Sensor Modeling, Feature Extraction, Landmark Association and Extended Kalman Filter Implementation for a Mobile Robot

by

Ramkumar Natarajan
Manigandan Nagarajan Santhanakrishnan
Ramkumar Kannan

Annexure - IV: Detailed Project Report

Abstract

Autonomous Mobile robot plays a significant role in minimizing human intervention in a highly hazardous zone. It is imperative for an agent like a mobile robot to independently build a model of the surrounding environment as well as to localize itself within that identified zone. It is essential that the robot has to address simultaneously both the mapping and localization problem. The research community over the past two decades have come up with many innovative algorithmic solutions to provide meaningful estimates using highly uncertain information. This project is oriented more towards solving this Simultaneous localization and mapping(SLAM) problem using a scientifically well established theory of Extended Kalman Filter (EKF).The mathematical elegance of this method and the user friendliness of this algorithm has always attracted the research people in terms of improving the performance an inch further and the other reason for their interest is due to the fact that the real time technical issues was not well addressed so far. It is quite hard to get a literature on the implementation issues of this algorithm for solving the SLAM problem. This project report presents the successful demonstrations of the EKF-SLAM implementation on the Coroware mobile robot using both odometer and range scanning sensor. This project addresses issues related to mathematical modeling of both system and sensors, selection of EKF tuning parameters, analysis and interpretation of covariance matrices, feature extraction and data association techniques for better mapping and localization. The real time implementation difficulties are explained with technical insights. The algorithm was tested in real time with a structured environment and proved to be effective with 30cm accuracy.

Contents

List of Figures

Chapter 1

BACKGROUND AND MOTIVATION

1.1 Introduction

The application of Mobile robot pervades and encompasses a wide spectrum which includes underwater exploration, land mine detection, transportation, surveillance and other defense related operations. The primary challenge in these tasks is to deal with highly uncertain environment which increases the overall technical complexities. The uncertainty can be classified into system and sensor errors. Handling these uncertainties is one of the key challenges faced by the research community in navigating a mobile robot in a structured and unstructured environment.

When there is a restriction on use of external sensors such as Global Positioning System (GPS) localization of the Autonomous Mobile Robot (AMR) in an unknown environment becomes very challenging. The robot is left to localize itself based on the data from the on−board sensors and relative to the features seen while exploring the environment. This problem of simultaneously constructing a map of an environment amidst uncertain pose of a mobile robot as well as estimating the pose amidst changing environment is referred to as the simultaneous localization and mapping (SLAM). This mutual relation between the pose and the map estimates makes the SLAM problem difficult as the information gathered through various local sensors are uncertain due to sensor noise, time bound deformity in the physical structure of mobile robot and ambiance of the environment. Since large amount of uncertainty prevails in the sensor data, linear filters can seldom be used. So in order to estimate the current state of the robot, decision may be taken based on fresh observations of the same environment. This paves the path for probabilistic and statistical based stochastic filters. It is also important to observe that for each new environment that needs to be navigated by the mobile robot it is essential to supply a new map and if there are any changes in the surroundings it will have a huge impact on the pose estimate which leads to the problem of robot getting completely lost in the environment. To avoid this problem of supplying fresh maps continuously to the agent, the research community has come up with the idea of simultaneous localization and mapping problem. It is clear that without a good map, localization is not possible and without an accurate pose estimate it is difficult to build the map too. The question that is framed at this point is more like this.

Given a pose estimate and a fresh observed data, is it possible to fuse these data in building a updated map of the environment?

In the past two decades many researchers have contributed richly in this domain. Every time a measurement is made the belief factor on the estimate is improved. Thus the estimates are updated by fresh measurements, which normally will refine the original estimate removing the uncertainty. The filtering approach for this problem comes under two main categories and they are Kalman filters and particle filters. These filters are generally called as online SLAM algorithms.

1.2 Basic Kalman Filter

Kalman filter is a linear optimal state estimation filter based on mathematical models of the system and observation models in a stochastic environment. The fundamental procedure is very much evident from Figure 1.1 and Figure 1.2 which shows the block diagram and flowchart of a Kalman filter structure respectively. In the case of Simultaneous Localization And Mapping (SLAM) the system considered is nonlinear and discrete so a slightly modified and extended version of the linear Kalman filter known to be Extended Kalman Filter (EKF) is used.

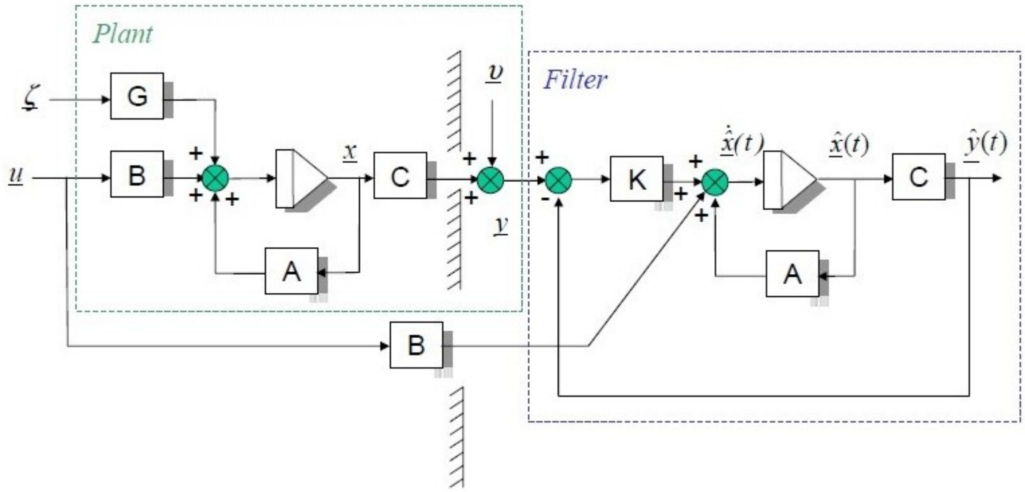


Figure 1.1: Block diagram of Kalman Filtering Algorithm.

