VITA

March 1991	Born
June 2009	Graduated High School
May 2013	Bachelor of Technology in Electrical Engineering
May 2015	Master of Science in Electrical Engineering *(expected)*

TABLE OF CONTENTS

VITA **i**

LIST OF FIGURES **iv**

LIST OF TABLES **v**

I. Introduction **1**

II. Overview of Extended Kalman Filter based Simultaneous Localization and Mapping **3**

2.1 Overview of SLAM 3
2.2 Kalman Filter 4
2.3 Extended Kalman Filter 5
2.4 Application of Extended Kalman Filter for SLAM 6

III. Integrated Mobile Platform **8**

3.1 Overview 8
3.2 Mechanical Design 9
3.3 Electrical Components 9
3.4 Control Diagram 9
3.5 Code Structure 9

IV. Encoder Based Odometry for State Estimate **10**

4.1 Overview 10
4.2 Motion model and the corresponding differentials 10
4.3 Experimental results 13

V. Feature Extraction with Point Features **14**

5.1 Overview 14

5.2 Feature Extraction Algorithm 14

5.3 Measurement Model and the corresponding Jacobian matrices 16

5.4 Experimental results 17

VI. RANSAC based linear feature Extraction **19**

6.1 Overview 19

6.2 Feature Extraction Algorithm 19

6.3 Measurement Model and the corresponding differentials 20

6.4 Experimental results 23

VII. Hough transform based linear feature extractor **24**

7.1 Overview 24

7.2 Feature Extraction Algorithm 24

7.3 Measurement Model and the corresponding differentials 24

7.4 Experimental results 24

VIII. Visual odometry **25**

8.1 Overview 25

8.2 Conceptual Description 25

8.3 Implementation and integration with EKF SLAM 25

8.4 Experimental results 25

IX. Conclusion **26**

LIST OF FIGURES

1 Motion model of Differential drive platform 11

2 Simulated Arena for LIDAR 14

3 Differential based LIDAR feature detection 15

4 Normal form of a line 21

5 Line in robot frame and world frame. 21

LIST OF TABLES

CHAPTER I

Introduction

Autonomous robots are growing increasingly popular in various fields. They occupy a major fraction of robotics based research today. Autonomous robots take many forms, from mobile robots to fixed manipulators. Even within mobile robots, the diversity in the type of locomotion, the terrain and the targeted use is huge. There are autonomous robots intended for indoor use, for rough terrains, for underwater and also aerial autonomous robots.

But irrespective of terrain, autonomous mobile robots all have one challenge in common. The ability to map their surroundings while simultaneously figuring out where exactly they are in that map. This concept of Simultaneous Localization and Mapping or SLAM is a vast field of research with numerous algorithms and implementations. Each of these algorithms, have their unique advantages and drawbacks making it impossible to pinpoint a generalized *best algorithm.*

One of the algorithms used for SLAM is the Extended Kalman Filter. The major advantage of this algorithm is that it is very simple to implement on an on board computer and can be used for real time SLAM applications. Going a bit into EKF SLAM we first use the proprioceptive sensors to get an estimate of our motion. This is called the *Prediction* step. Then we use the exteroceptive sensors to get and idea of how our environment looks like. We try to focus on specific features we know to look for or on generic features. This is called *Feature Extraction.* In subsequent time steps, we compare the environmental features we see, to our previous idea of our environment and use the difference to both improve both our position estimate and our map of our surroundings. Which is the *Correction* step. Hence we can see that, it consists of many sub parts within it which we shall discuss in depth at a later point. Each of these sub parts offer their own challenges and are again focused on the specific application. By making the base algorithm a simple one like EKF, we can plug in different methods and algorithms for each of the subparts without disturbing the rest to give a good idea of their various advantages and disadvantages.

For testing a basic algorithm like EKF, we use a classic experimental platform which is a differential drive ground vehicle. We use a six wheeled platform with the four exterior wheels being powered and the interior wheels being used for odometry. With this platform we try to map indoor environments such as corridors doors etc. The ground vehicle is also equipped with an autopilot including an Inertial Measurement Unit, an on board computer as well as exteroceptive sensors such as LIDAR and camera.

Going into the subparts of SLAM, the first subpart concentrated on in this thesis is the Feature Extraction. Features can be identifiable features of robust generic features. Features can be extracted both from the camera and from the LIDAR. When dealing with the range data from the LIDAR, a simplistic algorithm is to use rapid changes of the range as features. While this is a simple algorithm used for trying out SLAM for the first time, it has several drawbacks which are subsequently explored. Also modifications for this algorithm are suggested and tried for making it more efficient.

