Visual SLAM on Android

A thesis submitted in partial fulfillment
of the requirements for the degree of

Integrated Master of Technology

in

Mathematics & Computing

by

Ambrish Rawat

Entry No. 2010MT50583

Under the guidance of

Prof. Amitabha Tripathi

Department of Mathematics,
Indian Institute of Technology Delhi

July 2015

Abstract

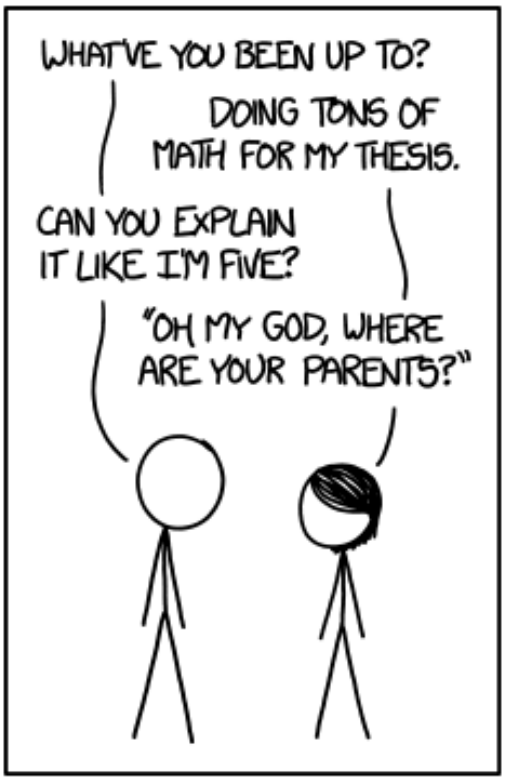
The Simultaneous Localisation and Mapping (SLAM) problem explores the possibility of incrementally building a map of an unknown environment while simultaneously localising the device or robot in the same environment. In this work the dynamics of an Extended Kalman Filter (EKF) based SLAM implementation on an Android device has been explored. Visual sensor namely, camera, has been utilised for landmark detection. Other sensors including accelerometer, magnetic field sensor and gravity sensor have been utilised for the providing measurement estimates of position and orientation of the device. An Android application was developed to demonstrate the working of EKF-based SLAM.

Acknowledgments

I would like to express my sincere gratitude to my project supervisor Prof. Amitabha Tripathi for his constant support during my project. His unconditioned support at every stage of the project helped me in exploring a new domain of applied mathematics. I would also like to thank Prof. Subhashis Banerjee for his words of guidance. It was his vast knowledge and expertise that led me to explore the problem of SLAM and its applications. It would be remiss of me not to thank my parents who inspite of all the long and often frustrating debugging sessions, always supported my work.

I am also thankful to the Department of Mathematics for providing me this great opportunity to work in the field of my choice.

Ambrish Rawat


WHAT'VE YOU BEEN UP TO?
DOING TONS OF MATH FOR MY THESIS.
CAN YOU EXPLAIN IT LIKE I'M FIVE?
"OH MY GOD, WHERE ARE YOUR PARENTS?"

Contents

List of Figures

Chapter 1

Introduction

Autonomous navigation in a previously unknown environment is a highly involved task and Simultaneous Localisation and Mapping (SLAM) provides one approach to solve this problem. SLAM asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map. SLAM is an essential tool for robots that are traveling in unknown environments where globally accurate position data (e.g. GPS) is not available. For instance, it is used for remote explorations like operating undersea on exploring the surface of another planet.

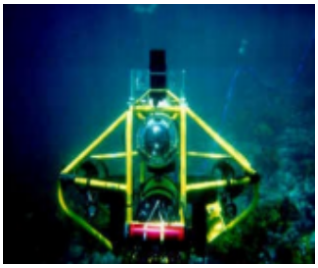


Figure 1.1: Undersea Navigation

Figure 1.2: Navigation on a Planet's Surface

Various statistical techniques are used to approximate the device and landmark positions. These include Kalman Filters, particle filters (FastSLAM

[9]) and scan matching of range data. They provide an estimation of the posterior probability function for the pose of the robot and for the parameters of the map. Different implementations of SLAM use different types of sensors, and the capabilities of these sensor types have been a major driver of new algorithms. Optical sensors used in SLAM range from single beam to 2D-(sweeping) laser rangefinders, sonar sensors and one or more 2D cameras. Since 2005, there has been intense research into Visual SLAM using primarily visual (camera) sensors, because of the increasing ubiquity of cameras in mobile devices. In this work, a possibility of utilising a smartphone and in particular Android devices, for robot navigation has been explored. A smartphone is equipped with a camera and sensors like a accelerometer, magnetic sensor and gyroscope. These can be utilised to provide the necessary data for a SLAM framework.

