A SEMINAR REPORT

ON

"POINT CLOUD DATA AND SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)"

B.TECH- IV (ELECTRONICS & COMMUNICATION)

SUBMITTED BY:

SHUBHAM GARG

(Roll No.: U11EC069)

GUIDED BY:

Dr. J.N. SARVAIYA

ECED, SVNIT

DEPARTMENT OF ELECTRONICS ENGINEERING

Year: 2014-15

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY (SVNIT)
SURAT-395007

Sardar Vallabhbhai National Institute of Technology, Surat

ACKNOWLEDGEMENT

The special thanks go to my helpful supervisor, **Dr. Jignesh N. Sarvaiya (Associate Professor at Electronics & Communication Engineering Department, SVNIT)**. The supervision and support that he gave truly helped in progression and completion of the seminar. I offer him my sincere gratitude.

I would also like to thank the faculty members **of Electronics Department, SVNIT** without whom this work would have been a distant reality. I also extend my heartfelt thanks to my family for the support and friends for their help and wishes for the successful completion of this seminar.

Last but not the least I would like to thank lab assistant who allowed me to use the Desktop Computer to accumulate the resources and use the lab property in case of emergency.

SHUBHAM GARG
Signature:

ABSTRACT

Navigating autonomously in a domestic environment is a problem that has attracted a great deal of interest in mobile robotics. A robotic system that operates in ordinary furnished rooms without the need of an engineered environment has many different applications such as service, cleaning and surveillance tasks or simply entertainment. Robotic systems that use artificial landmarks or pre-stored maps of the environment are available today. However, these systems are not very flexible. The user must in fact supply a map of the environment, which can be interpreted by the system.

This report deals with the problem of Simultaneous Localization and Mapping (SLAM). The mobile robot builds a map of an unexplored environment while simultaneously using this map to localize itself. The feature based approach is used in the report utilizes the Extended Kalman Filter (EKF) to estimate the pose of the robot and the location of the features. This approach is referred to as stochastic mapping. Point cloud data of the environment are robustly extracted from laser data using triangulation techniques. Here, in this report we have discussed the process of the EKF SLAM and the role of different hardware, sensor component involved in it. Graphical user interface is developed on the Matlab to create a random environment with various obstacles and landmarks and simulate the path followed by the robot. Navigation Error, memory complexity and accuracy of the developed map can be seen at the end of the Simulation.

Sardar Vallabhbhai National Institute of Technology

Surat-395007, Gujarat, India

ELECTRONICS ENGINEERING DEPARTMENT

CERTIFICATE

This is to certify that candidate **Mr.Shubham Garg** bearing **Roll No: U11EC069** of **B.TECH IV, 7TH Semester** has successfully and satisfactorily presented seminar & submitted the Report on the topic entitled "**Point Cloud Data and SLAM**" for the partial fulfilment of the degree of Bachelor of Technology (B.Tech) in **ELECTRONICS AND COMMUNICATION** in **Nov. 2014.**

Dr. Jignesh N. Sarvaiya	**Dr. U.D Dalal**
Seminar Guide	Head of Department
Associate Professor	Associate Professor

SEMINAR EXAMINERS

Name	**Signature with date**
1. Prof._____	_____
2. Prof._____	_____
3. Prof._____	_____

Departmental Seal

CONTENTS

List of Figures

CHAPTER 1: INTRODUCTION

Navigation in a realistic environment is a fundamental requirement for obtaining an autonomous mobile robot. The navigation problem according, to Leonard and Durrandt-Whyte (developers of (SLAM) Simultaneous Localization and Mapping) can be addressed as answering the questions: "Where am I?", "Where do I want to go?" and "How do I get there?" In order to give an answer to any of these questions, the platform needs to know what coordinate system and what environment these questions refer to. A coordinate system and the set of information about the location of features in the environment is called a map.

Implementation of navigation system that uses artificial landmarks or prior known maps of the environment, and accurate sensor systems to get precise measurements of the landmarks or map features, is straightforward for today's robots. Similarly, the task of building a map of the environment given the exact position of the robot is largely a solved problem. However, it is much harder to solve the complete problem simultaneously, enabling a mobile robot to build a map of an unexplored environment while simultaneously using this map to localize itself. The problem is known as Simultaneous Localization and Mapping (SLAM). So, SLAM is the process of building map of that environment such that it can easily clear all obstacles such as doors, stairs etc and also keep the track of its current location.

1.1 Point cloud data

A point cloud data is a set of data points in some coordinate system. In a three-dimensional coordinate system, these points are usually defined by *X*, *Y*, and *Z* coordinates, and are intended to represent the external surface of an object. There are many techniques for converting a point cloud to a 3D surface. Some approaches, like Delaunay triangulation, alpha shapes, and ball pivoting, build a network of triangles over the existing vertices of the point cloud, while other approaches convert the point cloud into a volumetric distance field and reconstruct the implicit surface so defined through a marching cubes algorithm[1].

1.2 Challenges faced in SLAM Problem

SLAM can be thought of as a Chicken or Egg problem. An unbiased map is needed for localization while an accurate pose estimate is needed to build that map. This is the starting condition for iterative mathematical solution strategies. Answering two characteristic equations is not as straightforward as it seems to be due to inherent uncertainties in the robot's relative movement from its various sensors. Generally, due to the budget of noise in a technical environment, SLAM is not served with just compact solutions, but with a bunch of physical concepts contributing to results.

If at the next iteration of map, building the measured distance and direction travelled has a budget of inaccuracies, driven by limited inherent precision of sensors and additional ambient noise, and then any features being added to the map will contain corresponding errors. Over time and motion, locating and mapping errors build cumulatively, grossly distorting the map and therefore the robot's ability to determine its actual location and heading with sufficient accuracy.

There are various techniques to compensate for errors, such as recognizing features that it has come across previously (i.e., data association or loop closure detection), and re-skewing recent parts of the map to make sure the two instances of that feature become one. Statistical techniques used in SLAM include Kalman filters, particle filters 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.

1.3 Types of SLAM

From a probabilistic perspective, there are two main forms of the SLAM problem, which are both of equal practical importance. One is known as the online SLAM problem. It involves estimating the posterior over the momentary pose along with the map:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \tag{1.1}$$

Here x_t is the pose at time t, m is the map, $z_{1:t}$ and $u_{1:t}$ are the measurements and controls respectively. This problem is called the online SLAM problem since it only involves the estimation of variables that persist at time t. The second SLAM problem is called the full SLAM problem. In full SLAM, we seek to calculate a posterior over the entire path $x_{1:t}$ along with the map, instead of just the current pose x_t:

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \tag{1.2}$$

1.4 Literature Survey

Here we will make a comparison of various simultaneous localization and mapping (SLAM) algorithms that have been proposed in literature. The performances of Extended Kalman Filter (EKF) SLAM, Fast SLAM, Distributed Particle (DP) SLAM algorithms are compared in terms of accuracy of state estimations for localization of a robot and mapping of its environment.