A number of studies have shown that for indoor SLAM, which is the target environment in this thesis, linear features such as walls and doors are found to give much better results []. There are a variety of methods [] to find these linear features. A few of them are implemented in python [] and compared with respect to how effective they are for SLAM and their computation time.

Once a good feature extractor which good results for EKF SLAM is implemented, the prediction subpart is analyzed. It essentially consists of using the odometry to get an estimate of the new position of the robot given it's old position. Initially when testing the other algorithms for feature extraction, a simple method of acquiring odometry was used. It was acquired through the encoders on the interior wheels of the mobile platform. Now, once a good feature extractor was implemented, there was scope to explore additional odometry methods such as visual odometry. Such a prediction which does not require the use of encoders, is really useful for application in a wide variety of platforms [] where it is difficult to get odometry like in legged systems or aerial vehicles.

Once a good visual odometry system is implemented, we combine that with the linear feature extraction based correction to get a good SLAM implementation that is real time implementable.

CHAPTER II

Overview of Extended Kalman Filter based Simultaneous Localization and Mapping

2.1 Overview of SLAM

Simultaneous Localization and Mapping, commonly abbreviated as SLAM, is concerned with the problem of a mobile robot building a map of the unknown environment, while at the same time navigating the environment using the map. Slam consists of multiple parts. To give a broad overview, the important parts of SLAM are:

- **State Estimation:** Proprioceptive sensors are used to estimate where the robot might be in the map.
- **Landmark Extraction:** Use exteroceptive sensors on the robot to detect prominent features of the environment. These can either be generalized features or specific landmarks about which we have prior information
- **Data Association:** The landmarks or features detected in the previous step is associated with existing features in the map that has been generated till now.
- **State Update:** The position of the robot is corrected based on the deviations of the features with the features on the existing map.
- **Landmark Update:** The positions on the features are also corrected based on the correction in the position of the robot.

While there are many algorithms for SLAM most contain these basic steps in some form or the other. There are also many ways to solve each of the smaller parts. The purpose of each part is better understood having an understanding of the Extended Kalman Filter.

2.2 Kalman Filter

The popular Kalman Filter [1,2] is a consists of mathematical equations that give an efficient computational recursive solution of the least squares problem. The filter has several advantages such as: it supports estimation of the past, present and future states, even when the precise nature of the system is not known.

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

$$[h]x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{(2.1)}$$

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

$$[h]z_k = Hx_k + v_k \tag{(2.2)}$$

The random variables $w_k and v_k$ represent the process and measurement noise respectively. They are assumed to be independent of each other and have a normal probability distribution functions given by equations (2.3) and (2.4)

$$p(w) \approx N(0, Q), \tag{(2.3)}$$

$$p(v) \approx N(0, R) \tag{(2.4)}$$

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

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

Defining the predicted state estimate as $\hat{x}_k^- \in \Re^n$ and the corrected estimate to be $\hat{x}_k \in \Re^n$ the actual equations for the prediction are given by (2.5) and (2.6) where A and B are from equation (2.1) and Q is from equation (2.3) [2].

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{(2.5)}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{(2.6)}$$

The first step in the correction step is to find a *gain* or *blending factor* that minimizes the error covariance. This is found by equation (2.7) [1,3–5]. The next step is to actually measure the process to get z_k and generate a better estimate using equations (2.8) and (2.9) [2].

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{(2.7)}$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \tag{(2.8)}$$

$$P_k = (I - K_k H) P_k^- \tag{(2.9)}$$

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

2.3 Extended Kalman Filter

In the previous section 2.2, the Kalman filter gives us a set of equations to estimate the state $x \in \Re^n$ of a discrete time process described by a set of *linear* equations. But, most applications including SLAM, consists of systems governed by *non-linear* equations. The Extended Kalman filter is the technique of linearizing a non-linear dynamics around the current mean and covariance for use in a Kalman filter.

Similar to the Taylor series, we can linearize the estimation around the current estimate using the partial derivatives of the process and measurement functions. To do this we slightly modify the equations in section 2.2. We again start with the assumption that the process has a state vector $x \in \Re^n$, but it is described a *non-linear* stochastic difference equation (2.10). This equation relates the state at time step x_{k-1} and x_k. The measurement $z \in \Re^m$ is given by equation (2.11) where h is also *non-linear.*

$$x_k = f(x_{k-1}, u_k, w_{k-1}) \tag{(2.10)}$$

$$z_k = h(x_k, v_k) \tag{(2.11)}$$

Here, the random variables w_k and v_k are again the process and measurement noise as in equations (2.3) ans (2.4). Similar to the section 2.2, we have 2 groups of equations. The prediction step is given by equations (2.12) and (2.13).