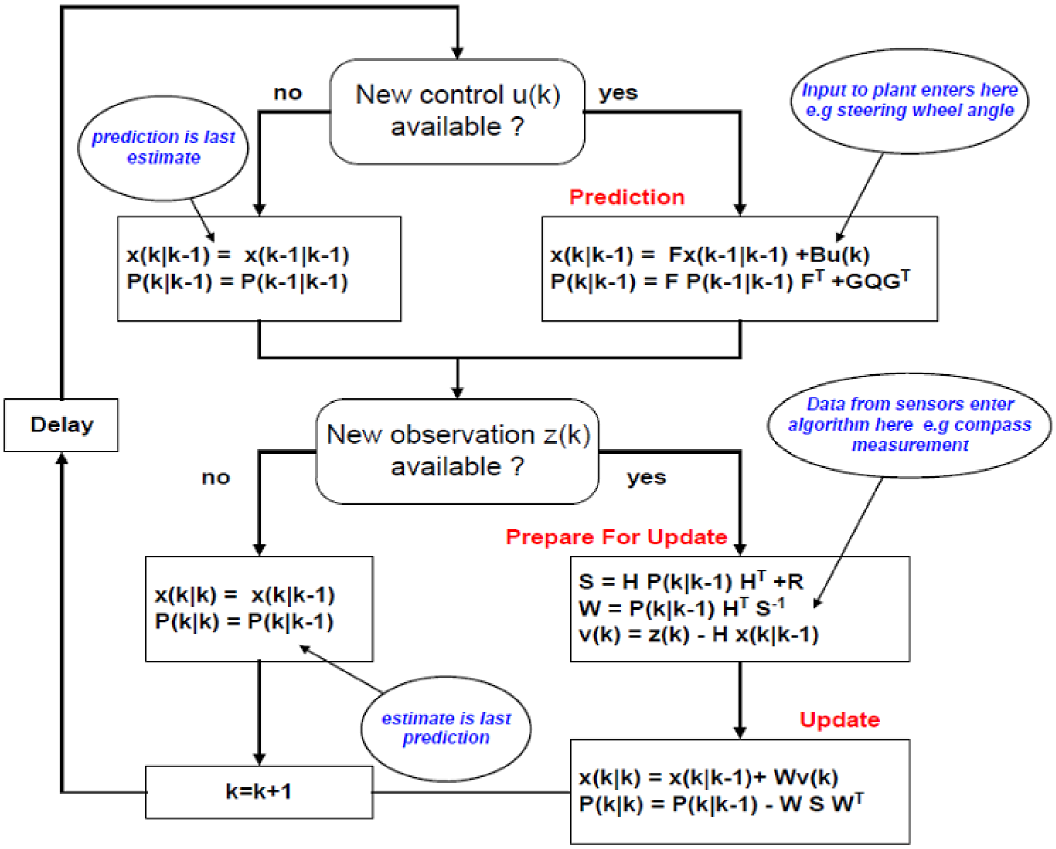

New control u(k) available ?
no
yes
prediction is last estimate
Input to plant enters here e.g steering wheel angle
Prediction
x(k|k-1) = x(k-1|k-1)
P(k|k-1) = P(k-1|k-1)
x(k|k-1) = Fx(k-1|k-1) +Bu(k)
P(k|k-1) = F P(k-1|k-1) F^T +GQG^T
Delay
New observation z(k) available ?
no
yes
Data from sensors enter algorithm here e.g compass measurement
Prepare For Update
x(k|k) = x(k|k-1)
P(k|k) = P(k|k-1)
S = H P(k|k-1) H^T +R
W = P(k|k-1) H^T S^-1
v(k) = z(k) - H x(k|k-1)
estimate is last prediction
Update
k=k+1
x(k|k) = x(k|k-1)+ Wv(k)
P(k|k) = P(k|k-1) - W S W^T

Figure 1.2: Flowchart of Kalman Filtering Algorithm.

To illustrate the SLAM problem let us consider the 1.3, which shows various landmarks in the map as m_j, the robot states as x_k and the control drive given to the robot as u_k. The robot obtains the location of the landmarks through the on−board sensors. The belief factor on these locations during the first observation is assumed to be the maximum. Once the locations of the visible landmarks are obtained from position x_k (the mapping problem) depending up on u_k the next possible state x_{k+1} of the robot is estimated. Then the range and bearing of the landmarks viewed at is estimated from. To refine the estimate a new re-observation is made once the robot physically moves to x_{k+1} (localization problem). This process is repeated for the entire mapping and localization process. The above process is being illustrated mathematically below in Equations 1.1, 1.2, 1.3 and 1.4.

The current state of the system is

$$s_k = f(x_k, m_k) \tag{1.1}$$

Note: The system model involves both robot and the map at any point of time.
The current robot state is

$$x_k = f(x_{k-1}, u_k, \nu_k) \tag{1.2}$$

The landmark state is

$$m_k = f(m_{k-1}) \tag{1.3}$$

The observation made at any instant of time is The landmark state is

$$y_k = f(x_k, m_k, \omega_k) \tag{1.4}$$

s_k is the state estimates and y_k is the observation made at each instant of time. ν_k and ω_k are the process noise and measurement noise respectively which is generally assumed to be totally uncorrelated and zero mean Gaussian noise.

This project addresses the real time implementation aspects of these stochastic filtering algorithms in order to satisfy the requirements of center of artificial intelligence and robotics (CAIR) lab-DRDO.

1.3 Related Works

Smith, Self and Cheeseman [19] in 1990 developed a path breaking approach of applying Extended Kalman Filter (EKF) as a stochastic estimating tool for SLAM. The mathematical foundation related to the veracity of a stochastic solution in estimating concurrently both the robot pose and the map was proved using EKF as a filtering tool in this work. They have established the idea of correlation between the estimates of not only the robots pose but as well as between the map features and they have proved that the correlation gets strengthened with every new observation using sensors. This seminal paper has opened up a new area of research in addressing SLAM problem because till that time mapping and localization were viewed as two different problems and cannot be addressed simultaneously. Though there were a number of practical implementations of EKF as a SLAM tool [20] [11] after the initial work done by Smith et al, the research community started to probe deeper into the shortcomings of EKF. The limitation was identified to fall under three broad categories such as computational complexity, estimation accuracy due to linearization errors and data associations. When the area of the map to be explored was small (less than 100 mts) EKF based SLAM gave appreciable results and when the map area increases the size of the co-variance matrix in EKF also increases phenomenally, making computation very difficult. When observed features in a map are distinct then data association of the information from the sensors on to the features are unique. But when identical features are observed and are very close to each other then data association itself becomes a challenging problem. The computational complexity issues associated with more number of features was solved by Thrun et al [24] using sparse information filter [SIF]. In the quest for addressing the linearization error problems with EKF as well as to reduce the computational burden the idea of Monto-carlo based particle filters was considered [2]. The problem with EKF is, it cannot represent the real probability distributions effectively using the first and second statistical moments. But a particle filter with the help of random samples can approximate the posterior probabilistic density function

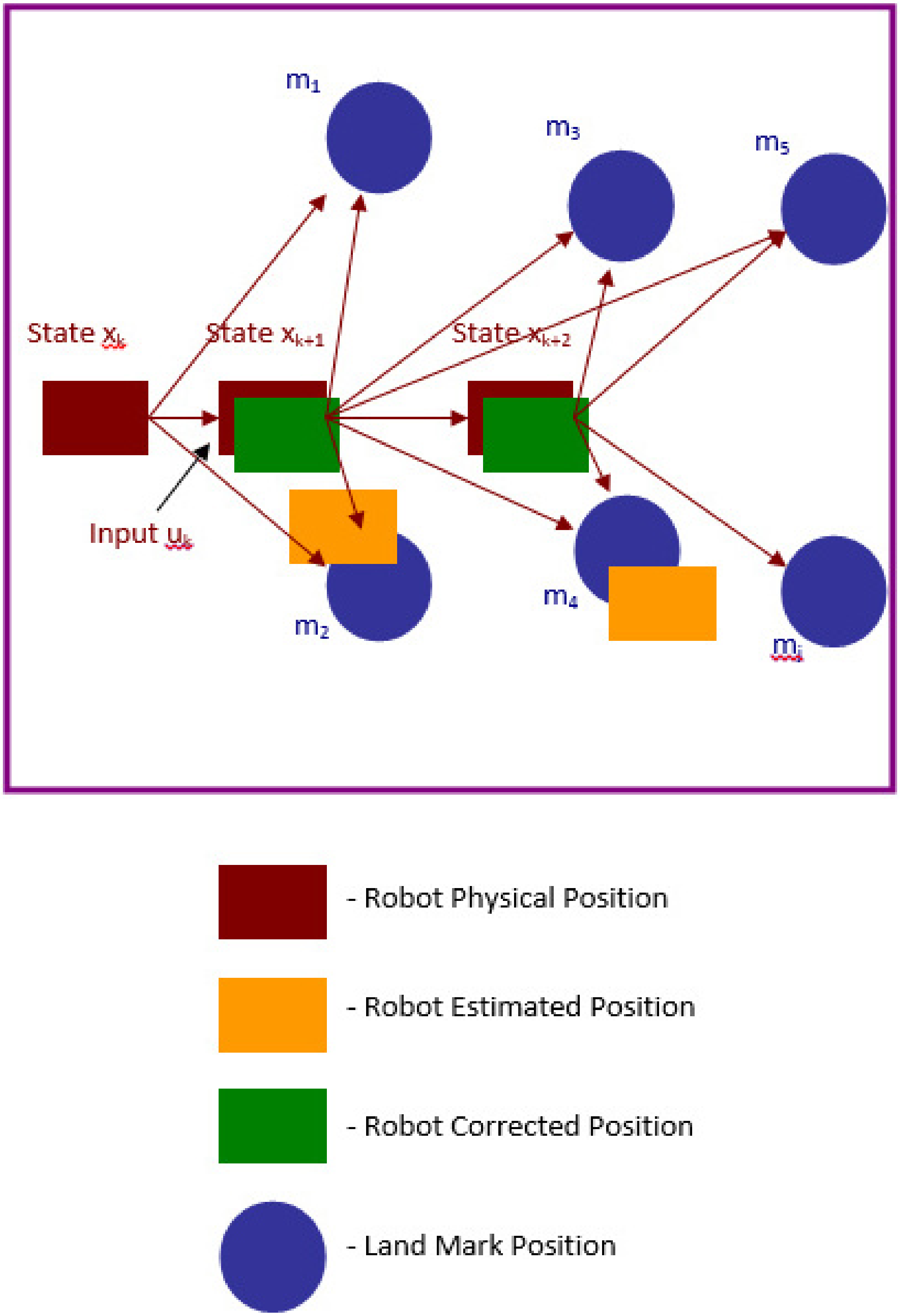


Figure 1.3: Pictorial representation for SLAM problem.

effectively. Montemerlo et al [15] [16] came up with two algorithms using particle filters for solving the SLAM problem and they are popularly called as FastSLAM algorithms. In order to address the linearization problem recently Unscented Kalman filter (UKF) based solution was also tried with reduced computational requirements [21]. In addition to these methods the problem associated with SLAM has propelled the research community over the past decade to come up with many novel versions of these algorithms.

