$$\frac{\partial \alpha}{\partial x_1} = \frac{-y_1}{x_1^2 + y_1^2} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{x_1}{x_1^2 + y_1^2}. \tag{4.22}$$

Having the Jacobian matrices we can use the same observer error given by Equation (4.9) 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.

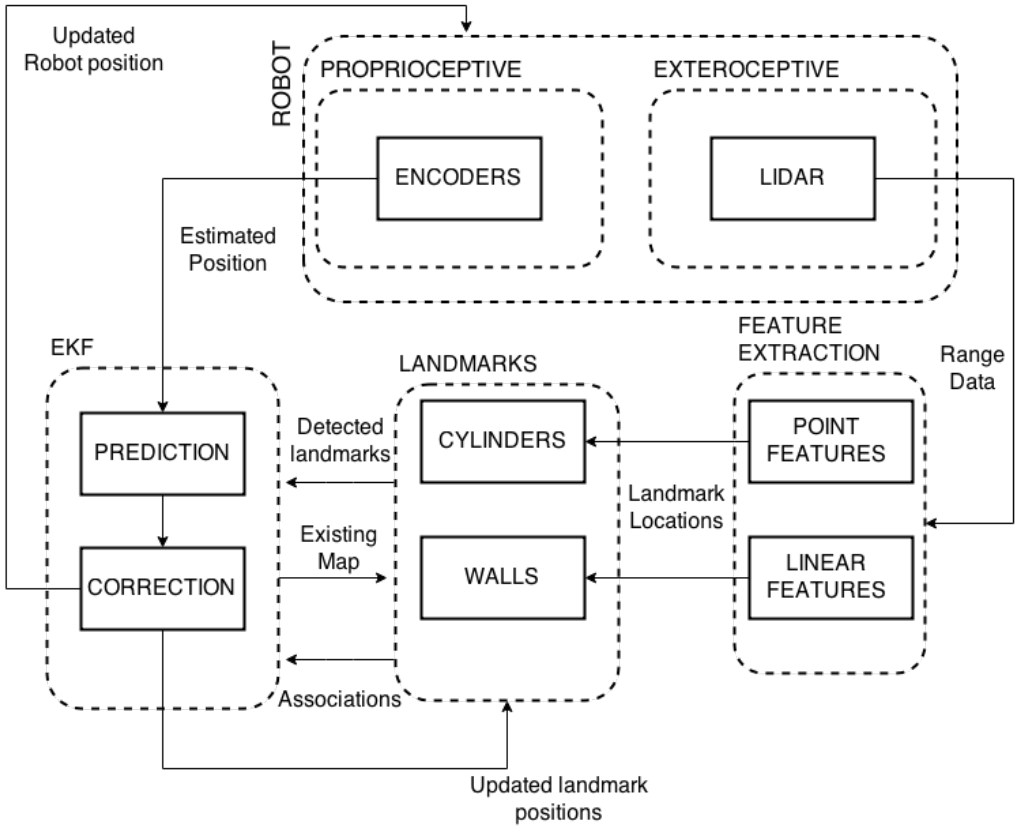


Figure 22: SLAM Implementation using only LIDAR.

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 23. 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 23. 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.