In Fast SLAM, a set of particles is used to keep track of the position of the helicopter and the positions and uncertainties of the landmarks. Each particle contains a guess on the position of the helicopter, and the position and uncertainty for each landmark it has observed so far [2]. Each time step, the helicopter moves in reaction to the inputs it is given. The different particles of Fast SLAM sample the possible locations where the helicopter might have moved, given the inputs. This is done by applying the dynamic model to the position of the helicopter, and having each particle sample from the probability distribution function of the resulting helicopter position after motion. The only information available to the helicopter is the distance at which the laser hit something in any given direction. However, since Fast SLAM is based on keeping track of landmark locations, an algorithm must decide which landmark the laser has hit at each time step. Is each observation associated with a landmark we have seen before or a new one, and if it seen before, then which one? This is called data association which we will be discussed in Chapter 3. The algorithm computes the probability that the current observation is of which landmark. This is done by computing the distance from the landmark to the observed obstacle, and using the uncertainty on the landmark position to calculate how likely it is that this distance could have been obtained by observing the landmark. The algorithm also computes

the probability that the observed obstacle is a new landmark, which is essentially the probability that the observation did not belong to the most likely landmark. As shown in Fig 1.1, the observed obstacle is very far from L2, so it is unlikely to be that obstacle. On the other hand, the observed obstacle is close to L1, which has a pretty high uncertainty. Thus, it is likely that the observed obstacle is L1 [3].

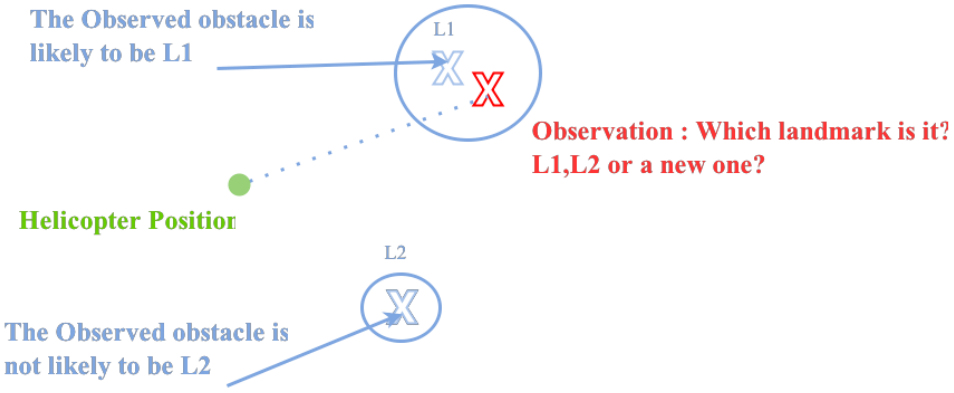

Fig 1.1 Data Association

For each particle, the weight for a landmark is decided which proportional to the probability that of observed Landmark.

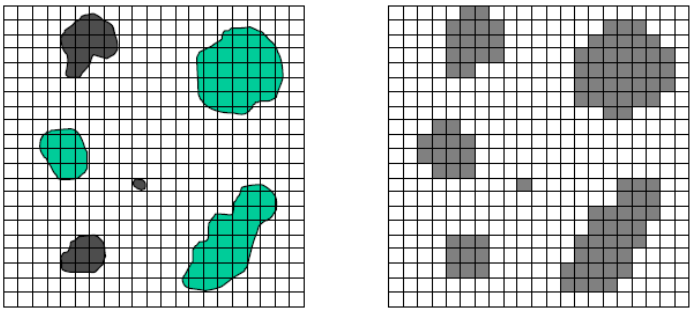

Fig 1.2 Overhead view of world with objects and its corresponding occupancy map representation [9]

The idealized representation of landmarks as single points in space does not always correspond well to the reality of three-dimensional space where objects have a non-zero size and may appear different from different angles. To avoid these problems, some SLAM algorithms (including DP-SLAM) [4] use an occupancy map to represent the environment rather than landmarks. An occupancy map is simply a discretization of space into a regular array. Each point in the array can then be marked as either "empty" or "occupied." Figure 1.2 shows an example 2-D world and an occupancy map representation of it. By using an occupancy map, it is no longer

necessary to find specific landmarks. The basic structure of the DP-SLAM algorithm is the same as a standard particle filter Fast SLAM algorithm. The difference lies in the way that DP-SLAM represents the state of the world. Rather than using a Kalman Filter on landmark positions, DP-SLAM uses probabilistic occupancy maps.

Standard Kalman filter assumes linear state transitions and linear measurement transitions with added Gaussian noise. The extended Kalman filter (EKF) overcomes the linearity assumption by linearizing the nonlinear state of robot. The Extended Kalman Filter (EKF) can be viewed as a variant of a Bayesian Filter; EKFs provide a recursive estimate of the state of a dynamic system, or more precise, solve an unobservable, nonlinear estimation problem. Roughly speaking, the state x_t of a system at time *t* can be considered a random variable where the uncertainty about this state is represented by a probability distribution. EKF uses a prediction and an update step. The prediction step calculates the probability of the current state x_t, while the measurement calculates the position for the current time step which is not yet available. A fundamental advantage of Fast SLAM over EKF based approaches to the SLAM problem is that the EKF suffers from a $O(K^2)$ complexity where *K* being the number of landmarks. In contrast, Fast SLAM has an $O(M \log K)$ complexity with *M* denotes the number of particles [5], [6].

Comparison of different SLAM techniques on various features:

- Speed: Fast SLAM is fairly fast, particularly as compared with DP-SLAM and EKF-SLAM.

- Memory requirements: Fast SLAM remembers only the helicopter's and landmark's positions. Thus, if the number of landmarks is not excessive, the memory requirement is low. Also, no additional memory is required to increase the resolution as compared with DPSLAM [8].

- Extension to 3-D: Fast SLAM extends easily to 3-D, with only incremental changes in computational requirements and time as compared with other SLAM techniques [9].

- Accuracy: The accuracy obtained was slightly disappointing. Partly this is due to the large size of the landmarks and the tendency of large landmarks to appear to move along with the helicopter. We can get high Accuracy with EKF-SLAM as compared with other two SLAM techniques [12].

- Robustness: Two points make Fast SLAM non-robust:
  i. The data association problem is not accurate. Moreover, a failure in the data association has drastic effect on the Kalman Filter.
  ii. The shape of the obstacles is not taken into account, since the Kalman Filter only uses points. Thus, when a rock is seen from one viewpoint or another, the landmark's position is moving as well, whereas a landmark should not move. This type of problem is not there in EKF-SLAM [13].
- Incomplete map: Unlike in DP-SLAM, the algorithm does not produce an actual map of the landscape, only a map of the landmarks. Some post processing (such as appending scans to the final calculated poses) has to be performed before getting a map of the actual landscape. But, EKF-SLAM provides the complete map of the environment [10].

1.5 Motivation

From this comparison, we have learned about the main challenges of SLAM, as well as the strengths and weaknesses of Fast SLAM, DP-SLAM and EKF-SLAM. EKF-SLAM was shown to provide a very accurate map of the world, even with a low number of particles. It is very robust, particularly when dealing with big objects, which are seen differently from different viewpoints. On the other hand, Fast SLAM was less accurate but very fast, needed much less memory, and was easily extended to 3D. However, it requires a problematic pre-processing of the sensor information, since it treats the world as a set of point landmarks. Also, the shape of the landmarks cannot be taken into account, making the result even less accurate in chaotic worlds. In a very complicated world, when memory and speed are not main concerns, one should use DP-SLAM. On the other hand, Fast SLAM provides a cheap algorithm in terms of memory and speed, but can have less accurate results. EKF-SLAM will be

our center of our attraction in further chapters since it maintains the balance between various aspects such as the accuracy of Map, speed, memory and complexity.

1.6 Report Organization

Chapter 1 deals with introduction of SLAM and Point Cloud Data. We have also discussed the challenges faced while performing the SLAM followed by Literature Survey on various SLAM Algorithms.