The basic idea behind Kalman filter stems from the linear least square solutions formulated by Gauss during the year 1805. Kalman published the seminal paper on recursive least square algorithm as an estimation tool for linear control system problem in the year 1960 [7] [13]. Kalman filter as an estimation tool for mobile robot SLAM problem was implemented during 1990 [19]. In this work Kalman filter is applied for solving non-linear estimation problems using first order Taylor series approximation and it is popularly called as EKF. It is found from literature that the family of Kalman filters has been employed extensively in addressing the SLAM problem [7]. The importance of EKF as a foundational tool in solving the SLAM problem is emphasized clearly in the two significant recent survey papers [7][26]. EKF SLAM usually work with a state vector comprises of the robots pose and landmark features position. The online algorithm will eventually estimate the uncertainties involved in the robots pose and features location. The complete posterior correlation between the robot location and landmark map can be generated effectively using EKF. However when the number of features increases it creates huge computational burden, leading to poor real time estimation. For example for a 2D SLAM problem there will be 2N+3 states involved in the computation in any instant of time and the Covariance matrix will hold (2N +3) × (2N+3) elements and when N increases it leads to a quadratic increase in storage requirements as well as the computing time taken for each update of this matrix. This leads to a condition that full covariance matrix based EKF can be applied in real time to small maps (reduced number of landmarks) and reduced sensor update rates. The problem associated with this constraint is well analyzed in [10][9]. To provide solution to this problem an innovative method of using the inverse of the covariance matrix is given [23]. This inverse matrix is called generally as information matrix and it is identified to be sparse in nature. So eventually the computational burden got significantly reduced. This algorithm was made more efficient by the calculation of the error bounds which is investigated in [18].

The sparse nature of the information matrix can also be generated by including both features and a sequence of robot poses in the state vector as discussed in [5][4]. To reduce the computational burden an innovative SLAM with state vector containing only robot poses is proposed in [22] and this technique is known as Exactly Sparse Delayed state filter (ESDF). Wang et al [29] proposed a novel D-SLAM technique using the SIF but without using the previous robot poses in the state vector. In the similar fashion Walter et al [17] came out Exactly Sparse extended information filter (ESEIF). Though sparse nature of these algorithms reduces the computational burden, it increases the complexity of the recovery of the relevant elements in the information matrix as required in the data association stage. This process increases the computational cost. A comparative study on these information matrix filters are found in [30] and it is stated that DSLAM was consistent, information loss is less, memory demand is less and data association is more exact.

In order to implement effectively the EKF in a real time environment both the nonlinear robot model and the observation model should be reasonably accurate. These models can be incorporated in the online algorithm using various linearizing approximation techniques, which leads to errors in both landmark and robot pose estimation. These errors in turn lead to imprecise covariance matrix. The effect of this problem is studied in detail in [25]. Moreover sensor bias also cripples the performance of EKF to a large extent and recently researchers are addressing this problem using some novel sensor bias estimation algorithms [3, 14, 27].

The problems related to EKF are also approached from different modified versions of Kalman filters. Most notably it is observed from literature that the UKF as a tool for SLAM and its implementation issues are investigated in [8][1]. An unscented transformation is more related to particle filters where in the probability distribution functions (pdf) are sampled randomly. But in UKF it is done through careful selection of deterministic sigma points in order to maintain the moments of the distribution. In spite of the superiority of Unscented Kalman filter over EKF both these methods are basically affected by inconsistency problem. But it is also observed that by applying the UKF only to the robots states will results in more accurate covariance estimates and a more computationally efficient nonlinear transformation [8]. However, this technique is restricted only where the system and sensor noises are assumed to be Gaussian in nature.

Guivant [6] introduced a novel method of compressed EKF (CEKF) with an objective of improving the computational requirements of EKF-SLAM in large environments. Zhou et al [28] proposed the mean extended Kalman filter (MEKF) with a similar interest in reducing the computational burden of EKF. It is proved through simulation that the MEKF is computationally efficient. A comparative study of these families of Kalman filters was done using simulation and the results are published in [12]. The simulation work in this paper was carried out to compare the non iterated Sigma Point Kalman filter (SPKF) and EKF with their corresponding iterated versions and gives the comparative performance of these various versions of Kalman filters. It was inferred that iterated versions of both EKF and SPKF shows much reduced mean square error, normalized estimation squared error (NEES) and reduced computational time. The experimental details are explained well in [12]. The results that come out of these studies are to be validated in the real time environment with more field trials.

Chapter 2

PHASE I: ENVIRONMENTAL SETUP

Procurement of the equipment and Simulations

The Hardware and Software

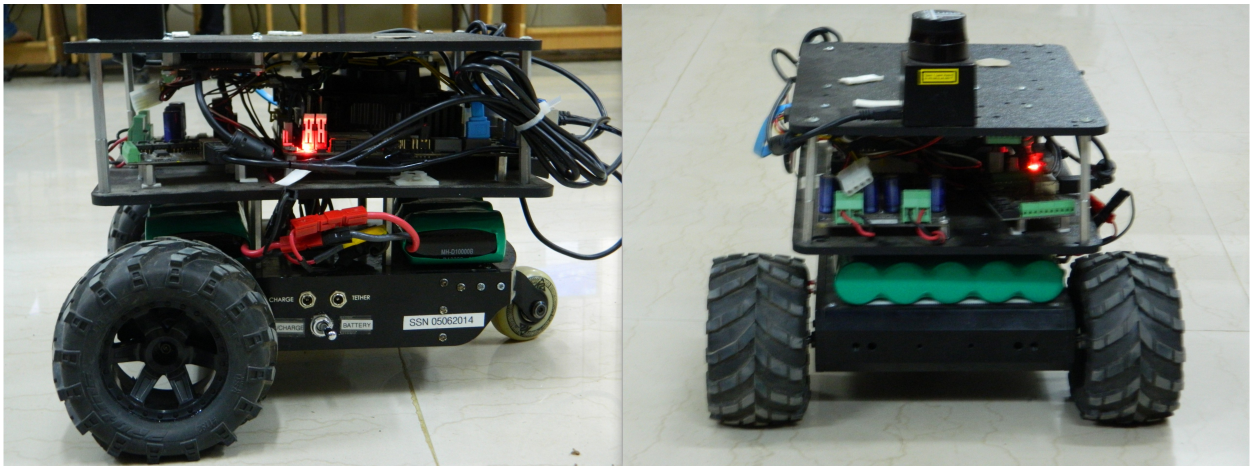


Figure 2.1: Snapshot of the experimental setup.

CoroBot Specifications

- Dimensions: 12"L x 13"W x 10"H
- CPU: 1.5 GHz
- RAM: 1 GB
- Disk Space / ROM: 80 GB
- Battery: 10AH

- Wheel Encoders: Yes
- Camera: High quality 2 MegaPixel color camera
- Inputs: 4 digital, 6 analog
- Outputs: 8 digital
- Wireless: 802.11 b/g/n
- Bumper Sensors: Front
- Base Payload Capacity: 5 lbs.
- Windows: XP, Supporting C-Language API
- Linux: Ubuntu, Player
- Pan/tilt camera: Yes
- Laser Range Finder: URG-04LX-UG01
- Max Speed: 1.5 feet/sec

2.1 Experimental Setup

The arena that is set for real time data recording and implementation is shown in Figure 2.6. Three types of environments such as E_1, E_2 and E_3 are created in the arena. E_1 of the arena is designed such a way that at any point of time if the robot is positioned in E_1, at least one feature is visible to the mobile robot. The features here are long orthogonal walls. The intersection points of these walls are considered as pointed features for localization purpose. E_2 is created to have small orthogonal walls and its intersection points are again considered as pointed features. E_3 does not have any of these defined features for a specified radius from the mobile robot.