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k, 0) \tag{(2.12)}$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \tag{(2.13)}$$

Here we can see the first equation is the same as (2.10), but the w_k variable is replaced by zero as in practice we don ot actually know the noise and $\hat{x}_k^-$ is just an estimate. The second equation is similar to the Kalman filter equations except here A and W are Jacobian matrices of partial derivatives with respect to x and w respectively. At each time step they are recalculated according to equations (2.14) and (2.15)

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{(2.14)}$$

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

As with the basic Kalman filter ,the equations for the correction step given by (2.16) to (2.18) correct the estimate of the state and covariance based on a measurement z at time k

$$K_k = P_k^- H_k^T ({}_k H P_k^- H_k^T + V_k R V_k^T)^{-1} \tag{(2.16)}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0)) \tag{(2.17)}$$

$$P_k = (I - K_k H_k) P_k^- \tag{(2.18)}$$

As in the previous case, h is from equation (2.11) and v_k is approximated to zero for finding an estimate of state. In the *Kalman gain* and covariance equations, H and V are again Jacobian matrices of the partial of h with respect to x and v respectively. They are also recalculated each step using equations (2.19) and (2.20)

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_k, 0) \tag{(2.19)}$$

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

2.4 Application of Extended Kalman Filter for SLAM

In the case of Simultaneous Localization and Mapping, the state we are trying to estimate is a combination of the position of the robot as well as the environment. The variable x can be assumed as a vector of the robot's pose and the position of various features in the environment that we are interested in. These may be explicitly defined features or generic in nature. The error in our knowledge of the robot pose and our knowledge of the environment is represented by the covariance P.

Of the subparts mentioned in section 2.1, it is now apparent that the *State Estimation* step involves the *Prediction* step of the Extended Kalman Filter . The function f from equation (2.10) is our system model. It typically involves the motion model of the robot. It is a function that gives us an estimate of the it's pose. As for the input u_k, it can take any information about the motion that it undergoes in time step k. It can be the commands given to the robot or , more commonly, the measurements of the proprioceptive sensors such as encoders. This motion model is differentiated with respect to both itself and the noise model to get the Jacobian matrices used in equation (2.13). The specific motion and noise models used will be discussed in further chapters.

Now that we have an estimate of the robot pose at time k, we need to improve upon it. For this we need a measurement of the environment using any exteroceptive sensor on the robot. This measurement is got in the *Landmark Extraction* step. A wide variety of sensors can be used such as a laser range finders and cameras depending on both the environment and the robot. Whatever sensor is used, the feature or landmark which is to be used as measurement is extracted and it's relation to the pose of the robot is represented by a measurement model. This model forms the function h from equation (2.11). Differentiating this with respect to the robot pose we get a part of the H matrix.

For the rest of the Jacobian we need to differentiate it with respect to the rest of the state vector which contains some representation of the environment already observed by the robot. Also to successively calculate the difference $z_k - h(\hat{x}_k^-, 0)$ from equation (2.17), commonly called the *innovation*, we need to know which measurement corresponds to which feature in the observed environment. This is the essence of the *Data Association* step. A number of methods such as Euclidean or Mahalanobis [6] distance are commonly used.

Once we have the innovation, both the robot pose and the environment is updated simultaneously using equations (2.16) to (2.18).

CHAPTER III
Integrated Mobile Platform

3.1 Overview

The primary purpose of building the Integrated Mobile Platform is to test the various algorithms that are used in SLAM. For this we need to keep the following considerations in mind.

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

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

Environment sensing Has to have sensors to understand the environment.

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

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

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

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

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

3.2 Mechanical Design

3.3 Electrical Components

3.4 Control Diagram

3.5 Code Structure

CHAPTER IV

Encoder Based Odometry for State Estimate

4.1 Overview

In the platform previously described in chapter III, we have seen that two of it's wheel have encoders attached. These encoders give us a good 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 (2.10) in section 2.3. Since this robot has only 3 degrees of freedom, our state vector, $x \in \Re^3$ and is defined as $x = [x, y, \theta]^T$ where x and y give the position of the robot from an arbitrary fixed point in inertial frame of reference.

4.2 Motion model and the corresponding differentials

To estimate the motion model, we first convert left and right wheel movement into distances traveled by them 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. We separately consider the cases when the robot is estimated to be going straight or to be turning. We can know this by just looking at the left and right distances. If they are exactly the same, the robot is going straight and the motion model is given by equations (4.1).