Chapter 2 & 3 introduces to the hardware requirement for the SLAM and briefly explains the process of EKF-SLAM. This chapter also deals with the mathematics involved in EFK-SLAM. The algorithms involved in the process are also explained briefly.

Chapter 4 provides the details of GUI developed for the Simulation of the EKF-SLAM followed by the Simulation results.

Chapter 5 explains the application of SLAM in various fields like: Space exploration, Underwater Robotics, indoor navigation (GPS denied environment) and Natural Disaster Management.

CHAPTER 2: Overview of SLAM Process

2.1 Hardware requirement of SLAM

To do SLAM there is the need for a mobile robot and a range measurement device. Most mobile robots for indoor applications have wheels. The movement of the robot, is usually measured through cheap sensors such as optical encoders mounted on the wheels and shaft to measure the number of rotations. By knowing a set of parameters, such as wheel radii and distance between the wheels, it is possible to compute the robot movement based on the information given by the encoders. This kind of information is referred to as odometry. The main disadvantage of odometry information is that, it is subjected to drift. This drift comes from two kinds of errors:

i. Systematic errors which are mainly due to imperfection in the structure of the robot, and

ii. Non-systematic errors are due to external causes such as wheel slippage when the robot is moving over a carpet, human intervention and so on. Non-Systematic errors are harder to deal as compared to systematic errors, since their appearance is of more random in nature. From a localization perspective, odometry drift is highly undesirable and may prevent an autonomous robot from accomplishing its task. The use of more advanced internal sensors such as gyros or accelerometers can only help to reduce the odometry drift, but not eliminate it.

2.1.1 Laser Scanner

The range measurement device used is usually a laser scanner nowadays. They are very precise, efficient and the output does not require much computation to process. The dominating techniques for laser based range measurement are time of flight (TOF) techniques and phase-shift techniques. In a TOF system a short laser pulse is sent out and the time until it returns is measured. The distance from the object hit is given by D = (c*T)/2, where c is the speed of light and T is the round trip time. In phase-shift systems a continuous wave is transmitted. The idea is to compare the phase of the returned signal with a reference signal generated by the same source.

Fig 2.1 Hokuyo UTM-30LX Laser Rangefinder [3]

Using doppler shift, the velocity of the target can be measured in addition to the distance to it [11]. On the downside (limitations) of laser scanners:

- They are also very expensive.
- Laser scanners are looking at certain surfaces including glass, where they can give very bad readings (data output).
- Also laser scanners cannot be used underwater since the water disrupts the light and the range is drastically reduced [2].

2.1.2 Sonar

Sonar was used intensively some years ago. They are very cheap compared to laser scanners. Their measurements are not very good compared to laser scanners

Fig 2.2 MaxBotix EZ-1 Sonar Sensor [4]

measurement emitted from the scanner with a width of as little as 0.25 degrees. Sonar can easily have beams up to 30 degrees in width. Underwater, though, they are the best choice and resemble the way dolphins navigate.

2.1.3 RGB-D Camera (Vision)

One thing that sonar, laser scanners and other range-finding sensors, lack is the ability to use surface properties to localize and identify objects. Color or grey-scale images allow us to use a wider set of information to identify and localize features in the environment.

Fig 2.3 RGB-D Sensor (Kinect and Xtion Pro Sensor) [5]

The main advantages of vision sensors are:

- Large amount of information.
- Capability of getting 3D information about the environment.
- Cameras are passive sensors; they don't have to emit sound or light pulses as sonar and laser sensors.

The drawbacks are:

- Vision is highly influenced by the lighting.
- High computing requirement to extract the information from the images.

Nowadays, Microsoft Kinect sensor or Asus Xtion Pro Live sensor is used for performing RGB-D SLAM [1].

2.2 Process of SLAM

The SLAM process consists of a number of steps. The goal of the process is to use the environment to update the position of the robot. Since the odometry of the robot (which gives the robots position) is often erroneous we cannot rely directly on the

odometry. We can use laser scans of the environment to correct the position of the robot. This is accomplished by extracting features from the environment and re-observing when the robot moves around.

An EKF (Extended Kalman Filter) is the heart of the SLAM process. It is responsible for updating where the robot thinks it is based on these features. These features are commonly called landmarks. The EKF keeps track of an estimate of the uncertainty in the robots position and also the uncertainty in these landmarks it has seen in the environment [14].

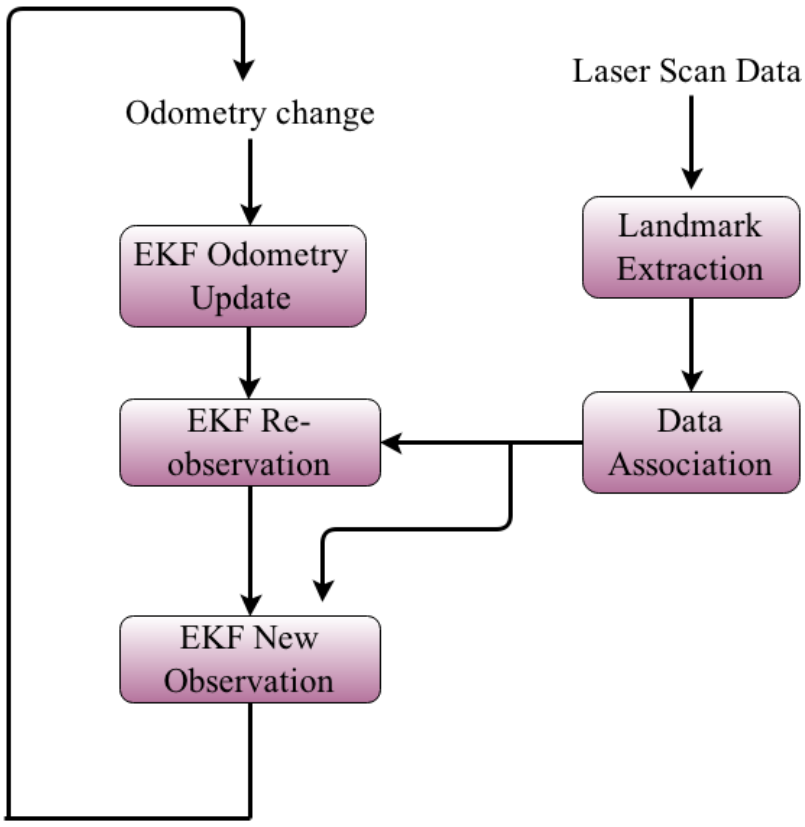

Fig 2.4 Overview of SLAM Process

When the odometry changes because the robot moves to a new position and the corresponding update is done in the EKF using Odometry update. Landmarks are then extracted from the environment for the robot's new position. The robot then attempts to associate these landmarks to the previously seen landmarks. Re-observed landmarks are then used to update the robots position in the EKF. Landmarks which have not previously been seen are added to the EKF as new observations so they can be re-observed later. It should be noted that at any point in these steps the EKF will have an estimate of the robots current position [11], [15].

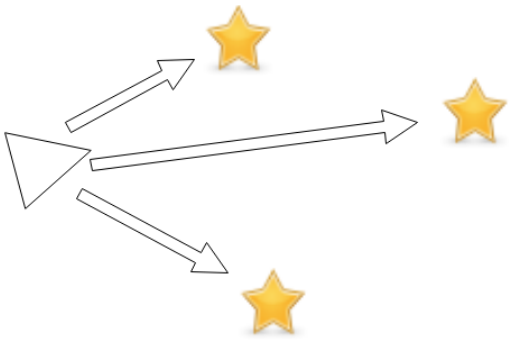

Fig 2.5 Position 1

The robot is represented by the triangle. The stars represent landmarks. The robot initially measures using its sensors the location of the landmarks (sensor measurements illustrated with arrows).