The basic idea behind setting up such an environment is check the robustness of the feature extraction algorithm in identifying the point features generated through two different scenarios, when the MR navigates from E_1 to E_2. This will increase the trueness in the updated estimate of the MR by Kalman filter. The additive error on the dead reckoning sensor like the wheel encoder is corrected instantaneously and kept minimum all the time. If the robot is in E_3, it does not see any reliable or defined features thus forcing it to give the entire belief on the model. So again when the MR navigates back to E_1 or E_2 from E_3, the effectiveness of the entire localization problem can be understood and the model deviation can be quantified. Allowing the MR to traverse in all these environment sequentially or in random fashion the effectiveness of the model, feature extraction algorithm, association problem and the EKF algorithm is experimentally evaluated.

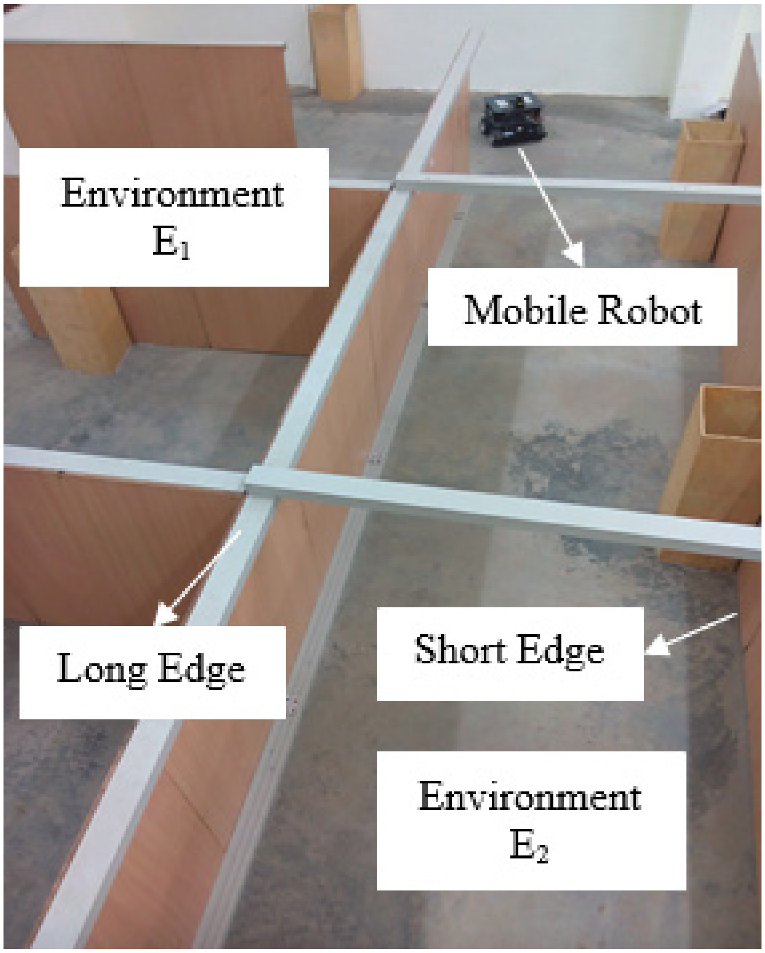


Figure 2.2: Experimental set up in a structured environment.

2.2 Simulation Results

Initially in order to test and understand the Kalman filter algorithm, simulation study was performed using Matlab and many insights about the tuning parameters of the filter was generated for doing further indepth studies.

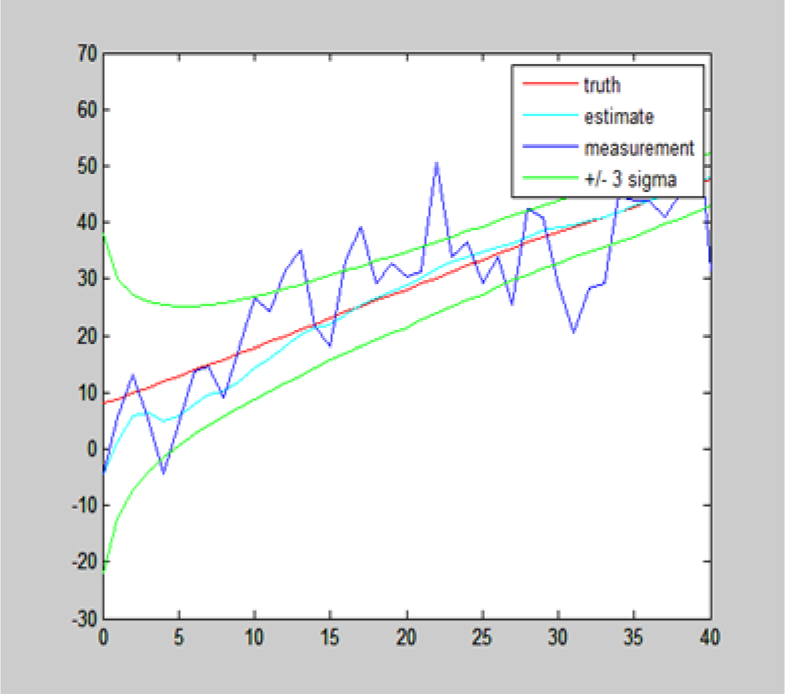


Figure 2.3: True position of the vehicle, the measured position, and the estimated position. Q = 0.01, R = 100.

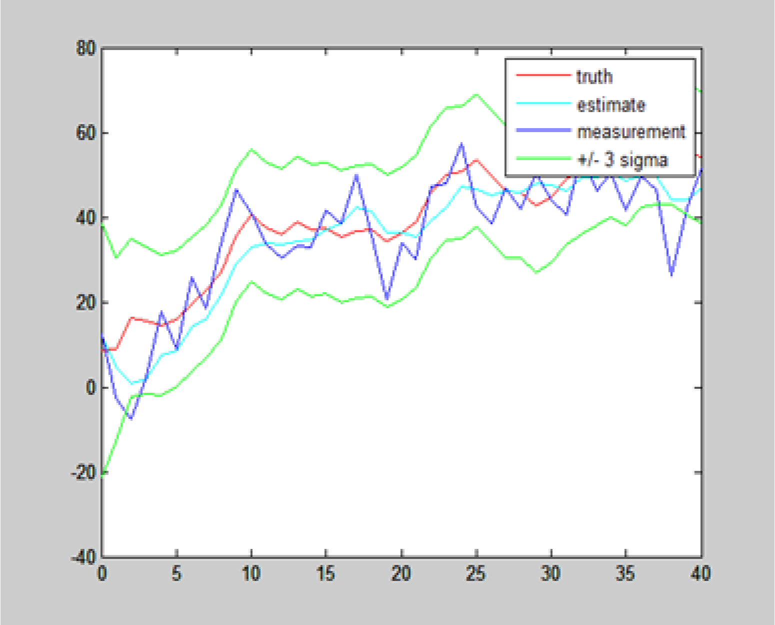


Figure 2.4: True position of the vehicle, the measured position, and the estimated position. Q = 10, R = 100

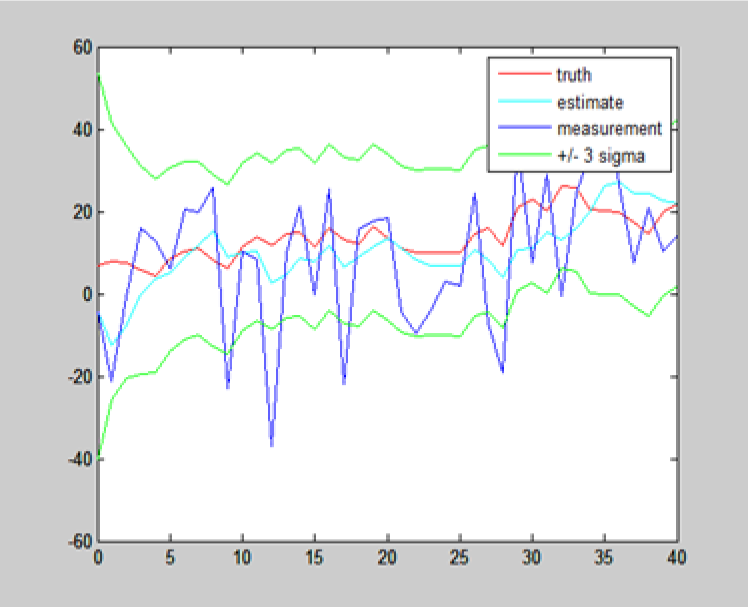


Figure 2.5: True position of the vehicle, the measured position, and the estimated position. Q = 10, R = 250