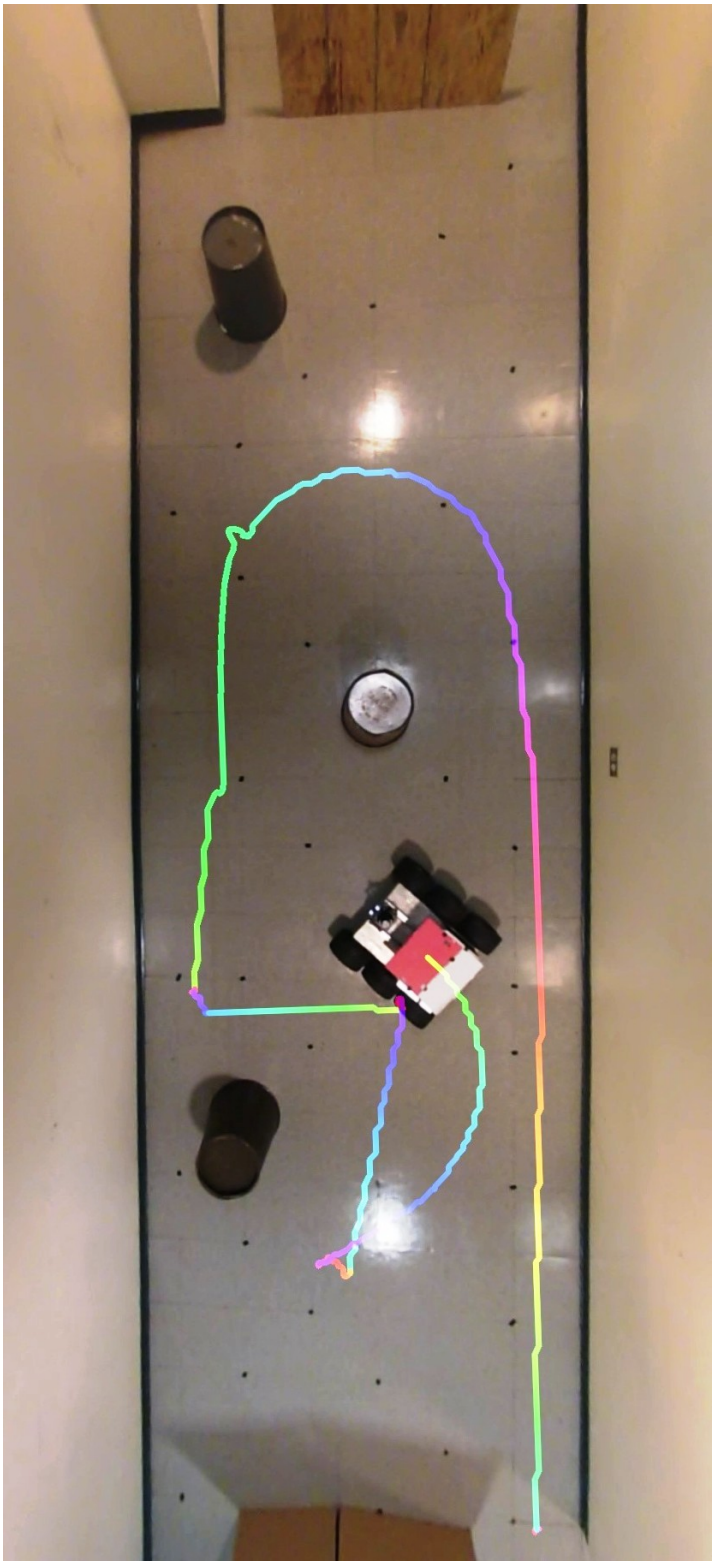


Figure 23: Controlled environment and path reconstructed by an overhead camera.

5.2.1 Using Point Features

In Figure 23 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 24. 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 24, as soon as the robot has lost the first cylinder from its field of view, only the middle