Fig 2.6 Position 2

The robot moves, so it now thinks that it is here in Position 2. The distance moved is given by the robots odometry.

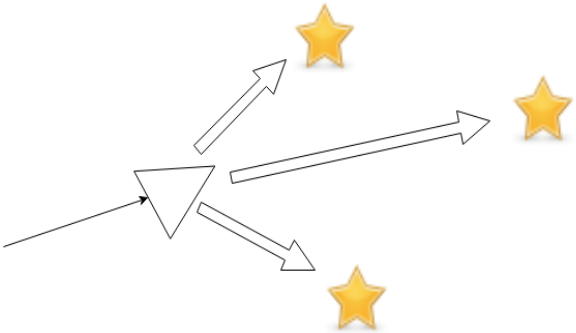

Fig 2.7 Position 3

The robot once again measures the location of the landmarks using its sensors but finds out they don't match with where the robot thinks they should be (given the robots location). Thus the robot is not where it actually thinks it is.

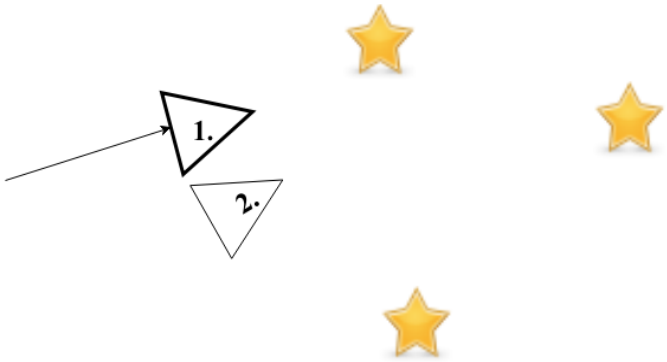

Fig 2.8 Position 4

As the robot believes more its sensors than its odometry it now uses the information gained about where the landmarks actually are to determine where it is (the location the robot originally thought is illustrated by the 2).

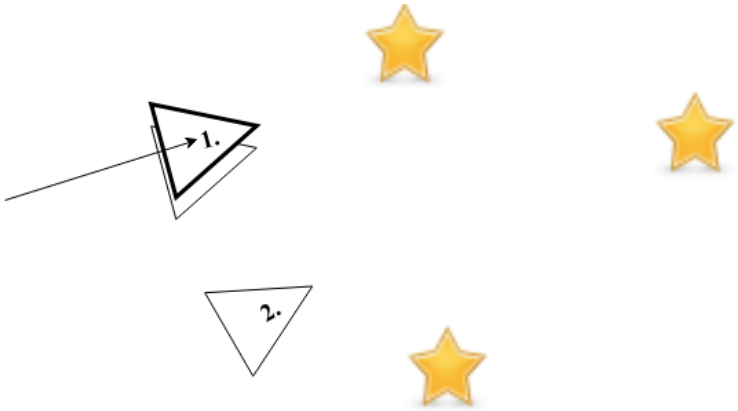

Fig 2.9 Position 5

In actual fact the robot is here at 1. The sensors are not perfect so the robot will not be able to precisely locate its position . However this estimate is better than relying on odometry alone. The triangle 2 represents position though by the robot; the triangle close to 1 is location given by the odometry; and the last triangle is the actual position of the robot.

2.4 Laser Data

The first step in the SLAM process is to obtain data about the surroundings of the robot. Generally, laser scanner we can output range measurements from an angle of

100° or 180°. It has a vertical resolution of 0.25°, 0.5° or 1.0°, meaning that the width of the area the laser beams measure is 0.25°, 0.5° or 1.0° wide. A typical laser scanner output will look like this:

2.98, 2.99, 3.00, 3.01, 3.00, 3.49, 3.50... 2.20, 8.17, 2.21

The output from the laser scanner tells the ranges from right to left in terms of meters.

2.5 Odometry Data

An important aspect of SLAM is the odometry data. The goal of the odometry data is to provide an approximate position of the robot as measured by the movement of the wheels of the robot, to serve as the initial guess of where the robot might be in the EKF. The difficult part about the odometry data and the laser data is to get the timing right. The laser data at some time t will be outdated if the odometry data is retrieved later. To make sure they are valid at the same time one can extrapolate the data. It is easiest to extrapolate the odometry data since the controls are known. It can be really hard to predict how the laser scanner measurements will be. If one has control of when the measurements are returned it is easiest to ask for both the laser scanner values and the odometry data at the same time.

2.6 Landmarks

Landmarks are features which can easily be re-observed and distinguished from the environment. These are used by the robot to find out where it is (to localize itself). One way to imagine how this works for the robot is to picturize ourselves blindfolded. If we move around blindfolded in a house we may reach out by touching objects or hugging walls, so that we don't get lost. Sonar and laser scanners are robot's feeling of touch. Landmarks should be unique enough so that they can be easily identified from one time-step to another without mixing them up. In other words if we re-observe two landmarks at a later point of time it should be easy to distinguish the current landmark and previously seen landmark. If two landmarks are very close to each other then this may be difficult to distinguish. Landmarks decided for a robot should not be so few in the environment that the robot may have to spend extended time. The robot may then get lost without enough visible landmarks. Landmark should be stationary. Using a person as a landmark is a bad idea.

CHAPTER 3: Algorithms involved in EKF-SLAM

3.1 Landmark Extraction

There are multiple ways to do landmark extraction and it depends largely on what types of landmarks are attempted extracted as well as what sensors are used. We will discuss the popular landmark extraction algorithm i.e. RANSAC using a laser scanner.

RANSAC

RANSAC (Random Sampling Consensus) is a method which can be used to extract lines from a laser scan. These lines can in turn be used as landmarks. In indoor environments straight lines are often observed by laser scans as these are characteristic of straight walls which usually are common [16].

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 lies 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. The below algorithm outlines the line landmark extraction process for a laser scanner with a 180° field of view and one range measurement per degree. Initially all laser readings are assumed to be unassociated to any lines. In the algorithm we only sample laser data readings from unassociated readings.

Algorithm:

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
- Using these S samples and the original reading calculate a least squares best fit line.

- Determine how many laser data readings lie within X centimeters of this best fit line.
- If the number of laser data readings on the line is above some consensus C do the following :
    - Calculate new least squares best fit line based on all the laser readings determined to lie on the old best fit line.
    - Add this best fit line to the lines we have extracted.
    - Remove the number of readings lying on the line from the total set of unassociated readings.

The EKF implementation assumes that landmarks come in as a range and bearing from the robots position. One can easily translate a line into a fixed point by taking another fixed point in the world coordinates and calculating the point on the line closest to this fixed point. Using the robots position and the position of this fixed point on the line it is trivial to calculate a range and bearing from this [16].

3.2 Data Association

The problem of data association is that of matching observed landmarks from different laser scans with each other.

In practice the following problems can arise in data association:

- We might not re-observe landmarks every time step.
- We might observe something as being a landmark but fail to ever see it again.
- We might wrongly associate a landmark to a previously seen landmark.