The motivation of this project was to learn about the various intricacies of implementing a SLAM algorithm on a mobile device. The project work comprises of developing a mobile app which harnesses the sensor data and an Extended Kalman Filter based SLAM implementation to generate a map for the environments. The application was developed This involved numerous steps, which have been detailed in the subsequent sections of this report.

1.1 Problem Description

Owing to the nature of the problem, simultaneous localisation and mapping is inherently a difficult task. This problem is difficult because of the following paradox: to move precisely, a mobile robot must have an accurate environment map; however, to build an accurate map, the mobile robot's sensing locations must be known precisely. In this way, simultaneous map building and localisation can be seen to present a question of 'which came first, the chicken or the egg?' (The map or the motion?). In an unknown environment, a device essentially performs two activities – it moves and it collects observations. SLAM attempts to recover a map of the environment and the path of the robot device from these controls and observations. The two-part survey of SLAM ([5] and [4]) provides a good description of the theoretical

basis of the problem along with a brief description of the different strategies used in the implementation of SLAM.

The challenges to the problem are introduced due to the noisy data obtained from the measurement devices and sensors. As the robot moves, its pose estimate is corrupted by motion noise. Similarly, the perceived locations of of landmarks is corrupted by both, the data from the sensors measurement noise and the error in the estimated pose of the robot. However, the error is estimated post of the robot have a systematic effect on the error in the map. This correlation between the error in the robot pose and the error in the map can be leveraged in factoring SLAM into smaller problems thereby, providing an efficient solution to the problem.

Figure 1.3 explains this correlation graphically. in this figure, the dashed line represents the path along which the robot moves, the circles represent the observed landmarks and the ellipses represent the uncertainty in estimates. It can be seen in figure 1.3 that as the uncertainty in the robot's pose estimate increases, so does the uncertainty in the landmark observations. However, when the robot completes a loop and revisits a previously observed landmark, the uncertainty in the robot's current pose is decreased along with the uncertainty in that landmark's position estimate. Moreover, the uncertainty in all other landmark positions is also decreased. Work by Smith and Cheesman [12] and [6] established a statistical basis for describing relationships between landmarks and manipulating geometric uncertainty

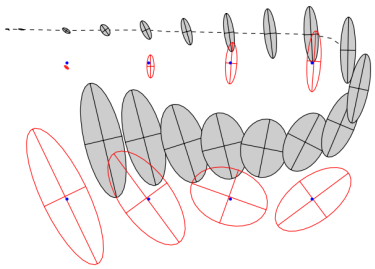


Figure 1.3: Before closing the loop

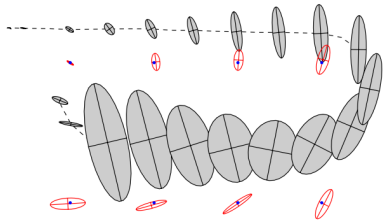


Figure 1.4: After closing the loop

1.2 Outline

- **Chapter 2** of the report details the mathematical foundation of Kalman Filters for discrete time linear systems as well as discrete time non-linear systems.
- **Chapter 3** details the various aspects involved in the implementation of SLAM on a device. This primarily includes pose tracking and landmark extraction. The strategies of pose tracking and landamark extraction that have been implemented in this project have also been explained in this chapter.
- **Chapter 4** describes the various variables used in the Android implementation of SLAM. This chapter gives details of how various variables of Extenden Kalman Filter extend to SLAM and how each is stored and manipulated in the SLAM algorithm. The description provides a stepping stone for a program implementation of SLAM.
- **Chapter 5** provides details of the experimental setup which includes the technical details of the application, specification of device along with type and placement of landmarks.

Chapter 2

Kalman Filter

This chapter describes the theoretical foundations required for implementation of Kalman FIlter based SLAM algorithm. Section 2.1 briefly discusses the mathematical foundation of discrete Kalman FIlters which is followed by Section 2.2 where the equations for Extended Kalman FIlter(EKF) have been explained.