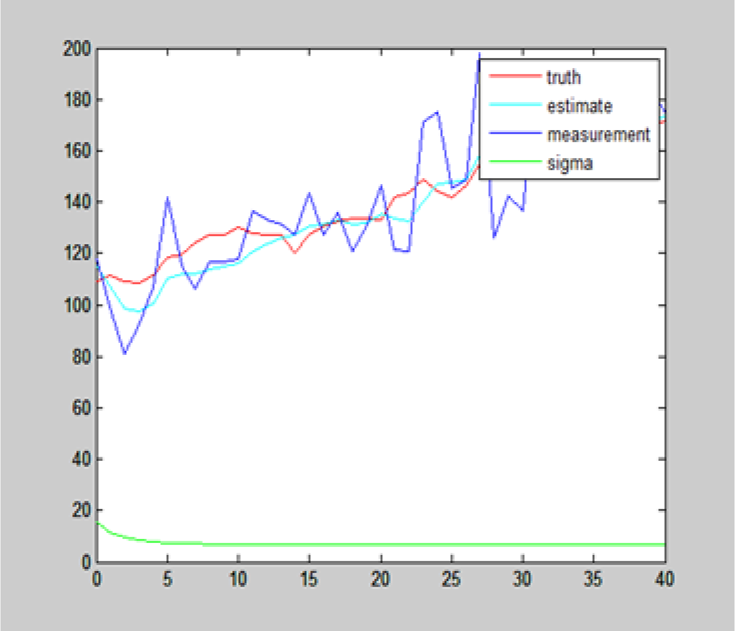


Figure 2.6: True position of the vehicle, the measured position, and the estimated position. Q = 10, R = 250 along with the variation in covariance.

Chapter 3

PHASE II: FEATURE EXTRACTION & OBSTACLE AVOIDANCE

3.1 Sensor Basics

Laser Range Finder: Ranger connected to the corobot at the port /dev/ttyACM0 provides readings at every 1/10th of a second. Ranger is having high credibility factor. Navigation of the mobile robot will be efficient when the decision of navigation is made in correspondence with Ranger data . The driver for Ranger with Player is as follows:

```
    driver
(
name "hokuyoaist"
provides ["ranger:0"]
portopts "type=serial,device=/dev/ttyACM0,timeout=1"
)
```

Ranger Data Statistics: The data from the range finder has the following specifications:

Range: 5.5m
Field of view: −2.07 rad to +2.07 rad
Number of reading: 682
Angular resolution: 0.35

The robot was initially set on a random walk to get an approximate error present in the sensor. The odometer was given 100% credibility. The Map was built with the corresponding Ranger and Odometry data. Feature(corner) extraction was done and the statistical values (mean, standard deviation and variance) were found for each corner. The results are presented below in section 3.2.

3.2 Feature Extraction

Features are landmarks 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). The feature that interests us is the corner of the walls which make a 90 degree. The algorithm used by us simply identifies the straight lines available in the ranger data with a threshold length. Then the existence of a corner in the straight lines is just a mere multiplication of any two given lines slopes. The pseudocode of Feature extraction is given in Algorithm 1. Also the graphical representation is depicted in Figure 3.1

```
Input: Sensor data and constants.

    • Laser ranger data from Ranger Proxy.

    • Threshold values.

Output: Position of corner (Feature).
while Any two points from Cartesian coordinates are processed. do
    Take any two points
    Form a straight line with the two points.
    Test the configuration space with the straight lin.
    if hits > threshold value then
        Include the line in the observed lines.
    end
    Test for any perpendicular lines in the observed lines
end
return Coordinates of the extracted feature (corner).
```

Algorithm 1: Feature Extraction Algorithm

3.3 Fuzzy based Obstacle Avoidance

System Architecture

The algorithm has two phases Perception and Decision. The two phases are implemented as separate modules. The diagram in 3.2 depicts the flow of data in the process. The Laser sensor and encoder hold communication with the perception module through the HAL (Hardware Abstraction Layer) provided by the Player toolkit. The Player/Stage toolkit provides the functionality of proxies which abstracts the layer of communication between the device and the program. Corresponding device proxies provides appropriate methods to control the devices. The perception module senses the environment and updates the robots current position from encoder by Laser proxy and Position2d proxy correspondingly. Perception module then interacts with the decision module to obtain the appropriate solution. As the

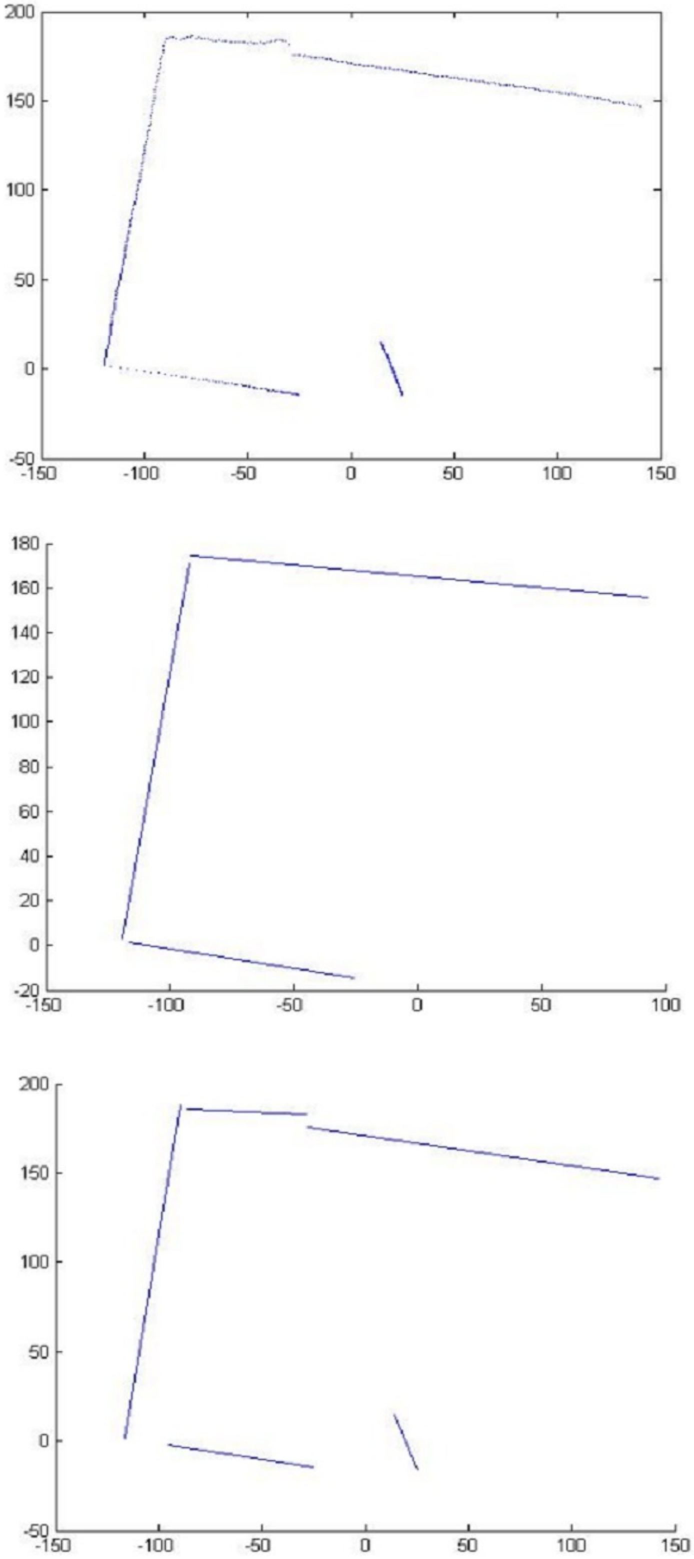


Figure 3.1: Sample line features.