We assume that a database is set up to store landmarks we have previously seen. The database is usually initially empty. The first rule we set up is that we don't actually consider a landmark worthwhile to be used in SLAM unless we have seen it N times. This eliminates the cases where we extract a bad landmark.

i. When we have new laser scan then use landmark extraction to extract all visible landmarks.

ii. Associate each extracted landmark to the closest landmark we have seen more than N times in the database.

iii. Pass each of these pairs of associations (extracted landmark, landmark in database) through a validation gate.

- If the pair passes the validation gate it must be the same landmark we have re-observed so increment the number of times we have seen it in the database.
- If the pair fails the validation gate add this landmark as a new landmark in the database and set the number of times we have seen it to 1. This technique is called the nearest-neighbor approach as we associate a landmark with the nearest landmark in the database.

The simplest way to calculate the nearest landmark is to calculate the Euclidean distance. Other methods include calculating the Mahalanobis distance which is better but more complicated. The validation gate uses the fact that our EKF implementation gives a bound on the uncertainty of an observation of a landmark. Thus we can determine if an observed landmark is a landmark in the database by checking if the landmark lies within the area of uncertainty. This area can actually be drawn graphically and is known as an error ellipse [17].

3.3 Extended Kalman Filter

The Extended Kalman Filter is used to estimate the state (position) of the robot from odometry data and landmark observations. As soon as the landmark extraction and the data association is in place the SLAM process can be considered as three steps:

i. Update the current state estimate using the odometry data
ii. Update the estimated state from re-observing landmarks.
iii. Add new landmarks to the current state

In the second step the re-observed landmarks are considered. Using the estimate of the current position it is possible to estimate where the landmark should be. There is usually some difference, this is called the innovation. So, the innovation is basically the difference between the estimated robot position and the actual robot position, based on what the robot is able to see. 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 third step new landmarks are added to the state, the robot map of the world.

This is done using information about the current position and adding information about the relation between the new landmark and the old landmarks[14],[15].

3.3.1 System State

X is probably one of the most important matrices in the system which contains the position of the robot, x, y and theta. Furthermore it contains the x and y position of each landmark.

$$X = \begin{bmatrix} x_r \\ y_r \\ \theta_r \\ x_1 \\ y_1 \\ \dots \\ \dots \\ x_n \\ y_n \end{bmatrix} \quad (3.1)$$

It is important to have the matrix as a vertical matrix to make sure that all the equations will work. The size of X is 1 column wide and 3+2*n rows high, where n is the number of landmarks. Usually the values saved will be in either meters or millimeters for the ranges. The bearing is saved in either degrees or radians.

3.3.2 Covariance matrix

The covariance matrix P is a very central matrix in the system. It contains the covariance on the robot position, the covariance on the landmarks, the covariance between robot position and landmarks and finally it contains the covariance between the landmarks. The figure on the right shows the content of the covariance matrix P. The first cell, A contains the covariance on the robot position. It is a 3 by 3 matrix (x, y and theta). B is the covariance on the first landmark. It is a 2 by 2 matrix, since the landmark does not have an orientation, theta. This continues down to C, which is the covariance for the last landmark. The cell D contains the covariance between the robot state and the first landmark. The cell E contains the covariance between the first landmark and the robot state. E can be deduced from D by transposing the sub-matrix D. F contains the covariance between the last landmark and the first landmark, while G contains the covariance between the first landmark and the last landmark, which

again can be deduced by transposing F. So even though the covariance matrix may seem complicated it is actually built up very systematically. Initially as the robot has not seen any landmarks the covariance matrix P only includes the matrix A. The covariance matrix must be initialized

$$P = \begin{bmatrix} A & E & \cdots & G \\ D & B & \cdots & \\ \vdots & \vdots & \ddots & \\ & F & & C \end{bmatrix} \quad (3.2)$$

using some default values for the diagonal. This reflects uncertainty in the initial position. Depending on the actual implementation there will often be a singular error if the initial uncertainty is not included in some of the calculations, so it is a good idea to include some initial error even though there is reason to believe that the initial robot position is exact [14].

3.3.3 Kalman Gain

The Kalman gain K 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. If we can see that the robot should be moved 10 cm to the right, according to the landmarks we use the Kalman Gain to find out how much we actually correct the position, this may only be 5 cm because we do not trust the landmarks completely, but rather find a compromise between the odometry and the landmark correction. This is done using the uncertainty of the observed landmarks along with a measure of the quality of the range measurement device and the odometry performance of the robot. If the range measurement device is really bad compared to the odometry performance of the robot we of course do not trust it very much, so the Kalman gain will be low. On the contrary, if the range measurement device is very good compared to the odometry performance of the robot the Kalman gain will be high. The first row shows how much should be gained from the innovation for the first row of the state X. The first column in the first row describes how much should be gained from the

innovation in terms of range, the second column in the first row describes how much should be gained from the innovation in terms of the bearing.

$$K = \begin{bmatrix} x_r & x_b \\ y_r & y_b \\ t_r & t_b \\ x_{1,r} & x_{1,b} \\ y_{1,r} & y_{1,b} \\ .. & .. \\ ... & .. \\ x_{n,r} & x_{n,b} \\ y_{n,r} & y_{n,b} \end{bmatrix} \quad (3.3)$$

Again both are for the first row in the state, which is the x value of the robot position. The matrix continues like down through the robot position; the first three rows, and the landmarks; each two new rows. The size of the matrix is 2 columns and 3+2*n rows, where n is the number of landmarks.

3.3.4 Jacobian of measurement model

The Jacobian of the measurement model is closely related to the measurement model. The measurement model defines how to compute an expected range and bearing of the measurements (observed landmark positions).

$$\begin{bmatrix} Range \\ Bearing \end{bmatrix} = \begin{bmatrix} \sqrt{((\lambda_x - x)^2 + (\lambda_y - y)^2)} + v_r \\ \tan^{-1}((\lambda_x - x)/(\lambda_y - y)) - \theta + v_\theta \end{bmatrix} \quad (3.4)$$

where lambda x is the x position of the landmark, x is the current estimated robot x position, lambda y is the y position of the landmark and y is the current estimated robot y position. Theta is the robot rotation. This will give us the predicted measurement of the range and bearing to the landmark. The Jacobian of this matrix with respect to x, y and θ [14].

$$H = \begin{bmatrix} (\lambda_x - x)/r & (\lambda_y - y)/r & 0 \\ (y - \lambda_y)/r^2 & (x - \lambda_x)/r^2 & -1 \end{bmatrix} \quad (3.5)$$

H shows us how much the range and bearing changes as x, y and theta changes. The first element in the first row is the change in range with respect to the change in the x axis. The second element is with respect to the change in the y axis. The last element is with respect to the change in theta, the robot rotation. Of course this value is zero as the range does not change as the robot rotates. The second row gives the same information, except that this is the change in bearing for the landmark. This is the contents of the usual H for regular EKF state estimation.