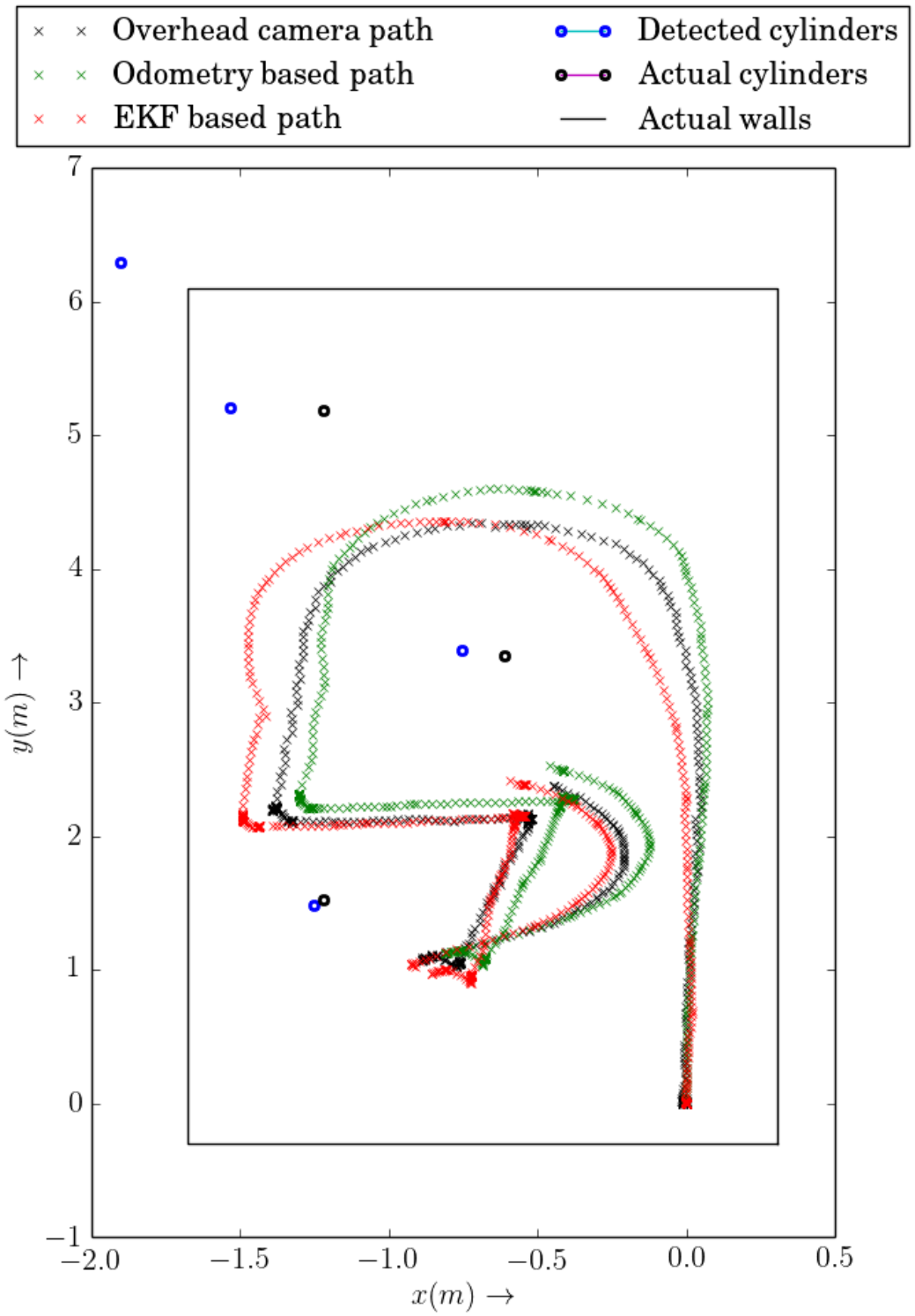


Figure 24: Controlled environment and path reconstructed by detecting point features in LIDAR data. Path starts from the origin.

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 23, 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 16. 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 (4.12) 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 20. 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 23. 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 25.