2.1 Kalman Filter

Kalman Filters were first introduced in a famous 1960 paper by R. E. Kalman [7]. Owing to the advances in Digital Computing, Kalman Filter was extensively researched. One particular application that was benefited from this research was autonomous navigation. In a nutshell Kalman Filter can be summarised as an efficient computational tool consisting of a set of equations which attempt to estimate the state of a process. This is done in a manner that minimises the mean of the squared error. A brief introduction to Kalman Filters can be found in [13].One of the main advantages of using a Kalman Filter is that it can estimate the state of the process even when specifics of the modelled system are unknown.

The general problem that the Kalman Filter addresses is the process of a linear, discrete-time, finite-dimensional time-varying system. Here, the filter attempts to estimate the state, $x \in R_n$, via measurement $z \in R_m$, which is governed by the following process represented as a discrete-time linear stochastic difference equation,

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.1}$$

In this equation (2.1), the matrix A relates the current state with the state at the previous time step. u is the control input vector and matrix B relates

it to the current state x. The observation or measurement is related to the state via the following equation.

$$z_k = Hx_k + v_k \tag{2.2}$$

In this equation (2.2), the measurement z is relates to the state x via matrix H. Here, w_k and v_k are the process and measurement noise respectively. In the framework of Kalman Filters these random variables are assumed to have normal probability distributions. Moreover, they are independent of each other and white, i.e. with a mean zero and no correlation between its values at different times. The process and measurement noises can thus be described using the following formula, with Q and R as the resp

$$\begin{aligned} p(v) &\sim N(0, Q) \\ p(w) &\sim N(0, R) \end{aligned} \tag{2.3}$$

The Kalman Filter equations work as a feedback control loop, where the 'predictor' or 'time-update' equations predict a future state and the 'corrector' or 'measurement-update' equations correct the prediction. The equations are such that the mean of squared error is minimised. In deriving the equations for the Kalman filter, first a posteriori state estimate x_{ok} is computed as a linear combination of an a priori estimate x_{pk} and a weighted difference between an actual measurement z and a measurement prediction Hx_{pk} as shown in the following equation,

$$x_{ok} = x_{pk} + K(z_k - Hx_{pk}) \tag{2.4}$$

Here, K is the Kalman gain and $(z_k - Hx_{pk})$ is called the innovation. K is calculated such that the posteriori error covariance is minimised.

The specific time-update equations thus obtained for a discrete-time Kalman Filter are,

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} \\ P_k &= AP_{k-1}A^T + Q \end{aligned} \tag{2.5}$$

The measurement update for a discrete-time Kalman filter are,

$$\begin{aligned} K_k &= P_k H^T (HP_k H^T + R)^{-1} \\ x_k &= x_k + K(z_k - Hx_k) \\ P_k &= (1 - K_k H)P_k \end{aligned} \tag{2.6}$$

The term $(z_k - Hx)$ in the above equation, is called the innovation. Here, P is the covariance matrix of state variables.

2.2 Extended Kalman Filter

In the previous section, the general case of Kalman Filter was described which was used to estimate the state of a linear, discrete-time, finite-dimensional time-varying system. However, the more interesting and successful applications of Kalman filtering have been in the cases when the process to be estimated and (or) the measurement relationship to the process is non-linear. With this non-linear assumption, the state x of the non-linear stochastic process is governed by the equation 2.7 where the non-linear function f relates the state at current time to the state at previous time-step. Similarly, the measurement z_k is relates to the state x_k via a non-linear function h (equation 2.8).

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{2.7}$$

$$z_k = h(x_k, v_k) \tag{2.8}$$

As with the basic Kalman Filter, the minimisation of error results in the following set of time-update and measurement-update equations.

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}, 0) \\ P_k &= A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \end{aligned} \tag{2.9}$$

$$\begin{aligned} K_k &= P_k H_k^T (H_k P_k H_k^T + V_k R_{k-1} V_k^T)^{-1} \\ x_k &= x_k + K_k(z_k - h(x_k, 0) \\ P_k &= (1 - K_k H_k) P_k \end{aligned} \tag{2.10}$$

Here, A and W are Jacobians of partial derivatives of f with respect to x and w and respectively. Similarly, H and V are Jacobians of partial derivatives of h with respect to x and w and respectively.