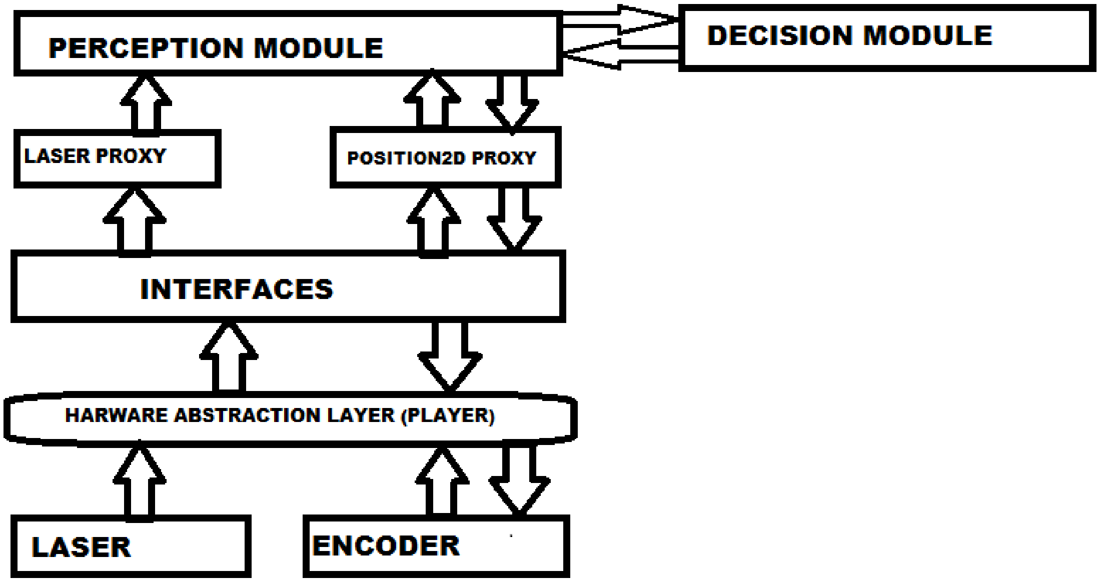


Figure 3.2: Software Architecture.

solution is to control the motor, which is done by Position2d proxy, the control flows from perception module to position2d proxy only.

Implementation

The algorithm implemented makes the robot move forward until it is confronted by an obstacle, and then it decides a navigable path. The perception phase computes all the navigable paths in the environment. When the forward path is challenged, it passes the solution set to the decision phase so that an appropriate path is selected based on the obstacle density in the environment. The detailed implementation of the algorithm is as follows:

Perception phase: During this phase the robot keeps moving forward [21] sensing the environment. The algorithm is supplied with a parameter $safedistance$, which keeps the sensor at a distance safe from the obstacles and prevents it from colliding from the obstacle. This should be greater than $2 * l$, where l is the length of the robot. Another parameter is *deciding distance*; a suitable distance chosen that is greater than $safe$ *distance* and closer to the range of the sensor with maximum credibility factor.

Membership function: The membership function for the LRF readings is derived as,

$$\mu(x) = x/(1 + max(x)), \quad \forall x \in [safe\_distance, max\_range] \tag{3.1}$$

where x ∈ LRF readings. This provides a fuzzy set with values [0,1) corresponding to the environment.

Defuzzification: The Defuzzification by $\lambda-$cut is deployed to determine all the navigable paths for the robot. The λ value is chosen based on the deciding distance. The fuzzy value for deciding distance is the λ. Every fuzzy value ≥ λ-cut is assigned 1 otherwise 0. All the possible paths are determined from the defuzzified crisp set obtained. The minimum width ($THRES$) for navigation is obtained from the number of range readings lying above the deciding distance as follows:

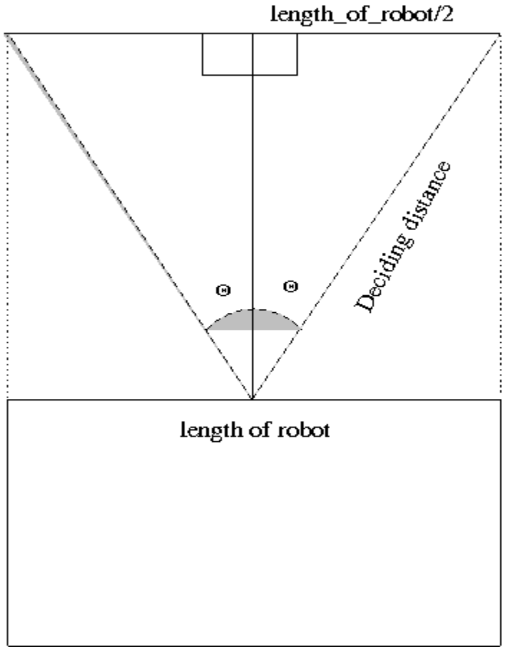


Figure 3.3: Determining THRES.

With known angular resolution, dimension, safe distance, FOV, deciding distance, THRES is computed.

Deciding distance is user chosen value ≥ Safe_distance
Safe
$\Theta = sin^{-1}$ **(length_of_robot/(2*deciding_distance))**
THRES=2* Θ/(**sensor_angular_resolution)**

The THRES consecutive number of range readings should be 1 in order to be considered as potential solution.

Decision Phase

The following in Algorithm 2 is the psuedocode in deciding the path by analyzing the nature of the environment.

```
if Length of $S_i > 0$ then
    if Counterfailed $> 3$ then
        Environment = CLUTTERED
        Take the solution from the set $S$ with largest count.
    end
    else
        Environment = SPARSE
        Take the solution from the set $S$ with minimum turn.
    end
end
else
    Environment = CLOSED
    Backtrack to the previous decision point.
    Decide from $S_{i-1}$-{solution}
end
return Type of environment(Environment).
```

Algorithm 2: Algorithm to analyse the nature of environment.

Based on the value of *Counterfailed*, the robot learns the nature of the environment. The parameter *Counterfailed* associates to numbers free spaces that are not potential navigable paths. With this, the robot classifies the environment into one of the following three types and decides a path based on that.

(i)Cluttered Environment: When the $Counterfailed > 3$, which signifies that there are more than three free spaces but cannot accommodate the robot. The robot learns that it is in a cluttered environment 3.4. In such a case the robot finds a path having the widest navigable width from the set S, which comprises of all the possible paths from that position of the robot. It tries to evade the cluttered obstacle in a way that provides a better navigable path.

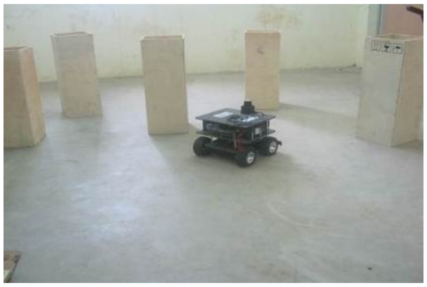


Figure 3.4: Cluttered environment.

(ii)Sparse Environment: If the $Counterfailed < 3$, the robot assumes that it is in a sparse environment 3.5. The robot from its set of solution, S chooses that path which has relatively minimum turn angle with respect to the current pose. This is better as the robot's orientation is not disturbed much when confronted with fewer obstacles.

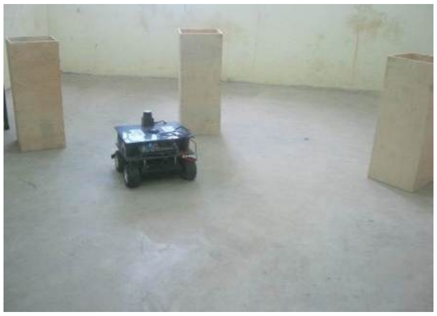


Figure 3.5: Sparse environment.

(iii)Closed Environment: The worst case lies in the fact that when the environment is closed 3.6. The solution set is null. Whenever a solution set becomes null the robot retraces the path to reach previous decision point and chooses the path not chosen. Retracing the path in a static environment does not result in collision. When n^{th} position is closed it goes to the $(n-1)^{th}$ position which is possible.

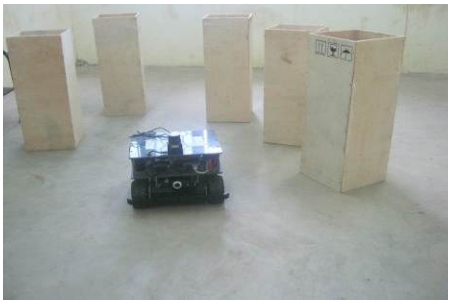


Figure 3.6: Closed environment.