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

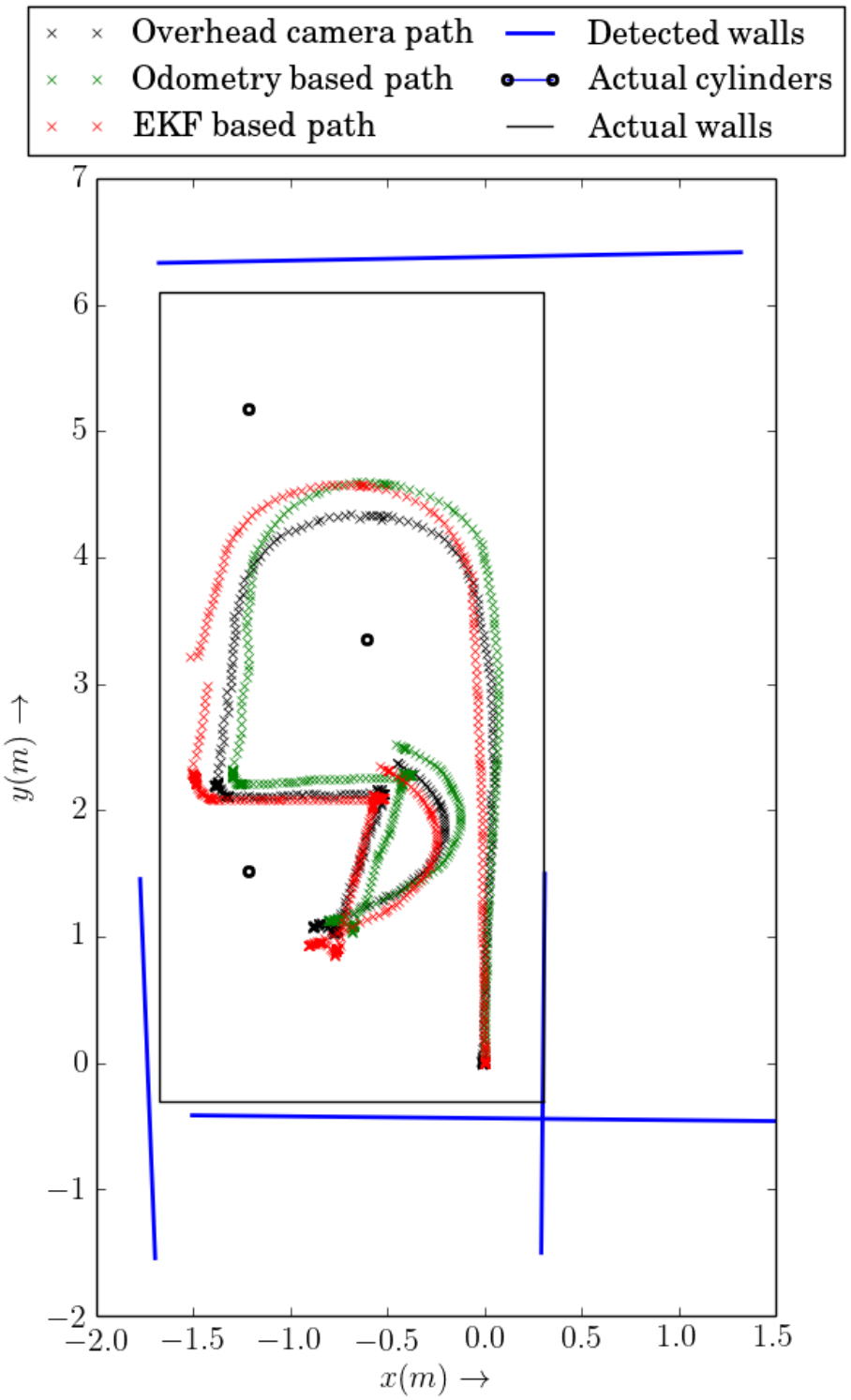


Figure 25: Controlled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin.

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 25, 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 26.

From the path, it is seen that a few of the errors from the previous two reconstructions have been corrected. For example, in Figure 24, 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 25, 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