If $r = l$:

$$\begin{bmatrix} \hat{x}^- \\ \hat{y}^- \\ \hat{\theta}^- \end{bmatrix}_k = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{k-1} + \begin{bmatrix} l.\cos(\hat{\theta}_{k-1}) \\ l.\sin(\hat{\theta}_{k-1}) \\ 0 \end{bmatrix} \tag{(4.1)}$$

If not, we use equations (4.2) and (4.3). Where R, α and w are as shown in figure 1

If $r \neq l$:

$$\alpha = \frac{r - l}{w} \qquad R = \frac{l}{\alpha} \tag{(4.2)}$$

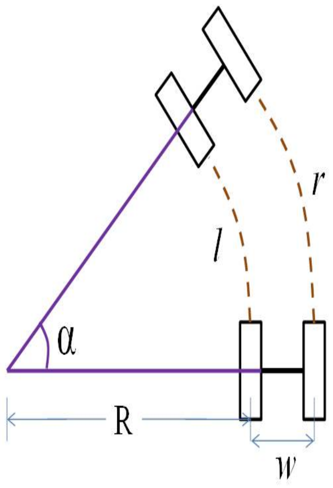


Figure 1: Motion model of Differential drive platform

$$\begin{bmatrix} \hat{x}^- \\ \hat{y}^- \\ \hat{\theta}^- \end{bmatrix}_k = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix}_{k-1} + \begin{bmatrix} \left(R + \frac{w}{2}\right)(\sin(\hat{\theta}_{k-1} + \alpha) - \sin(\hat{\theta}_{k-1})) \\ \left(R + \frac{w}{2}\right)(-\cos(\hat{\theta}_{k-1} + \alpha) - \cos(\hat{\theta}_{k-1})) \\ \alpha \end{bmatrix} \tag{(4.3)}$$

In general for ease of representation we will refer to this motion model only by f.

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k) \tag{(4.4)}$$

Once we have the motion model we need to find it's Jacobian with respect to the state. Since both the motion model f and the state have 3 dimensions the Jacobian will be a 3×3 matrix given by (4.5).

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} \tag{(4.5)}$$

Since the motion model is piecewise, we can calculate it's Jacobian also in two parts by differentiating the respective equations (4.1) and (4.3).

If $r = l$:

$$A = \begin{bmatrix} 1 & 0 & -l\sin\theta \\ 0 & 1 & -l\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{(4.6)}$$

If $r \neq l$

$$A = \begin{bmatrix} 1 & 0 & (R+\frac{w}{2})(\cos(\theta+\alpha)-\cos\theta) \\ 0 & 1 & (R+\frac{w}{2})(\sin(\theta+\alpha)-\sin\theta) \\ 0 & 0 & 1 \end{bmatrix} \tag{(4.7)}$$

Where R, w and α are according to equation (4.2) and figure 1

Once we have the Jacobian with respect to the state, we need to differentiate it with respect to the noise. Here we assume the process noise is essentially due to the noise in encoder measurement and it is additive in nature. Hence we can know how the process moves with noise by just differentiating it 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

$$W = \frac{\partial f}{\partial(u+w)} = \frac{\partial f}{\partial(u)} = \begin{bmatrix} \frac{\partial f_1}{\partial l} & \frac{\partial f_1}{\partial r} \\ \frac{\partial f_2}{\partial l} & \frac{\partial f_2}{\partial r} \\ \frac{\partial f_3}{\partial l} & \frac{\partial f_3}{\partial r} \end{bmatrix} \tag{(4.8)}$$

We then derive each of the terms separately in a piecewise manner.

If $r = l$:

$$\frac{\partial f_1}{\partial l} = \frac{1}{2}(\cos\theta + \frac{l}{w}\sin\theta) \tag{(4.9)a}$$
$$\frac{\partial f_2}{\partial l} = \frac{1}{2}(\sin\theta - \frac{l}{w}\cos\theta) \tag{(4.9)b}$$
$$\frac{\partial f_1}{\partial r} = \frac{1}{2}(-\frac{l}{w}\sin\theta + \cos\theta) \tag{(4.9)c}$$
$$\frac{\partial f_2}{\partial r} = \frac{1}{2}(\frac{l}{w}\cos\theta + \sin\theta) \tag{(4.9)d}$$
$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{(4.9)e}$$