3.3.5 Jacobian of Prediction model

Like H, the Jacobian of the prediction model is closely related to the prediction model. The prediction model defines how to compute an expected position of the robot given the old position and the control input. It is done using the following formula, which is denoted f:

$$f = \begin{bmatrix} x + \Delta x + q\Delta x \\ y + \Delta y + q\Delta y \\ \theta + \Delta\theta + q\Delta\theta \end{bmatrix} \quad (3.6)$$

Where x and y is the robot position, theta the robot rotation, Δx, Δy, Δtheta are control input for x, y and theta and q is the error term. The calculations of jacobian matrix are the same as for the H matrix, except that we now have one more row for the robot rotation. Since it is only used for robot position prediction it will also not be extended for the rest of the landmarks [14].

$$A = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \\ 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

CHAPTER 4: Simulation of SLAM on Matlab

4.1 Overview of the GUI developed for the Simulation

Graphical user Interface of SLAM simulation that I produced in MATLAB using the Extended Kalman Filter is shown in Fig 4.1. The implementation generates two motion estimates from laser scan matching and wheel odometry. The resulting pose estimate is a weighted average of the two estimates (decided by Kalman Gain) already discussed in earlier chapter, which takes into account their relative uncertainties. The update step uses artificial landmarks placed around the environment for pose correction.

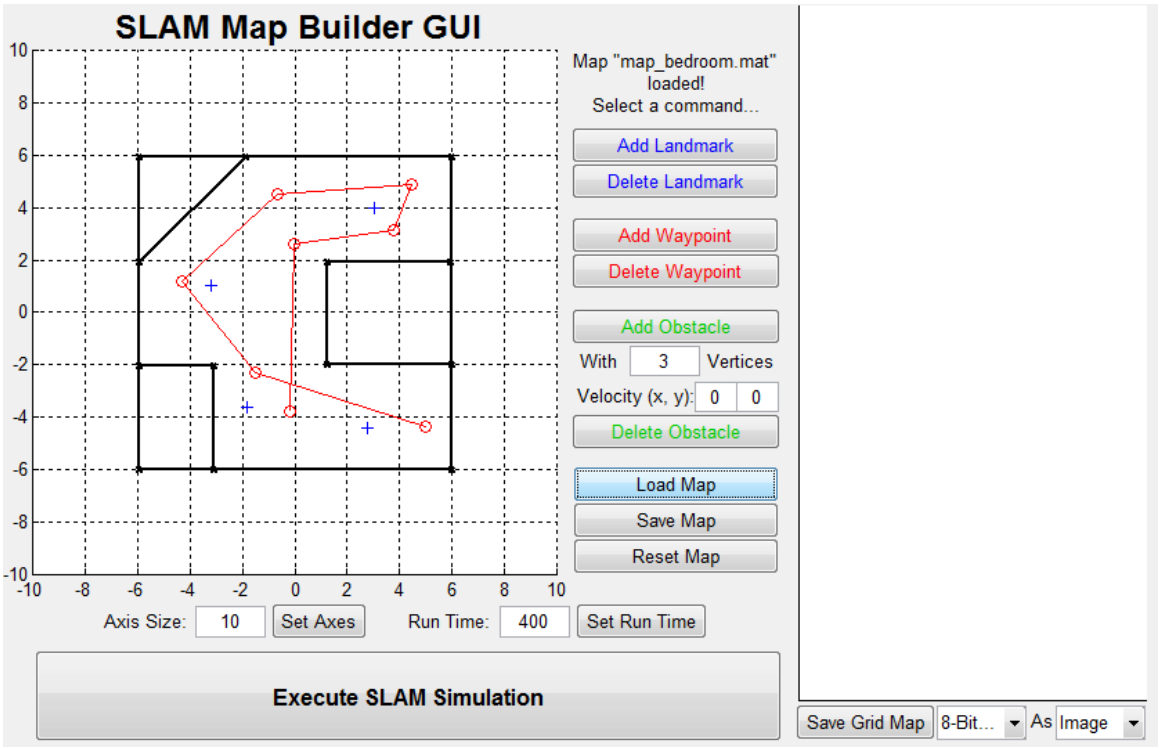

Fig 4.1 Graphical User Interface of the Simulation

'+' shows the landmarks available in the world, 'o' red line connects to the way point. We can create any custom map using the Landmark, Way point buttons and Obstacle given in the Graphical user interface. The Time of the simulation is decided by Run Time. New custom map can be saved in .mat file (In the form of array). The Left side of the GUI shows the map developed by the robot.

4.2 Results of the Simulation

As Robot moves along the waypoints, due to error in the control signals and range sensors, it deviates from its path. During simulation I have added some random noise in range sensor data and control data generated from motion. 1st shows the graph between EKF position error from over the time. Second Graph in the figure also explains about the error of the robot. And another graph is between the EKF uncertainty along x and y axis separately. Fig4.2 briefly explains the deviation of the robot from the actual waypoints in the world around x and y axis separately. Same graph is between the angular errors of robot movement from the actual angular movement. This graph plots the error generated due to wheel odometry and another graph is also showing error, but they are due to the error in the rangefinder readings. Third graph tells us about the Kalman gain. Graph is between the weight of reading obtained by the rangefinder and wheel odometry with respect to time Fig 4.3 tells about the map, on the left side is the binary map of the environment generated after the motion of the robot in the environment.