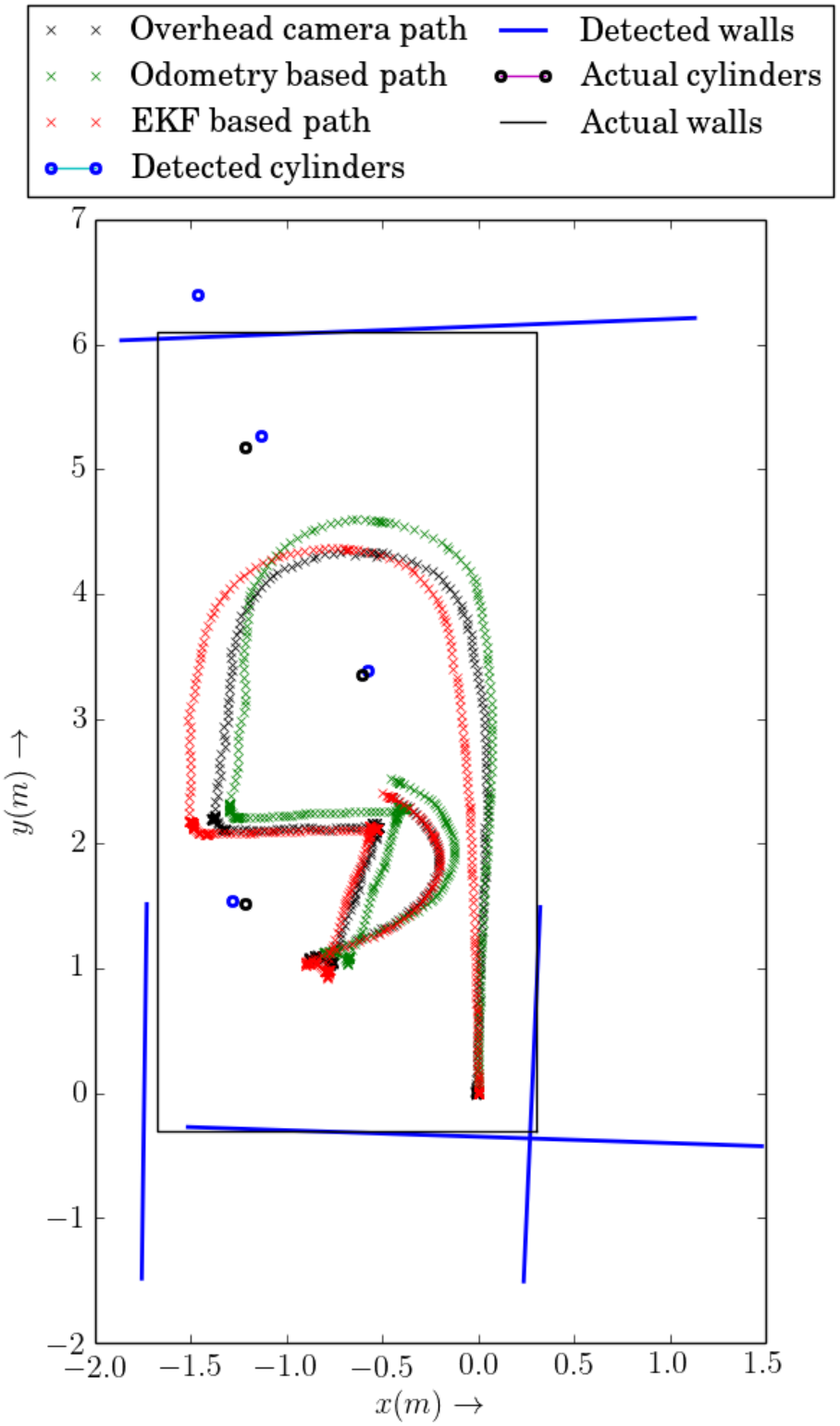


Figure 26: Controlled environment and path reconstructed by detecting both point and linear features in LIDAR data. Path starts from the origin

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 27: 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 28: 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 27. There are also people walking around and entering and leaving through those doors. As seen in Figure 28, 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 29. 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