Chapter 3

SLAM

The last chapter introduced Kalman Filters and Extended Kalman Filters from a mathematical point of view. This chapter lays out the details of SLAM process using Extended Kalman Filters. An elaborate description of how SLAM is implemented using Kalman Filters has been detailed in [11]. [10] also provides a good introduction to EKF SLAM and a comparative study between EKF SLAM and other SLAM implementations. A detailed discussion on SLAM implementation can be found in the PhD thesis [3].The emphasis of this chapter is on the practical implementation of SLAM and details the steps involved therein. First a description of Extended Kalman Filter based SLAM is discussed in Section 3.1. This is followed by the Section 3.2 and Section 3.3 which details the two major steps of its algorithmic implementation namely, Pose Tracking and Landmark Detection, respectively.

3.1 Extended Kalman Filter based SLAM

As mentioned previously an Extended Kalman Filter has a non-linear assumption for the process whose state is being estimated. In SLAM an Extended Kalman Filter is used to estimate the state (position) of the robot from odometer data and landmark observations. Along with robot pose, in case of SLAM the state also consists of map details. The map details comprises of the positions of landmarks. The measurement as explained in the previous section is used to correct the state estimates. In SLAM, the landmark observation provides these necessary measurement. This consists of measurement of range and bearing. Range defines the distance of the landmark form the device while the bearing defines the angle at which the landmark is observed. The details of landmark extraction have been explained in Section 3.3. The process of Extended Kalman Filter based SLAM can thus be summarised in the following three steps

- Use the odometer data to update the current estimate of state (time-update)
- Use reobservation of landmark to improve upon the estimated state (measurement-update)
- New landmarks are added to state as and when they are observed.

The non-linear functions f and h in the previous chapter are sometimes referred to as time and measurement models respectively. The measurement model defines how to compute an expected range and bearing of the measurements (observed landmark positions).

3.2 Pose Tracking

As mentioned in the previous section a rough initial estimate of position of the device is required at every stage. Modern day devices have various sensors deployed in them that can be utilised for this purpose. Almost every Android-powered handset and tablet has an accelerometer, and it uses about 10 times less power than the other motion sensors. However, raw computation of accelerometer-based position and velocity estimates is usually poor and unreliable. But they provide reasonably good starting point and when fed into a Kalman Filter result in good state estimation. Since the current framework demands and provides just that, accelerometer based computation of velocity and distance has been adopted in this work.

In general an accelerometer measures both the physical acceleration of the sensor and the contribution of normal forces that prevent the accelerometer from accelerating toward the centre of the Earth (i.e. the force exerted on the accelerometer by the table it is sitting on is actually measured by the sensor). In order to measure only the component of acceleration that is caused by physical acceleration, normal forces must be removed. Also, accelerometers uses the standard sensor coordinate system (figure). In order to convert the obtained values to world coordinate system, the body-frame is rotated to intertial-frame (world coordinates)

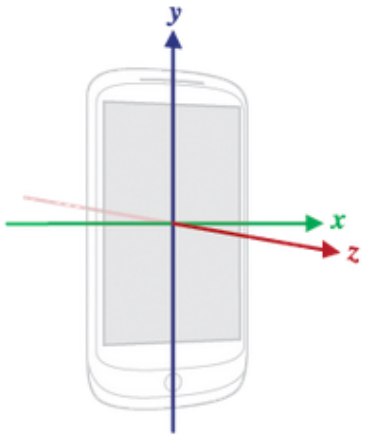


Figure 3.1: Device's Coordinate System

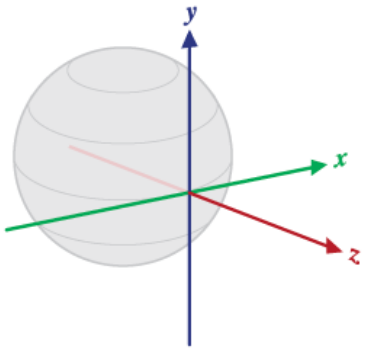


Figure 3.2: World Coordinate System

3.2.1 Estimating Distance

Once the measured inertial-frame acceleration (a) has been obtained, it can be integrated to obtain the inertial frame velocity v and position a values.

$$\begin{aligned} v &= \int a dt \\ x &= \int v dt \end{aligned} \tag{3.1}$$

In practice, however, approximate integration using data obtained at discrete time-intervals

$$\begin{aligned} v[i] &= \frac{1}{2}(a(i) + a(i-1))\Delta t \\ x &= \sum_{i=1}^{n} \frac{1}{2}(v(i) + v(i-1))\Delta t \end{aligned} \tag{3.2}$$

One of the major source of error in this computation arises from inaccurate rotation from body-frame to intertial-frame ([8]). This gets translates as a constant in the computation of acceleration which on double integration yields a parabolic error in the computation of the distance. This is why the data from this computation is only used as an estimate for pose tracking and fed into a Kalman Filter.