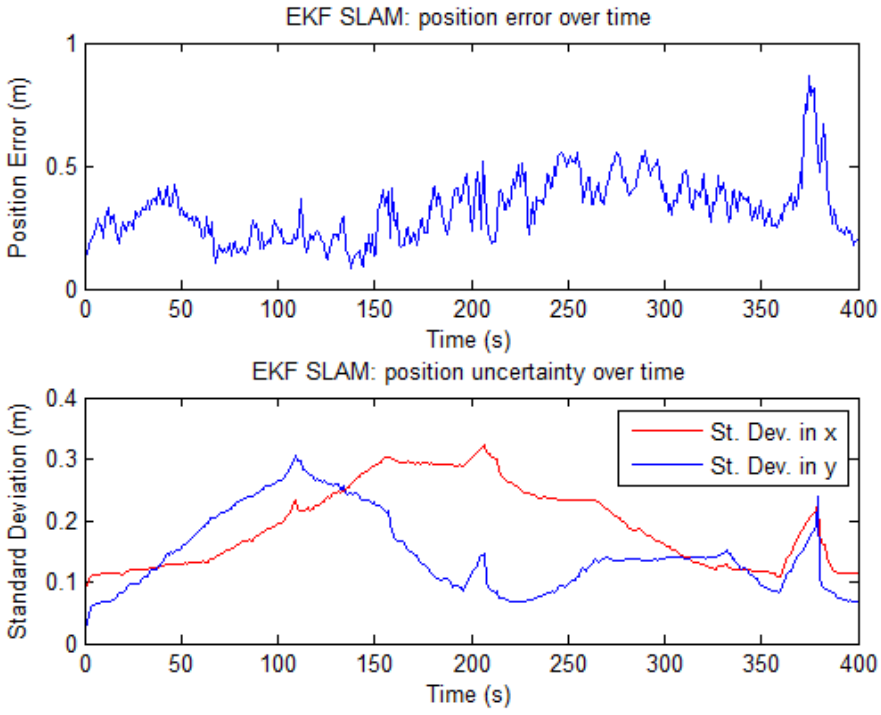

Fig 4.2 Graph between EKF SLAM error with respect to time

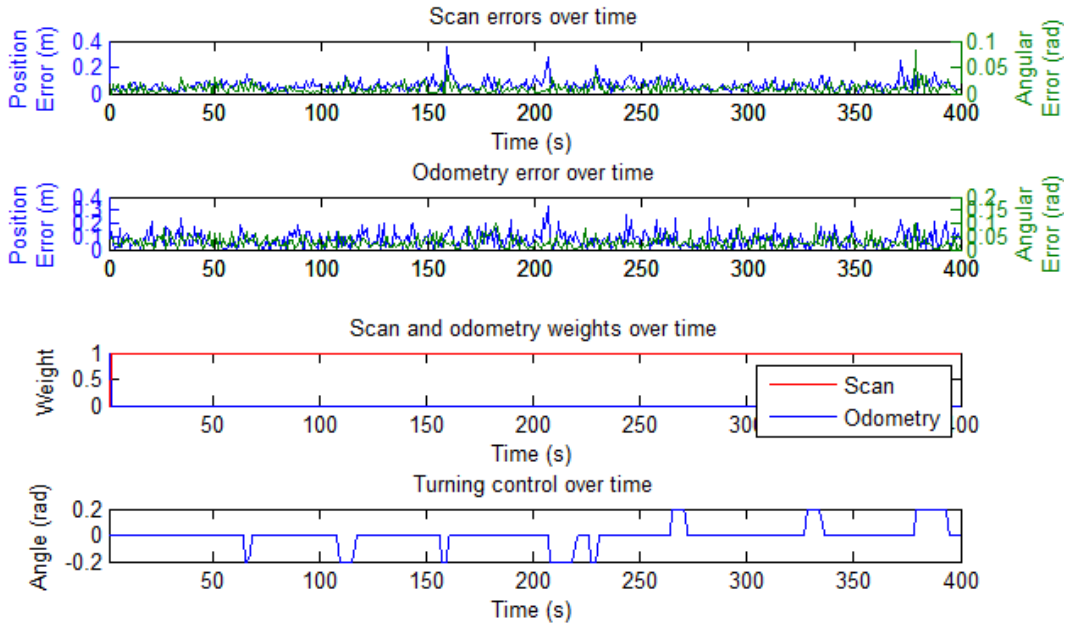

Fig 4.3 Graph between robot error with respect to time and weight of scan, odometry w.r.t. to time

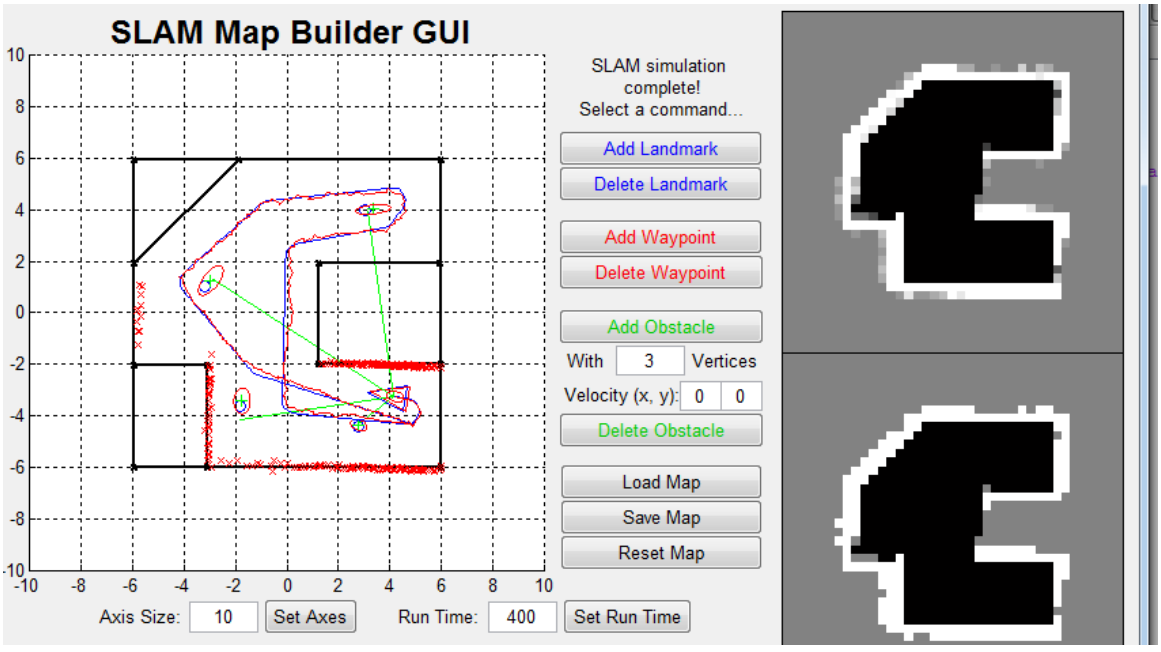

Fig 4.4 Map of Environment generated by the robot

CHAPTER 5: Applications of SLAM

5.1 Application of SLAM

5.1.1 Natural Disaster

In the context of a natural disaster, or when a military pilot has to eject in enemy territory, Search and Rescue teams often have to find people in unknown or hazardous areas. For safety reasons, Search and Rescue teams of the future will probably make use of unmanned aerial vehicles. For such a rescue vehicle, the ability to localize itself, both to avoid dangers (mountains/enemy bases) and to scan the entire area until survivors are found, is essential. However, this area may be unknown (enemy territory), or not mapped precisely (mountain summits). Moreover, global positioning systems (GPS) may not be usable in the area, or they may be not accurate enough, as is often the case in areas with dense foliage. In such a situation, a helicopter capable of mapping its surroundings while localizing itself on this map would be of special interest. Here, we can implement a SLAM algorithm for a helicopter moving in an area with uneven terrain and using 3-D rangefinder sensors.

5.1.2 Space exploration

Nowadays, SLAM is also being used in the planetary Rovers for exploring space. Since Rover cannot be remotely controlled in real-time since the speed at which radio signals travel is far too slow for real time or near-real time communication. For example, sending a signal from Mars to Earth takes between 3 to 21 minutes. These rovers are thus capable of operating autonomously with little assistance from ground control as far as navigation and data acquisition are concerned.

SpaceIL is an Israeli nonprofit organization of some engineers who are developing the smallest, smartest spacecraft to ever land on the moon. SpaceIL will be the first ever to use SLAM (Simultaneous Localization and Mapping) technology in space. There is no GPS in space, and the spacecraft has to auto-navigate all the way (384,000 km) to the Moon, so SpaceIL has developed innovative SLAM navigation technologies navigate the surface [18].

5.1.3 Autonomous Mine mapping

Hazardous operating conditions and difficult access routes makes mine exploration difficult for humans but SLAM can be used to develop the maps of mines. To create with an autonomous mobile robot a 3D volumetric map of a scene it is necessary to gage several 3D scans and to merge them into one consistent 3D model. In this paper [5] provides a new solution to the simultaneous localization and mapping (SLAM) problem with six degrees of freedom. Robot motion on natural surfaces has to cope with yaw, pitch and roll angles, turning pose estimation into a problem in six mathematical dimensions. A fast variant of the Iterative Closest Points algorithm registers the 3D scans in a common coordinate system and relocalizes the robot. Finally, consistent 3D maps are generated using a global relaxation.

To build 3D volumetric representations of environments with 2D laser range finders [6] has used two 2D laser rangefinder. One laser scanner is mounted horizontally and one is mounted vertically. The latter one grabs a vertical scan line which is transformed into 3D points using the current robot pose. The horizontal scanner is used to compute the robot pose. The precision of 3D data points depends on that pose and on the precision of the scanner. All these approaches have difficulties to navigate around 3D obstacles with jutting out edges. They are only detected while passing them.