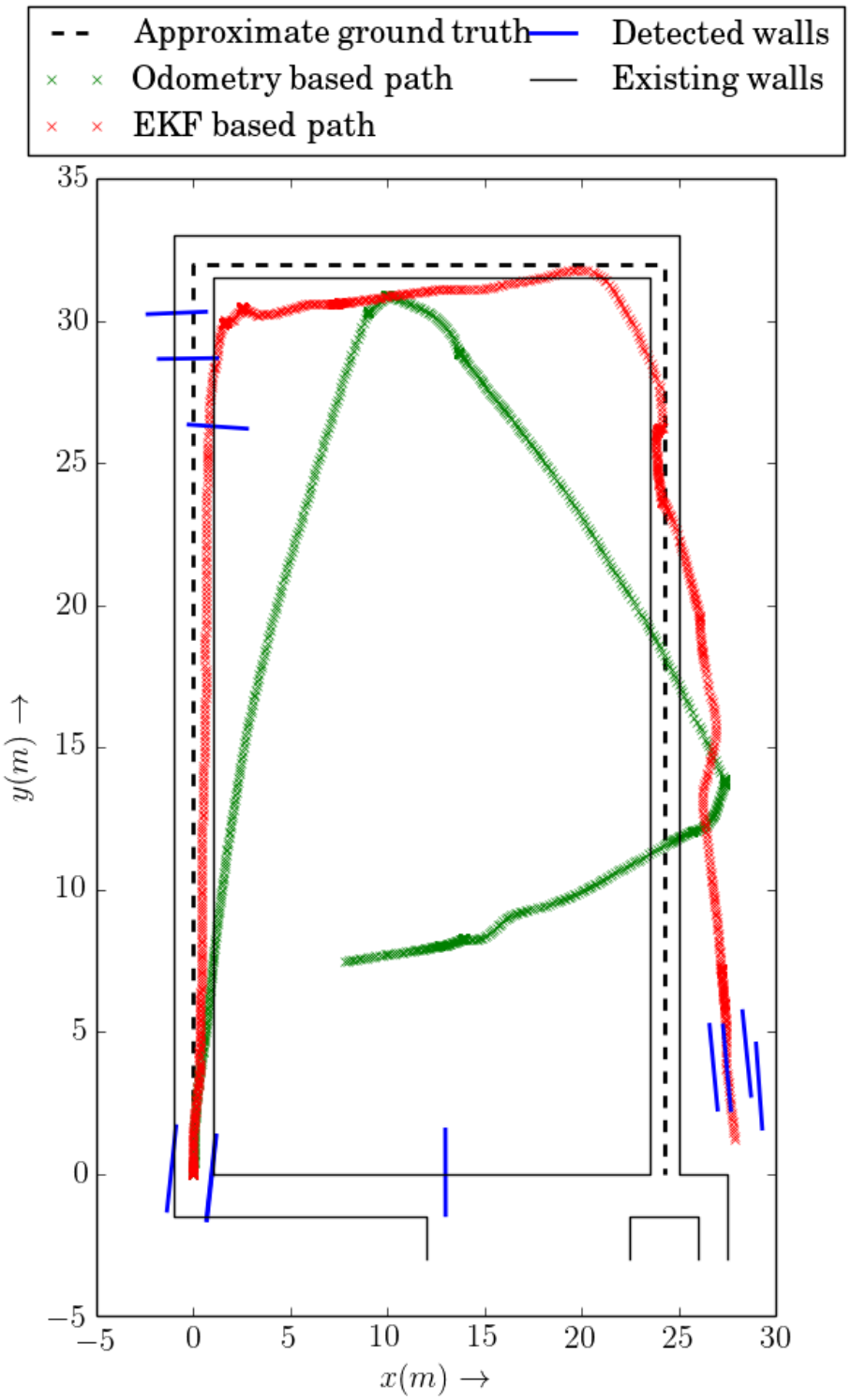


Figure 29: Uncontrolled environment and path reconstructed by detecting linear features in LIDAR data. Path starts from the origin.

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 Sections 4.1.3 and 5.2.2, are robust and do not need to be changed for similar environments.

CHAPTER VI

Conclusion

This thesis presented an implementation of Simultaneous Localization and Mapping on the Integrated Mobile Platform. A general overview of EKF and its application to SLAM was given. The Integrated Mobile Platform, a differential drive platform with capabilities of real-time implementation of SLAM was designed and built. The platform was equipped with encoders for odometry, a LIDAR and a camera for environmental sensing. A multi level system architecture was set up. The motion models for the platform and how they are used in EKF was discussed. Two kinds of features were detected from the LIDAR data: 1) Point features which consisted of cylinder like objects placed in the arena, and 2) A linear feature extractor which was used to reconstruct the walls of the arena. A simplistic algorithm for point features was shown to be not robust enough to be used in a real world applications. Modifications to the algorithms were proposed. For linear features, two algorithms were discussed, one based on RANSAC and one based on Hough transform. The Hough transform based algorithm was shown to be robust even to humans moving through the environment. Both the linear and point features were utilized in the SLAM algorithm and results were presented. The reconstructed paths were analyzed to obtain intuition about the process. It was seen that in a controlled environment, the combination of point features and linear features give the least error. In an uncontrolled environment with people moving across the field of view of the LIDAR, only the Hough transform is robust enough to be used and it is seen to reconstruct the path with reasonable accuracy.

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 should also be considered and 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 augmenting encoder based odometry by visual odometry. With visual odometry, the SLAM implementation will not be depen-

dent on the robot having encoders and can be used for a variety of platforms. 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. The computational power on the platform is capable of running more complex autonomous navigation algorithms. The maps obtained by SLAM can be used as a starting point for algorithms such as path planning. Lastly,sensor fusion for LIDAR and camera to detect and discern features togrther are an important area of future work.

References

[1] M.D. Hasler, P. Gaynor, and D. Carnegie. Simulation for improvement of dynamic path planning in autonomous search and rescue robots. In *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, pages 1603–1608, Dec 2009.