The integration mentioned above is performed in two steps

- Since the data obtained from accelerometer is noisy, it is first passed through a low pass filter. In this work, a lowpass Butterworth filter was used.
- After the smoothening the data was integrated over the region where there is high *enough* acceleration difference for the longest time. The longest time where local standard deviation was greater than an experimentally found threshold was considered for integration.

A sensor delay of 20,000 microseconds (SENSOR_DELAY_GAME) was chosen for this work. This value was chosen such that integration computation was easily managed with a smooth video rendering in the developed application.

3.2.2 Estimating Orientation

Orientation of the device is one of the state variables in the state vector. Along with the x and y coordinate of the device, orientation of the device is also estimated. As with the distance, a rough estimate of orientation is

also needed at every stage. This can be found out using the sensor data from gravity sensor and magnetic sensor present on the device. Most of the modern day devices are equipped with these sensors a simple geometric computation from the values of the orientation matrix can give the required angle. Using gravity and magnetic field sensors, an orientation data for the following three dimensions can be obtained. A simple geometric manipulations of which gives the required angle of orientation.

- Azimuth (degrees of rotation around the z axis). This is the angle between magnetic north and the device's y axis. For example, if the device's y axis is aligned with magnetic north this value is 0, and if the device's y axis is pointing south this value is 180. Likewise, when the y axis is pointing east this value is 90 and when it is pointing west this value is 270.
- Pitch (degrees of rotation around the x axis). This value is positive when the positive z axis rotates toward the positive y axis, and it is negative when the positive z axis rotates toward the negative y axis. The range of values is 180 degrees to -180 degrees.
- Roll (degrees of rotation around the y axis). This value is positive when the positive z axis rotates toward the positive x axis, and it is negative when the positive z axis rotates toward the negative x axis. The range of values is 90 degrees to -90 degrees.

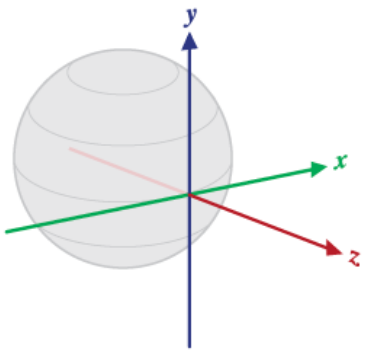


Figure 3.3: World Coordinate System

3.3 Landmark Detection

Landmark detection is a key step in the SLAM process. The innovation term as discussed in the previous section is is used in updating the state vector that the KalmanFilter is trying to estimate. It is essentially the difference between the estimated device position and the current measurement of the position. In the second step the uncertainty of each observed landmark is also updated to reflect recent changes. An example could be if the uncertainty of the current landmark position is very little. Re-observing a landmark from this position with low uncertainty will increase the landmark certainty, i.e. the variance of the landmark with respect to the current position of the robot.

In the current framework for SLAM, landmark detection was achieved using image *matchTemplate* method provided in OpenCV library. This method matches a template image with the source image by sliding the template image over each pixel of the source image. At each step a metric is calculated which so it represents how "good" or"bad" the match at that location was. For each location of template image, T, over the source image, I, the metric is stored in the result matrix (R). The metric used in this work was [**?**],

$$R(x,y) = \sum_{x',y'} (T'(x',y')I(x+x',y+x')) \tag{3.3}$$

where T' and I' are given by,

$$\begin{aligned} T'(x',y') &= T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'') \\ I'(x+x',y+y') &= I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'') \end{aligned} \tag{3.4}$$

Figure 3.4: Template matching – sliding window

Chapter 4

Implementation on Android

This chapter details the various aspects with respect to the implementation of Extended Kalman Filter based SLAM on an Android device. Section 4.1 describes the various state variables that are stored as part of the SLAM algorithm. This is followed by Section 4.2 which describes the various update rules of the algorithm.