Multiple 3D scans are necessary to digitalize environments without occlusions. To create a correct and consistent model, the scans have to be merged into one coordinate system. This process is called registration. If the localization of the robot with the 3D scanner were precise, the registration could be done directly by the robot pose. However, due to the imprecise robot sensors, the self localization is erroneous, so the geometric structure of overlapping 3D scans has to be considered for registration. The matching of 3D scans can either operate on the whole 3D scan point set or can be reduced to the problem of scan matching in 2D by extracting, e.g., a horizontal plane of fixed height from both scans, merging these 2D scans and applying the resulting translation and rotation matrix to all points of the corresponding 3D scan. Matching of complete 3D scans has the advantage of having a larger set of attributes to compare the scans. This results in higher precision and lowers the possibility of running into a local minimum of the cost function. Furthermore, using three dimensions enables the robot control software to recognize and

take into account changes of height and roll, yaw and pitch angles of the robot.

5.1.4 Underwater exploration

In the last decade, different underwater vehicles have been developed in order to explore underwater regions, especially those of difficult access for humans. Some widely used sensors for land and aerial robots do not work or are not precise enough underwater. For instance, the use of cameras is difficult due to the lack of visibility and scattering; the laser range finders are imprecise working in these scenarios because of light attenuation; and GPS signal does not work underwater [7].

A solution to the lack of GPS signal and the presence of noise are the Simultaneous Localization and Mapping (SLAM) algorithms. SLAM algorithms aim to build an approximate map of the area and calculate the approximate position of the vehicle within this map. In order to do so, SLAM algorithms combine the information coming from all sensors. To have a SLAM algorithm working properly, we need to select robust landmarks, i.e. objects, rocks and other salient elements. These robust landmarks must be easy to observe when seen for a second time, and easy to associate with previous observations. This procedure is important to close a loop, i.e. revisiting an area, because closing a loop means a reduction on the uncertainty and a more consistent final map.
SPARUS is underwater robot which is equipped with several sensing devices: Doppler velocity log (DVL), inertial measurement unit (IMU), down-looking camera, forward-

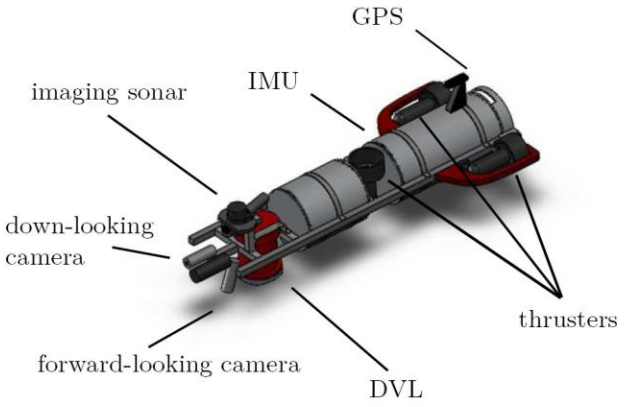

Fig 5.1 SPARUS Underwater Robot diagram [7]

looking camera, imaging sonar and GPS. Only DVL, IMU and down-looking camera are used, producing information about velocities, orientations and about the sea floor. The SLAM approach used here is the so called selective submap joining algorithm as discussed in [8]. The main idea of this approach is to use EKF based SLAM to build map.

Chapter 6: Conclusion

From this report, we have learned about the main challenges of SLAM, as well as the strengths and weaknesses of EFK SLAM. It is very robust, particularly when dealing with big objects, which are seen differently from different viewpoints. However, it requires a problematic pre-processing of the sensor information, since it treats the world as a set of point landmarks. Also, the shape of the landmarks cannot be taken into account, making the result even less accurate in chaotic worlds. In a very complicated world, when memory and speed are not main concerns, one should use EKF-SLAM [14], [15]. Finally we successfully implemented the SLAM simulation on the Matlab. Simulation provides the 2D map of the environment along with error in the track followed by the robot.

Future Scope

In coming future robots would have persistent autonomous navigation and mapping in dynamic Environment where objects surrounding the robots would be moving. So the day is not so far where we will see unmanned vehicles will roaming around from locations to location. Outdoor augmented reality typically relies on GPS, magnetic compass and inertial sensors for estimating position and orientation of the user's point of view. These sensors can exhibit large errors depending on the environment and transient disturbances. By combining these robust and absolute, but noisy, sensors with an online reconstruction system we can merge the relative, but accurate measurements of the SLAM system with the noisy global estimates. Efficient implementations of visual SLAM for mobile and handheld devices and appropriate sensor fusion models are still challenging task. Size of the RGBD sensor should also be reduced, occipital have launched a small form factor depth camera and they can be integrated with mobile. Now, mobile applications would be storing the 3d maps of our rooms so while purchasing items for your house, you can check whether it suits your house or it fits nicely in the place.

References

[1] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy.*" Estimation, planning and mapping for autonomous flight using an RGB-D camera in GPS-denied environments"* International Journal of Robotics Research, 2013

[2] David M. Cole and Paul M. Newman "*Using Laser Range Data for 3D SLAM in Outdoor Environments"* in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IR*OS),* Edmonton Alberta Canada, 2-6 August 2013, pp. 2089 – 2094.

[3]Downloaded HOKUYO LASER Image from from http://www.acroname.com/robotics/parts/R283-HOKUYO-LASER1.html downloaded on 04/11/2014 at 1:30 p.m

[4] https://www.maxrobotics.com/sonars/EZ2050 downloaded on 04/11/2014 at 1:15p.m

[5]S. Thrun, D. Fox, and W. Burgard. *"A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping"*, In Proc. of the IEEE ICRA, USA, 2012.

[6] D. Ferguson, A. Morris, D. H¨ahnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, W. Whittaker, W. Whittaker, W. Burgard, and S. Thrun. "*An autonomous robotic system for mapping abandoned mines".*

[7] RIBAS, D.; RIDAO, P.; TARDÓS, J.D.; NEIRA, J. (2013), "*Underwater SLAM in man made structured environments"*, J. Field Robotics 25/11-13, pp.898-921.

[8] AULINAS, J.; LLADÓ, X.; SALVI, J.; PETILLOT, Y.R. (2012), "*Selective submap joining for underwater large scale" 6-DOF SLAM*. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp.2552-2557, Taipei.

[9] Zeyneb Kurt-Yavuz and Sırma Yavuz *"A Comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM Algorithms"* IEEE 16th International Conference on Intelligent Engineering Systems • June 13–15, 2012, Lisbon, Portugal

[10] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit "*Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges"* in *IJCAI*, pages 1151, 2008

[11] Sebastian Thrun "*Learning Metric-Topological Maps Maps for Indoor Mobile Robot Navigation Artificial Intelligence"*, in *IJCAI*, pages 1298, 2012

[12] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit." *FastSLAM*: *A factored solution to the simultaneous localization and mapping problem*" 2002.

[13] Greg Welch and Gary Bishop "*An introduction to the kalman Filter"*, Technical Report TR 95-041, 2004.

[14] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte "*A new method for the nonlinear transformation of means and covariances in filters and estimator"*. IEEE Transactions on Automatic Control, 45(3):477–482, 2010.

[15] S. Huang and G. Dissanayake "*Convergence analysis for extended Kalman filter based SLAM*" in IEEE International Conference on Robotics and Automation, 2006.

[16] Javier Civera Oscar G. Grasa "*1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry"* Autonomous Robots, 2012

[17] Castellanos, J., Neira, J., and Tardos, J. (2004) *"Limits to the consistency of EKF-based SLAM "* in 5th IFAC Symposium on Intelligent Autonomous Vehicles

[18] Referred website for collecting data and understanding technology, http://www.spaceil.com/technology/