[2] A. Macwan and B. Benhabib. A multi-robot coordination methodology for autonomous search and rescue. In *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pages 675–680, Sept 2009.

[3] 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*, 278(5344):1743–1748, 1997.

[4] 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*, volume 2, pages 1322–1328 vol.2, 1999.

[5] P. Tripathi, H. Singh, K.S. 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*, pages 47–51, Feb 2014.

[6] G. Lakemeyer and B. Nebel. *Exploring Artificial Intelligence in the New Millennium*. The Morgan Kaufmann Series in Artificial Intelligence Series. Morgan Kaufmann Publishers, 2003.

[7] 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*, volume 3, pages 4270–4275 vol.3, Sept 2003.

[8] F. Hidalgo and T. Braunl. Review of underwater slam techniques. In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, pages 306–311, Feb 2015.

[9] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual slam for flying vehicles. *Robotics, IEEE Transactions on*, 24(5):1088–1093, Oct 2008.

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

[11] 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*, pages 3834–3839, Aug 2005.

[12] O. Stasse, A.J. 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*, pages 348–355, Oct 2006.

[13] Georg Arbeiter, Jan Fischer, and Alexander 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)*, pages 1–5, June 2010.

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

[15] 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*, 22(1):22–33, March 2015.

[16] Li Guangshun and Shi Hongbo. Study of technology on path planning for mobile robots. In *Control and Decision Conference, 2008. CCDC 2008. Chinese*, pages 3295–3300, July 2008.

[17] Nan Ying, L. Eicher, Wang Xunzhang, G.G.L. Seet, and M.W.S. Lau. Real-time 3d path planning for sensor-based underwater robotics vehicles in unknown environ-

ment. In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, volume 3, pages 2051–2058 vol.3, 2000.

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

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

[20] 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*, pages 1–6, Nov 2013.

[21] Shoudong Huang and G. Dissanayake. Convergence and consistency analysis for extended kalman filter based slam. *Robotics, IEEE Transactions on*, 23(5):1036–1049, Oct 2007.

[22] Shoudong Huang and G. Dissanayake. Convergence analysis for extended kalman filter based slam. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 412–417, May 2006.

[23] 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*, pages 365–370, Dec 2008.

[24] Jiantong Cheng, Jonghyuk Kim, Zhenyu Jiang, and Xixiang Yang. Compressed unscented kalman filter-based slam. In *Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on*, pages 1602–1607, Dec 2014.

[25] Su Kuifeng, Deng Zhidong, and Huang Zhen. Novel slam algorithm for ugvs based on unscented kalman filtering. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, volume 3, pages 63–67, May 2012.

[26] 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*, pages 1–6, Sept 2007.

[27] R. Havangi, M.A. Nekoui, H.D. Taghirad, and M. Teshnehlab. Slam based on intelligent unscented kalman filter. In *Control, Instrumentation and Automation (ICCIA), 2011 2nd International Conference on*, pages 877–882, Dec 2011.

[28] Yutong Zang, Kui Yuan, Wei Zou, and Huosheng Hu. A two-step particle filter for slam of corridor environment. In *Information Acquisition, 2006 IEEE International Conference on*, pages 370–375, Aug 2006.

[29] B.D. 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*, pages 1583–1588, Sept 2014.

[30] 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*, pages 3575–3580, May 2006.

[31] Hee Seok Lee and Kyoung Mu Lee. Multiswarm particle filter for vision based slam. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 924–929, Oct 2009.

[32] 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*, pages 1576–1581, Sept 2008.

[33] Heon-Cheol Lee, Seung-Hee Lee, Seung-Hwan Lee, Tae-Seok Lee, Doo-Jin Kim, Kyung-Sik Park, Kong-Woo Lee, and Beom-Hee Lee. Comparison and analysis of scan matching techniques for cooperative-slam. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2011 8th International Conference on*, pages 165–168, Nov 2011.

[34] Rongbing Li, Jianye Liu, Ling Zhang, and Yijun Hang. Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments. In *Inertial Sensors and Systems Symposium (ISS), 2014 DGON*, pages 1–15, Sept 2014.

[35] Daobin Wang, Huawei Liang, Tao Mei, Hui Zhu, Jing Fu, and Xiang Tao. Lidar scan matching ekf-slam using the differential model of vehicle motion. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 908–912, June 2013.