4.1 Variables

Following is the set of matrices that need to be stored for SLAM processing

- X - The state vector consists of x-y coordinate and orientation for the device along with the x-y coordinates for the landmarks. These x-y coordinates are in world coordinate system.
- P - This is the covariance matrix of the state vectors. It contains the covariance of the device position, the covariance on the landmark position, the covariance between device and landmark and finally it contains the covariance between the landmark positions.
- K - This is the Kalman Gain matrix and is computed to find out how much we will trust the observed landmarks and as such how much we want to gain from the new knowledge they provide
- H - The measurement model, h is defined as equation 4.1, and and its Jacobian is given by the equation 4.2. Here, x and y represent the current estimates of the device's coordinates, λ_x and λ_y are the estimates for the position of landmark under consideration.

$$\begin{bmatrix} r \\ b \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)^2} \\ \tan^{-1}\left(\frac{\lambda_y - y}{\lambda_x - x}\right) \end{bmatrix} \tag{4.1}$$

$$\begin{bmatrix} \frac{x-\lambda_x}{r} & \frac{y-\lambda_y}{r} & -1 \\ \frac{\lambda_y-y}{r^2} & \frac{\lambda_x-x}{r^2} & 0 \end{bmatrix} \tag{4.2}$$

The matrix used as H consists of some additional values to the above mentioned Jacobian are required. Consider a case when there landmarks have been observed till the current stage and the second landmark has been re-observed in the current step. The first three columns are dependence on the x-y coordinate and theta of the device, while the following columns are for the dependencies on x-y coordinates of landmarks.

$$\begin{bmatrix} \frac{x-\lambda_x}{r} & \frac{y-\lambda_y}{r} & -1 & 0 & 0 & \frac{\lambda_x-x}{r} & \frac{\lambda_y-y}{r} & 0 & 0 \\ \frac{\lambda_y-y}{r^2} & \frac{\lambda_x-x}{r^2} & 0 & 0 & 0 & \frac{y-\lambda_y}{r^2} & \frac{x-\lambda_x}{r^2} & 0 & 0 \end{bmatrix} \tag{4.3}$$

- A - This is the Jacobian of the prediction model, f. The prediction model is defined as equation f 4.4, and A is given by the equation 4.8. Here, , $\Delta x and \Delta y$ are measured directly using the sensor data of the device.

$$\begin{bmatrix} x + \Delta x \\ y + \Delta y \\ \theta + \Delta\theta \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

- Q - This represents the covariance matrix of the process noise and is given by the following matrix. Here, c is the measure of the error in odometer data. This needs to be set according the measurement device. In this framework a value of 0.01 has been used for c.

$$\begin{bmatrix} c\Delta x^2 & c\Delta x\Delta y & c\Delta x\Delta y \\ c\Delta y\Delta x & c\Delta y^2 & c\Delta y\Delta t \\ c\Delta t\Delta x & c\Delta t\Delta y & c\Delta\theta^2 \end{bmatrix} \tag{4.6}$$

$$\Delta t = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.7}$$

- R - This represents the covariance matrix of the measurement noise and is given by the following equation. The values for c and b are so chosen that they reflect the error in measurement of range and bearing of the landmarks detected. For this work, the value of c was chosen as 0.01 and the value of b was chosen as 1 (bearing was measured in degrees).

$$\begin{bmatrix} rc & 0 \\ 0 & bd \end{bmatrix} \tag{4.8}$$

- J_{xr} and J_z - The EKF based algorithm for SLAM differs from the regular SLAM in one aspect which is - addition of new landmark. The step of addition of new landmarks requires these two matrices J_{xr} and J_z. Here, J_{xr} (equation 4.9) is the Jacobian of the prediction model for x and y coordinate of landmark with respect to the x, y and θ of the device and J_z (equation 4.10) is the Jacobian of x and y coordinates of landmark with respect to range and bearing measurements.

$$J_{xr} = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \end{bmatrix} \tag{4.9}$$

$$J_z = \begin{bmatrix} \cos(\theta + \Delta\theta) & -\Delta t \sin(\theta + \Delta\theta) \\ \sin(\theta + \Delta\theta) & \Delta t \cos(\theta + \Delta\theta) \end{bmatrix} \tag{4.10}$$

4.2 Algorithm

As mentioned in the previous chapter, there are 3 steps involved in the algorithm of SLAM which are discussed in the following subsections.