The algorithm saves the solution set at every decision point and laser scan to aid exploration and mapping.

Obstacle Avoidance Algorithm

The proposed algorithm for Fuzzy based Obstacle Avoidance is given in Algorithm 3. The algorithm runs for every scan data coming from the Laser Range Finder. This algorithm runs after classifying the environment as cluttered, sparse or closed.

```
Input: Sensor data and constants.
  • $L$ is the laser ranger data from Ranger Proxy.
  • $THRES$ is the threshold value.
Output: Largest opening to navigate.
while $L \neq \emptyset$do
    if path in front is navigable then
        Move forward
    end
    else
        S= {}
        Fuzzify the entire LRF reading using the function $\mu(x) = x/(1 + max(x))$
        Apply $\lambda - cut\ to\ the\ fuzzy\ set$
        Counterfailed=0
        for Each element in Fuzzy set do
            if element is 0 || end of the fuzzy set is reached then
                if count >THRES  count >0 then
                    Solution=(index,count)
                    S= S $\cup$ {Solution}
                end
                else
                    Counterfailed+=1
                end
                Count = 0
            end
            else
                Increment count
            end
        end
    end
end
return Largest opening in the surrounding.
```

Algorithm 3: Fuzzy obstacle avoidance algorithm

The THRES in the algorithm is obtained from Equation 3.2. S mentioned in the algorithm denotes the set of all possible solutions in the environment at a particular decision point. Counterfailed keeps count of set of continuous values that are not potential solutions. The value of the Counterfailed is directly proportional to the number of obstacles in the environment.

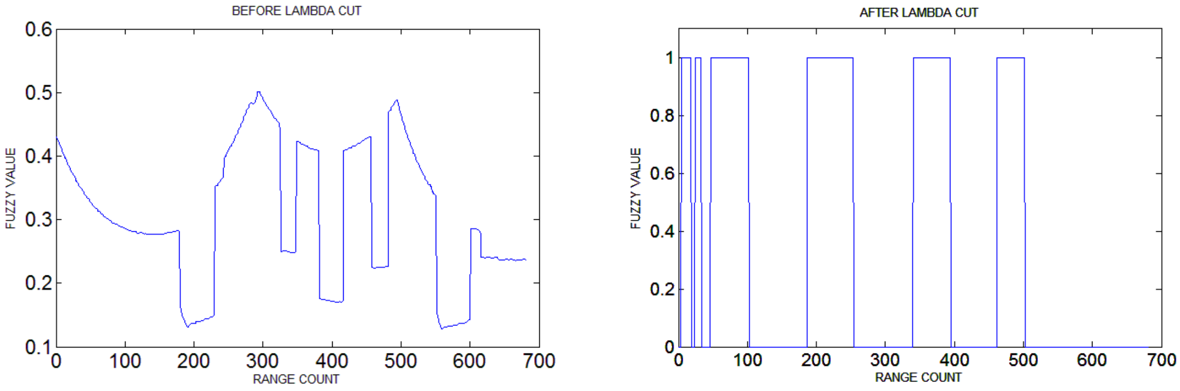
$$THRES = 2 * \Theta/(Sensor\_angular\_resolution) \tag{3.2}$$

The algorithm above makes the robot move forward until confronted by obstacles. As the robot keeps sensing the environment, it finds every potential solution with respect to the deciding distance. Then chooses a path from the set of potential solutions. The chosen solution depends on the type of the environment the sensor is in.

The number of solutions and Counterfailed is considered to categorize the environment. In a SPARSE environment, the solution having minimum turn is chosen. On the other hand in a CLUTTERED environment the sensor chooses the solution having maximum width. It goes to the previous decision taken point if the environment is CLOSED.

3.4 Experimental Result

The algorithm was tested in real time and graphs of fuzzy set and defuzzified crisp sets in different environments are provided below to explain the results briefly. The data is obtained from the LRF of the robot. Value of λ in λ-cut in all the examples quoted is assumed as 0.3.



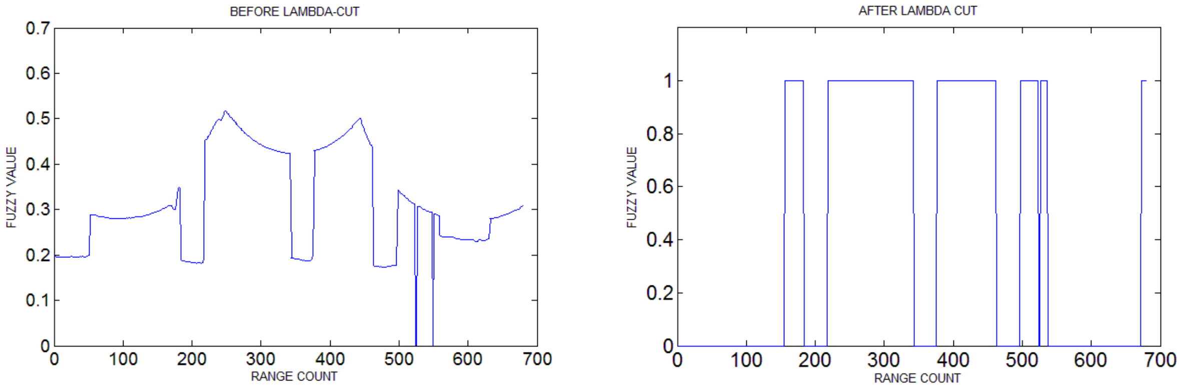
(a) Fuzzy set

(b) Defuzzified set

Figure 3.7: Cluttered environment.

In Figure 3.7a, the robot is confronted by cluttered obstacles. This is known from the number of spaces that cannot accommodate the robot through it. The robot then decides to navigate through the path having larger width, which is found from the count in the solution set. Figure 3.7b, shows the values post λ-cut. This clearly shows two navigable paths.

In Figure 3.8a, the robot is in front of sparse obstacles. The set S, comprising of every possible solutions from that pose has lesser number of solutions and the $Counterfailed$ also

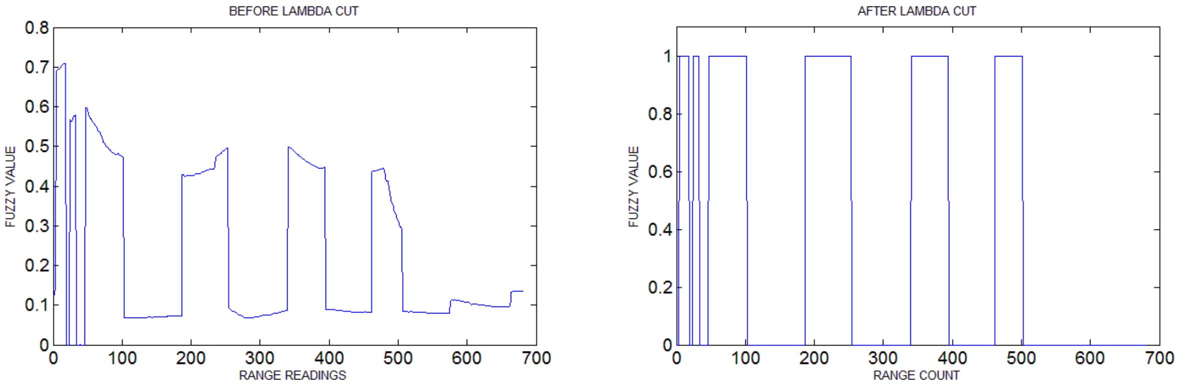


(a) Fuzzy set

(b) Defuzzified set

Figure 3.8: Sparse obstacle environment.

denotes the existence of sparse obstacles. The robot then decides to navigate through the path having minimum turn angle, which is found from the *solutionset*. The defuzzified crisp set′s graphical representation in FIgure 3.8b, shows the values post λ-cut.



(a) Fuzzy set

(b) Defuzzified set

Figure 3.9: Closed environment.

From Figure 3.9a, it is clear that the robot has a closed environment in front of it. The robot reverts back to previous decision point. The Figure 3.9b, shows that the robot has not detected the threshold distance to navigate the obstacle, so it re-tracks.

Chapter 4

PHASE III: IMPLEMENTATION OF EXTENDED KALMAN FILTER

4.1 Introduction

Localization of autonomous robot with respect to the environment & mapping âĂŞ A challenging problem, Why?