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

[37] Myung-Jin Jung, Hyun Myung, Sun-Gi Hong, DongRyeol Park, Hyoung-Ki Lee, and SeokWon 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*, pages 371–376, Oct 2004.

[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*, pages 1057–1069, May 2010.

[39] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, June 2007.

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

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

[42] Hyon Lim and Young Sam Lee. Real-time single camera slam using fiducial markers. In *ICCAS-SICE, 2009*, pages 177–182, Aug 2009.

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

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

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

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

[47] Congdao Han, Zhiyu Xiang, Jilin Liu, and Eryong Wu. Stereo vision based slam in outdoor environments. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pages 1653–1658, Dec 2007.

[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*, pages 125–128, April 2009.

[49] S. Takezawa, D.C. 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*, volume 2, pages 1866–1871 vol.2, Sept 2004.

[50] Jae-Hean Kim and Myung Jin Chung. Slam with omni-directional stereo vision sensor. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 442–447 vol.1, Oct 2003.

[51] Jan Hartmann, Dariush Forouher, Marek Litza, Jan Helge Kluessendorff, and Erik Maehle. Real-time visual slam using fastslam and the microsoft kinect camera. In *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1–6, May 2012.

[52] M.F.A. Ghani, K.S.M. Sahari, and Loo Chu 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*, pages 776–781, Dec 2014.

[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*, pages 85–89, Oct 2014.

[54] J.W. 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*, pages 1783–1786, Oct 2013.

[55] E. Jose and M.D. 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*, volume 1, pages 155–160 Vol. 1, Dec 2004.

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

[57] M.F. Fallon, J. Folkesson, H. McClelland, and J.J. Leonard. Relocating underwater features autonomously using sonar-based slam. *Oceanic Engineering, IEEE Journal of*, 38(3):500–513, July 2013.

[58] Sungjin Jo, Hyukdoo Choi, and Euntai 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*, pages 1461–1464, Oct 2012.

[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*, pages 139–143, Sept 2010.

[60] Fang Zhang, Changguo Shen, and Xuemei 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, volume 1, pages 360–364, Aug 2013.

[61] C. Vianchada, P.J. Escamilla, M.N. Ibarra, J.M. Ramirez, and P. Gomez. Attitude estimation using fusion of monocular slam and inertial sensors. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 12(6):977–984, Sept 2014.

[62] Edwin Brock Olson, Seth Teller, and John Leonard. *Robust and efficient robotic mapping*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2008.

[63] A. Diosi, L. Kleeman, Albert Diosi, and Lindsay 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] Geovany Araujo Borges and Marie-José Aldon. Line extraction in 2d range images for mobile robotics. *J. Intell. Robotics Syst.*, 40(3):267–297, July 2004.

[65] P. Nunez, R. Vazquez-Martin, J.C. 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*, pages 1167–1172, May 2006.

[66] Jos E. Guivant, Favio R. Masson, and Eduardo M. Nebot. Simultaneous localization and map building using natural features and absolute information. *Robotics and Autonomous Systems*, 40(23):79 – 90, 2002. Intelligent Autonomous Systems - {IAS} -6.

[67] Michael Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[68] Matthew R. Walter, Ryan M. Eustice, and John J. Leonard. Exactly sparse extended information filters for feature-based slam. *Int. J. Rob. Res.*, 26(4):335–359, April 2007.

[69] Yufeng 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*, volume 1, pages 1227–1233 vol.1, Sept 2003.

[70] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

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

[72] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[73] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[74] *Stochastic Models: Estimation and Control: v. 1: Estimation and Control:*. Number v. 1 in Mathematics in Science and Engineering. Elsevier Science, 1979.

[75] O.L.R. Jacobs. *Introduction to control theory*. Oxford sciences publications. Oxford University Press, Incorporated, 1993.

[76] R.G. Brown and P.Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises, 4th Edition*. Wiley Global Education, 2012.

[77] M. Freese E. Rohmer, S. P. N. Singh. V-rep: a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[78] 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*, pages 1929–1934, Aug 2005.

[79] Søren Riisgaard and Morten Rufus Blas. Slam for dummies. *A Tutorial Approach to Simultaneous Localization and Mapping*, 22(1-127):126, 2003.

[80] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[81] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.