4.2.1 Time-update

In this step, sensor data is used update the current estimate of state as per the prediction mode f. These distances are calculated by integrating the linear accelerator data as described in section. Apart from the updating the first three elements of the state vector, X, according to the prediction model, A, P and Q re also updated to reflect these changes. The following set of equations summarise this,

$$X'_{rr} = X + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} \tag{4.11}$$

$$A' = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \\ 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$Q' = \begin{bmatrix} c\Delta x^2 & c\Delta x\Delta y & c\Delta x\Delta y \\ c\Delta y\Delta x & c\Delta y^2 & c\Delta y\Delta t \\ c\Delta t\Delta x & c\Delta t\Delta y & c\Delta\theta^2 \end{bmatrix} \tag{4.13}$$

$$P'_{r,r} = A'P_{r,r}A' + Q' \tag{4.14}$$

$$P'_{r,i} = A'P_{r,i} \tag{4.15}$$

Here, P'_{rr} represents the top left 3x3 sub-matrix of P. Similarly, P_{ri} represents the top three rows of the covariance matrix P.

4.2.2 Measurement-update

Position estimated using sensor data is fed to Kalman Filter in order to compensate for the error in measurement. This is achieved using landmark observations. Using the landmark position observation and landmark posi-

tion estimation it is possible to calculate and correct the position estimate of the device. This is what is achieved in this step.

The matrix H is updates as discussed by equation 4.3. The following set of equations summarise this mathematically,

$$K = PH'^{T}(H'PH'^{T} + VRV^{T}) \tag{4.16}$$

$$X' = X + K(z - h) \tag{4.17}$$

For this step is is crucial that a system for landmark extraction and data association for landmarks is in place. In the framework for this work, landmark extraction is achieved using template matching. It was experimentally observed that the landmarks were detected at a range of 0.43 meters from the device and owing to the *matchTemplate* function they were observed when the bearing or the angle made by the landmark from the device was close to 0 degrees. This resulted in $z = (0.43\ 0)^T$)

4.2.3 New Landmark Update

Measurement update for this case are slightly different from re-observation update. In this step, the state variable estimates aren't corrected because the landmark is observed for the first time. The following set of operations are performed in this step,

$$X' = \begin{bmatrix} X x_{N+1} y_{N+1} \end{bmatrix} \tag{4.18}$$

$$P^{N+1,N+1} = J_{xr} P J_{x,r}^{T} + J_z R J_z^{T} \tag{4.19}$$

$$P^{r,N+1} = P^{r,r} J_{xr}^{T} \tag{4.20}$$

$$P^{N+1,r} = (P^{r,N+1})^{T} \tag{4.21}$$

$$P^{N+1,i} = J_{xr}(P^{r,i})^{T} \tag{4.22}$$

$$P^{i,N+1} = (P^{N+1,i})^T \tag{4.23}$$

Two new elements are added to X, namely, the x and y coordinates, (x_{N+1}, y_{N+1}), of the new landmarks detected at current location. These are calculated using the current location of the device as measured during the odometer update and the approximation that the landmarks are detected at a distance of 0.43 m and at an angle of 0 degrees from the device.

Chapter 5

Android Application

5.1 Experimental Setup

As part of the project, an Android application was developed to demonstrate the working of visual SLAM. The application **SLAMActivity** was developed using Android SDK 24.0.2 and the experiments were conducted using a Google Nexus 7 (2013) tablet. The OS version installed on the device was Android 5.1. For the functioning of the developed application OpenCV Manager 2.18 was also installed on the device. The application was coded in Java and following libraries were used as part of the implementation,

- OpenCV SDK was used for the image processing required for landmark detection ([2]).
- Efficient Java Matrix Library (EJML) was manipulating dense matrices involved in SLAM ([1])

QR codes were used as landmarks for this implementation. Figure 1.1 shows one such landmark. Different copies of this were placed at different position inside a room as shown in figure 1.2 . The application was tested with different maps where the number of landmarks ranging from 3 to 5. In the current setup a person physically moved around the room while the device was running the application. The device stored the values of its position along with the landmark position at every step of the algorithm. A map was thereafter generated using these stored values.