If $r \neq l$:

$$\frac{\partial f_1}{\partial l} = \frac{wr}{(r-l)^2}(\sin\theta' - \sin\theta) - \frac{r+l}{2(r-l)}\cos\theta' \tag{(4.10)a}$$
$$\frac{\partial f_2}{\partial l} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{(4.10)b}$$
$$\frac{\partial f_1}{\partial r} = \frac{-wr}{(r-l)^2}(\sin\theta' - \sin\theta) + \frac{r+l}{2(r-l)}\cos\theta' \tag{(4.10)c}$$
$$\frac{\partial f_2}{\partial r} = \frac{wr}{(r-l)^2}(-\cos\theta' + \cos\theta) - \frac{r+l}{2(r-l)}\sin\theta' \tag{(4.10)d}$$
$$\frac{\partial f_3}{\partial l} = -\frac{1}{w} \quad \frac{\partial f_3}{\partial r} = \frac{1}{w} \tag{(4.10)e}$$

Where, $\theta' = \theta + \alpha$ and R, w and α are as per figure 1. Once we have the Jacobian, the only thing we need for the estimation according to equation (2.13) is the Process noise covariance Q. This has to contain some information about the amount the noise in each time step. Since we are assuming all noise to be only sensor measurement noise, we choose a diagonal matrix with the error in each encoder as the covariance as in equation

$$Q = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \qquad ((4.11))$$

4.3 Experimental results

Description of the arena and the run performed.

Images of the path ground truth and prediction.

We see that while it gives a good estimate of the path taken, it gradually deviates from the actual truth.

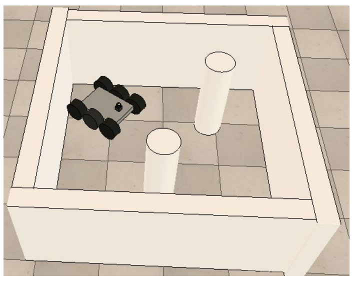
CHAPTER V
Feature Extraction with Point Features

5.1 Overview

To improve the estimate of the state and to get an idea of our environment, we need to understand the input of our exteroceptive sensors. As seen in chapter III, our robot is equipped with both a Camera and a LIDAR. In this chapter we try to use a simple algorithm based to detect fixed features in the environment. We then try to improve upon the algorithm by using our preexisting knowledge of the environment.

5.2 Feature Extraction Algorithm

Consider an extremely clean environment. One which has only fixed features and they are all away from the wall. And the entire environment is within the range of the LIDAR. In such an environment the distance readings gradually increase and decrease all along the walls except when they encounter and obstacle or a *feature*. This algorithm essentially looks for large jumps in the distance readings. One such arena is shown in figure 2. The gradual nature of the distance readings are more clearly seen when we plot the scan values as a function of the angle of the scan beam with respect to the robot as done in figure 3a.


(a) A simulated arena

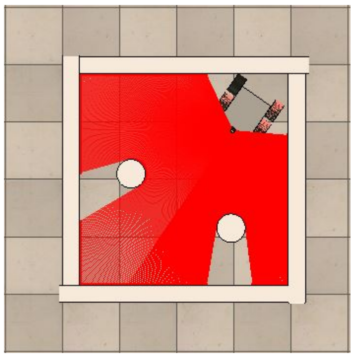
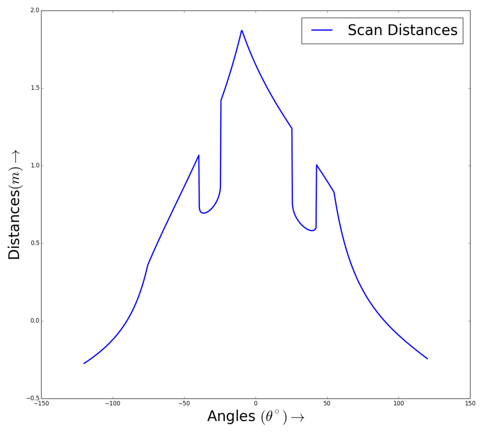
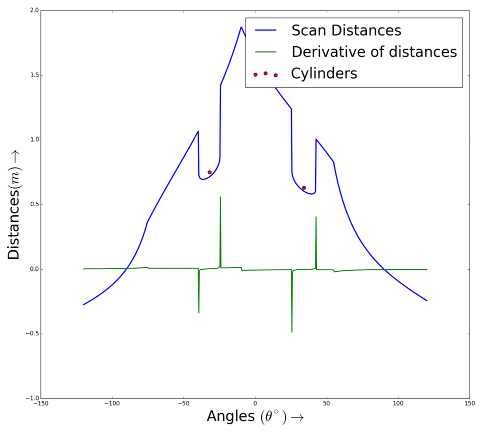

(b) Top view

Figure 2: Simulated Arena for LIDAR



(a) LIDAR scan



(b) Detected cylinders

Figure 3: Differential based LIDAR feature detection

For the robot to detect the large jumps we differentiate the distance with respect to angles. As seen in figure 3b this will give a specific pattern each time a cylinder is present. A condition can then be designed in the following way. Each time the derivative is larger than a fixed threshold and is a negative number, a cylinder's beginning is found, and when a large positive value is encountered, it's end. Finding the mid point of these two readings we find the location of the cylinder as in figure 3b.

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

To overcome this, based on preexisting knowledge of the nature of features a filter of sorts can be designed. A low threshold is chosen and a large number of prospective features are enumerated. All those that lack a minimum number of points on them are immediately eliminated. Next, based on the fact that we know the radius of the cylinders that are the landmarks, we can approximate the angular width of any feature detected. From the large set, only the ones within a range around this approximate angular width are retained. Then, by computing least squares fit we see if the features are indeed cylinders or segments of the wall. Only the features having a large enough curvature are retained. By this method we add a small amount of robustness to a simplistic algorithm of feature detection. The results of such a filter when applied with EKF can be seen in section 5.4.

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

5.3 Measurement Model and the corresponding Jacobian matrices

Once we have found the cylinder coordinates in the LIDAR data, we need to store it's position in the map. Since we know the estimate of the robot's position in the inertial frame, we can map the cylinder coordinates from robot to inertial frame using a homogeneous transformation as in equation (5.1). Where (x_w, y_w) are coordinates in world frame. (x_r, y_r, θ_r) are estimated pose of the robot and (x_1, y_1)

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_r) & -\sin(\theta_r) & x_r \\ \cos(\theta_r) & \sin(\theta_r) & y_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{(5.1)}$$

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

For the correction of the robot pose as explained in section 2.4, we need to find the measurement model of the point features. That is we need the laser reading or the distance r and angle α from the robot to the cylinder. For this we first find the coordinates of the cylinder in the robots frame of reference using the inverse of the homogeneous transformation that we use in equation (5.1). But the robot pose used now is the new position at this particular time step. Once we have (x_1, y_1) in the robot frame we can find the measurement h using equation (5.2). Same equations with the detected cylinder coordinates will give z allowing us to calculate the *innovation* for use in equation (2.17).

$$h = \begin{bmatrix} r \\ \alpha \end{bmatrix} = \begin{bmatrix} \sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2} \\ \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) - \theta_r \end{bmatrix} \tag{(5.2)}$$

Once we have the measurement model to find the Kalman Gain according to equation (2.16) we need the derivatives of it with respect to the state vector $x \in \Re^n$ which contains the robot pose as well as all the landmarks already existing in the map at that particular time step. Since we assume each landmark is independent of each other. most part of the Jacobian H contains zeros except for the part corresponding to the robot pose and if the measurement has been associated with an existing landmark, then it will depend on that landmark's position. Hence for the Jacobian H we differentiate h as per equation (5.3).

$$H = \begin{bmatrix} \frac{\partial r}{\partial x_r} & \frac{\partial r}{\partial y_r} & \frac{\partial r}{\partial \theta_r} & \cdots & \frac{\partial r}{\partial x_1} & \frac{\partial r}{\partial y_1} & \cdots \\ \frac{\partial \alpha}{\partial x_r} & \frac{\partial \alpha}{\partial y_r} & \frac{\partial \alpha}{\partial \theta_r} & \cdots & \frac{\partial \alpha}{\partial x_1} & \frac{\partial \alpha}{\partial y_1} & \cdots \end{bmatrix} \tag{(5.3)}$$

We derive each of the terms separately as per equation (5.4).

$$\frac{\partial r}{\partial x_r} = \frac{(x_r - x_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial x_r} = \frac{(y_1 - y_r)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{(5.4)a}$$

$$\frac{\partial r}{\partial y_r} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_r} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{(5.4)b}$$

$$\frac{\partial r}{\partial \theta_r} = 0 \qquad \frac{\partial \alpha}{\partial \theta_r} = -1 \tag{(5.4)c}$$

$$\frac{\partial r}{\partial x_1} = \frac{(x_r - x_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_1 - y_r)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{(5.4)d}$$

$$\frac{\partial r}{\partial x_1} = \frac{(y_r - y_1)}{\sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2}} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{(y_r - y_1)}{(y_1 - y_r)^2 + (x_1 - x_r)^2} \tag{(5.4)e}$$

The only other information we need 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 we assume all landmarks are uniformly affected by noise, we can assume V to be identity. R is the measurement noise. It is a diagonal matrix with one of the eigenvalues representing the error in distance measurement of the LIDAR and the other the angle measurement as in equation (5.5).

$$R = \begin{bmatrix} \sigma_r & 0 \\ 0 & \sigma_\alpha \end{bmatrix} \tag{(5.5)}$$

Using all this information we can correct our estimate of the robot pose while simultaneously correcting the position of the landmarks.