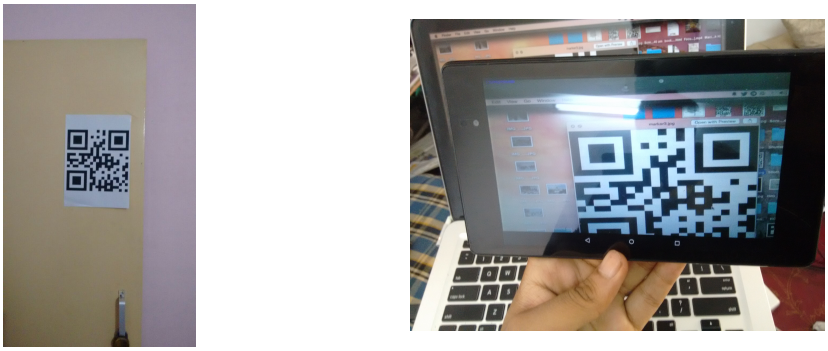


Figure 5.1: QR code & Google Nexus 7 (2013)

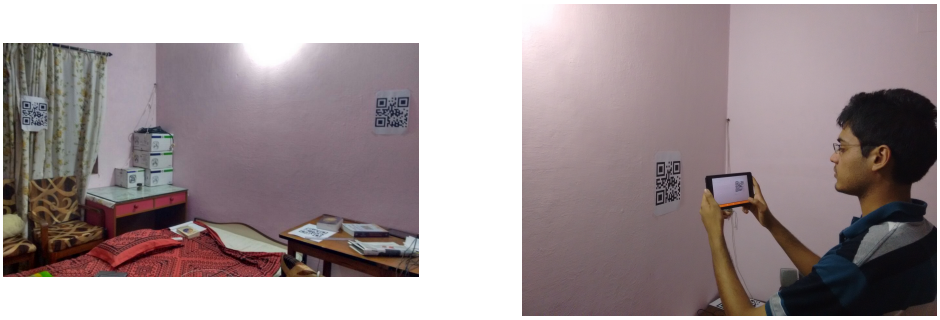


Figure 5.2: Landmarks placed in a room

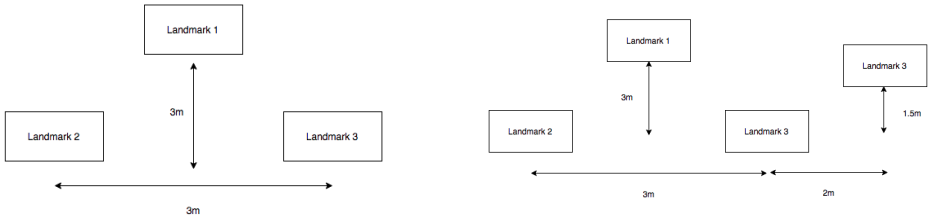


Figure 5.3: Sample Scenarios

5.2 Conclusion

It was observed that the generated values of device locations and landmark positions weren't in agreement with the actual values and they often deviated from them by considerable numbers. There are many reasons which could have led to this.

- Device sensors like accelerometers are known to provide highly noisy data. An improper smoothening and integration could have led to erroneous odometery update
- A simplistic landmark detection model could also have influenced the working of the program. In most of the visual SLAM based systems, stereo cameras provide the necessary feature detection and matching for landmarks. Corner detectors are often used as landmarks in these cases which provide a more extensive landmark framework.
- Since there were too few landmarks in this framework the number of SLAM iterations were also less. This could be another source for the deviation observed in the results.

Bibliography

[1] Efficient java matrix library. *ejml.org*.

[2] Opencv documentation. *docs.opencv.org*.

[3] Tim Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Citeseer, 2002.

[4] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[5] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[6] Hugh F Durrant-Whyte. Uncertain geometry in robotics. *Robotics and Automation, IEEE Journal of*, 4(1):23–31, 1988.

[7] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[8] Paul Lawitzki. Android sensor fusion tutorial, 2013.

[9] Michael Montemerlo and Sebastian Thrun. Fastslam 2.0. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pages 63–90, 2007.

[10] Megan R Naminski. An analysis of simultaneous localization and mapping (slam) algorithms. 2013.

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

[12] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[13] Greg Welch and Gary Bishop. An introduction to kalman filters. 2001.