5.4 Experimental results

Description of the arena and the run performed.

Images of the path ground truth and correction.

We see that it estimates the path taken by the robot to a good extant as well as gives us an idea of the environment.

CHAPTER VI

RANSAC based linear feature Extraction

6.1 Overview

In indoor environments on of the most ubiquitous features are walls. There maybe other features and obstacles to avoid and plan a path around, but it is essential you first map the walls so that you know where you are. In this implementation we use the LIDAR described in chapter III to find walls in our environment. We do this using a RANSAC based algorithm, which is also robust to people and other obstacles in the environment []. Once we have the walls we need to derive the measurement model for how a wall like linear feature is used in EKF SLAM. It is obvious that the measurement model and it's derivatives will not be the same as in point features as the wall moves very differently as the robot moves. For example it is not possible to know the robot's motion from the previous motion model if the robot is moving along a wall. Hence we need different measurement models and derivatives.

6.2 Feature Extraction Algorithm

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

Algorithm 1 Multiple line fitting with RANSAC

```
while
    • there are still unassociated laser readings,
    • and the number of readings is larger than the consensus,
    • and we have done less than N trials.
do
    • Select a random laser data reading.
    • Randomly sample S data readings within D degrees of this laser data reading
    • Using these S samples and the original reading calculate a least squares best fit
      line.
    • Determine how many laser data readings lie within X distance from this best fit
      line
    • If the number of laser data readings on the line is above some consensus C do
      the following:
        – calculate new least squares best fit line based on all the laser readings de-
          termined to lie on the old best fit line.
        – Add this best fit line to the lines we have extracted.
        – Remove the number of readings lying on the line from the total set of unas-
          sociated readings.
end while
```

As it is seen in the algorithm, we only extract lines and not line segments. Hence all lines are assumed to be infinite. While this is not actually true in most environments it is a good approximation for relatively long line segments which model walls.

6.3 Measurement Model and the corresponding differentials

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

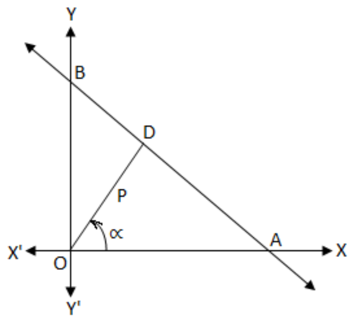


Figure 4: Normal form of a line

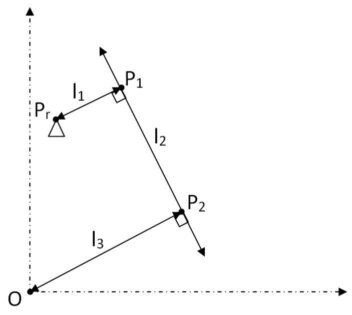


Figure 5: Line in robot frame and world frame.

point of intersection of the normal from the origin to the line is stored. For example, in figure 4 the line can be represented solely by the point D.

Once we have a line, we need to convert it to the inertial frame of reference as in section 5.3. Unlike point features we are not trying to find the same point in a different frame but the same line. In figure 5 we have point P_1 in robot frame and we need to find point P_2 in inertial given coordinates of the robot in inertial frame as P_r. For which we use Algorithm 2.

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

Algorithm 2 is also called the inverse measurement model as in subsequent time steps for the measurement model we do the exact opposite process. That is given a line using it's normal form in inertial frame, we need to get the *measurement* from the robot. That is, we need the perpendicular distance from the robot to the wall and the

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

```
1. Convert point $P_1$ from robot frame of reference to inertial frame as in section 5.3
2. Given points $P_r$ and $P_1$ in inertial frame, the equation of line $l_1$ is found using two
   point form of a line
3. Since $l1 \perp l2$ the slope of $l_2$ is the negative reciprocal of the slope of $l_1$.
4. Using the slope and the point $P_1$ the equation of line $l_2$ is found using slope-point
   form of a line.
5. The equation of line l3 is found similarly using slope of $l_1$ and point $O$.
6. Solving equations for $l_3$ and $l_2$ we get the coordinates of point $P_2$.
```

angle of the laser beam which would hit the wall normally. For this given coordinate pf point P_2 as (x_2, y_2) and the robot pose as (x_r, y_r, θ_r) we get P_1 using equation (6.1).

$$x_1 = x_2 - \frac{y_2(x_2y_r - y_2x_r)}{(x_2^2 + y_2^2)} \qquad y_1 = y_2 + \frac{x_2(x_2y_r - y_2x_r)}{(x_2^2 + y_2^2)} \tag{(6.1)}$$

$$r = \sqrt{(y_1 - y_r)^2 + (x_1 - x_r)^2} \qquad \alpha = \tan^{-1}\left(\frac{y_1 - y_r}{x_1 - x_r}\right) - \theta_r \tag{(6.2)}$$

Once we have P_1, we can find the perpendicular distance r and angle α using equation (6.2) giving measurement model h as per equation (6.3). This measurement can be used to calculate the *innovation* as per equation (2.17).

$$h = \begin{bmatrix} r \\ \alpha \end{bmatrix} \tag{(6.3)}$$

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 5.3 and is given by equation (5.3). We derive each of the terms separately as per equation (6.4) and (6.5).

$$\beta = \frac{x_r x_1 - x_1^2 - y_1^2 + y_r y_1}{x_1^2 + y_1^2} \tag{(6.4)a}$$

$$\frac{\partial r}{\partial x_r} = 2x_1\beta \quad \frac{\partial r}{\partial x_r} = 2y_1\beta \quad \frac{\partial r}{\partial \theta_r} = 0 \tag{(6.4)b}$$

$$\frac{\partial \alpha}{\partial x_r} = 0 \quad \frac{\partial \alpha}{\partial y_r} = 0 \quad \frac{\partial \alpha}{\partial \theta_r} = -1 \tag{(6.4)c}$$

$$\gamma = \frac{x_r x_1 - x_1^2 - y_1^2 + y_r y_1}{x_1^2 + y_1^2} \tag{(6.5)a}$$

$$\frac{\partial r}{\partial x_1} = -2x_1\gamma^2 - 2\gamma(2x_1 - x_r) \tag{(6.5)b}$$

$$\frac{\partial r}{\partial y_1} = -2y_1\gamma^2 - 2\gamma(2y_1 - y_r) \tag{(6.5)c}$$

$$\frac{\partial \alpha}{\partial x_1} = \frac{-y_1}{x_1^2 + y_1^2} \qquad \frac{\partial \alpha}{\partial y_1} = \frac{x_1}{x_1^2 + y_1^2} \tag{(6.5)d}$$

Having the Jacobian matrices we can use the same observer error given by equation (5.5) to correct the position estimate using equations (2.16) to (2.18).

6.4 Experimental results

Description of the arena and the run performed.

Images of the path ground truth and correction.

We see that it estimates the path taken by the robot to a good extant as well as gives us a better idea of the environment. As walls are an ubiquitous feature found in most indoor environments, this representation gives us more information. But it is very slow.

CHAPTER VII

Hough transform based linear feature extractor

7.1 Overview

In this implementation we use the LIDAR described in section ... as the input for the measurement z discussed in section 2.4. Both the measurement model and data association is derived based on there being linear features in the environment. Instead of a custom implementation which has a large number of tuning parameters we leverage the existing implementation of Hough transform on OpenCV and find the lines and represent them in the normal form. The measurement model is derived based on this form.

7.2 Feature Extraction Algorithm

More description of what is Hough transform. HOUGH feature extraction

7.3 Measurement Model and the corresponding differentials

The mathematical model for measurement.

7.4 Experimental results

Description of the arena and the run performed.
Images of the path ground truth and correction.
It gives good estimate of walls and is much faster.

CHAPTER VIII

Visual odometry

8.1 Overview

Traditionally a Camera is used as and exteroceptive sensor to get measurements of the environment. But it is interesting how it can also be used as a proprioceptive sensor using visual odometry []. By this, the number of sensors on a robot can be reduced saving cost and weight which are of extreme importance in certain cases.

8.2 Conceptual Description

Description of epipolar geometry The mathematical base for visual odometry

8.3 Implementation and integration with EKF SLAM

Visual odometry based motion model and Jacobian

8.4 Experimental results

Description of the arena and the run performed. Images of the path ground truth and prediction. Images of the path ground truth and correction.

Not as good. But good enough. Shown using corrected path

CHAPTER IX

Conclusion

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

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

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

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

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

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