- Influence of noise in the sensors
- Physical/ structural deformities in the system (The robot)
- Limitations on the sensor/ system
- Dynamic terrain conditions
- Data processing speed
- Memory limitations

The end effect is *uncertainty* in the measured data.

The basic steps of a SLAM process are:

- **Landmark extraction:**The LASER sensor is used to detect landmarks. Appropriate algorithms can be used to extract features from the available LASER readings. In the developed algorithm only point landmarks are considered.
- **Data association:**This involves the processes to be carried out when a landmark already seen before, is seen again. The positions of the landmarks are obtained and compared with the positions of the already stored landmark positions. A match is seen with an allowance of 20

- **State estimation:**Use the odometer data to estimate the next state of the robot. Here the robots position is estimated. Essentially the robots position is updated and the landmarks position is refined. (Stationary landmarks are assumed.)
- **State update:**Using the already obtained odometer readings, the probable LASER readings are predicted. The difference in the actual LASER readings and the predicted readings are then used to update the State matrix.

4.2 Mathematical Model

The Coroware made CL2 which is a two wheeled differential drive mobile robot was used for the experiment. The schematic of the setup used is shown in Figure 4.1 and the encoder mathematical model was used as system model as suggested by Sebastin Thrun. The Hukoyo made URG−04LX−UG01 model LRF was used which has a linear resolution of 0.02m and angular resolution of 0.35°. The maximum measurable distance of the LRF is 5.6m with ± 5% accuracy but however the range was restricted between 0.6 to 4 m with the accuracy limited to ± 3%. The LRF was fixed exactly at the centre of rotation (C_R) point of the MB. This increases the degree of accuracy in locating the features and also avoids using any additional algorithm for reestablishing the CR. Any minimum error that gets introduced in LRF assembly is left for the EKF algorithm to correct.

The fundamental model for the system and sensor is given by the Equation 4.1 and 4.2 from where the rest of the implementation of EKF begins. Equation 4.1 is a linear Kalman filter model which is modified to a nonlinear EKF model as in Equation 4.2.

$$\begin{aligned} X_k &= AX_{k-1} + BU_{k-1} + \omega_{k-1} \\ Y_k &= HX_k + \nu_k \end{aligned} \tag{4.1}$$

$$\begin{aligned} X_k &= f(X_{k-1}, U_{k-1}) + \omega_{k-1} \\ Y_k &= h(X_k) + \nu_k \end{aligned} \tag{4.2}$$

Where X is the state vector, A the state matrix, B the input matrix, U the input vector and ω the system noise with covariance Q($\sigma_\omega{}^2$,0) and mean zero. Y is the output vector, H the measurement matrix and ν the measurement noise with covariance R($\sigma_\nu{}^2$,0) and mean zero with ω and ν completely independent. The mathematical model of the MR is the wheel encoder model in this experiment as proposed by Sebastin Tharun and is described by the equations below.

$$X_k^- = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} \tag{4.3}$$

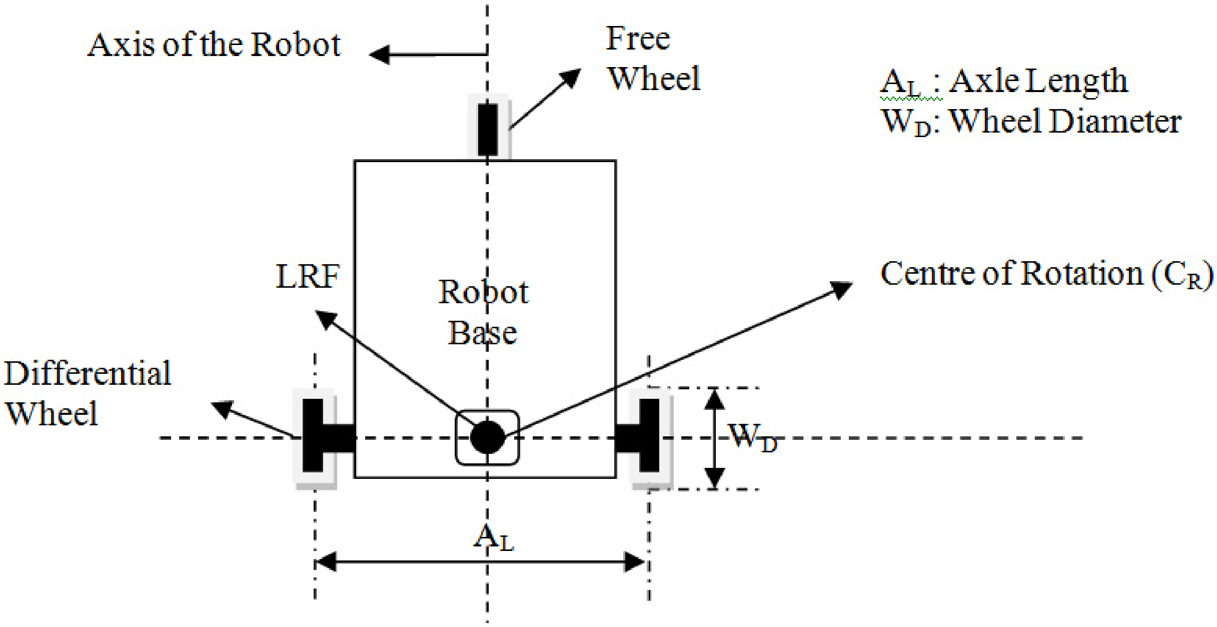


Figure 4.1: Schematic of a mobile robot

X_k^-, signifies the priori state estimation, prior in the sense that the position of the MR at k_{th} instant is estimated just with the knowledge of the $k-1_{th}$ instant posteriori estimate and using the mathematical model.

If $\Delta\theta \to 0$

$$\begin{aligned}\Delta x &= \Delta S \times cos(\theta_{k-1} + \Delta\theta)\\ \Delta y &= \Delta S \times sin(\theta_{k-1} + \Delta\theta)\end{aligned} \tag{4.4}$$

In other cases,

$$\begin{aligned}\Delta x &= \frac{\Delta S}{\Delta\theta} \times [sin(\theta_{k-1} + \Delta\theta) - sin(\theta_{k-1})]\\ \Delta y &= \frac{\Delta S}{\Delta\theta} \times [cos(\theta_{k-1}) - cos(\theta_{k-1} + \Delta\theta)]\end{aligned} \tag{4.5}$$

Δ x and Δ y are the displacement undergone by the MR along x and y direction respectively after an input U_{k-1} is applied. $\Delta\theta$ is the angular change exhibited by the MR around z direction.

$$\Delta S = \frac{D_r + D_l}{2}; \quad \Delta\theta = \frac{D_r - D_l}{A_L} \tag{4.6}$$

$$D_r = \frac{[L_t(k) - L_t(k-1)] \times \pi \times W - D}{T_R}; D_l = \frac{[R_t(k) - R_t(k-1)] \times \pi \times W - D}{T_R} \tag{4.7}$$

Where D_r and D_l indicates the distance travelled by right and left wheel respectively, $L_t(.)$ and $R_t(.)$ represents the left and right wheel encoder ticks and T_R the number of ticks per revolution.

The model is executed in two different forms i.e. when the MB moves straight, the change in linear distance is updated whereas the angular change is assumed to be zero. But when the MB undergoes a differential turn then the linear shift is assumed to be zero and only the angular shift is updated. The assumption in the model is that the C_R does not change during a differential turn. Any error that may occur due to this assumption and all other approximations are assumed to be the noise ω_k and are left for the EKF to attenuate. Since the equation 4.3 is non−linear in nature as the system states are not independent, Jacobian of the 4.3 as shown in 4.9 is used for covariance and state updating.

$$A_J = \frac{\partial f}{\partial X_k} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial \theta} \end{bmatrix} \tag{4.8}$$

Applying 4.8 on equation 4.3, we get

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

The observation model of the LRF shown by equation 4.10 gives the distance of the obstacle and its angle from C_R and its Jacobian is given by equation 4.12 and any noise that accompanies this measurement model is ϑ_k.

$$\gamma = \sqrt{(x_k - x_{F_i})^2 + (y_k - y_{F_i})^2}; \vartheta_{F_i} = \frac{\pi}{2} + tan^{-1}(\frac{y_k - y_{F_i}}{x_k - x_{F_i}}) \tag{4.10}$$

$$H_{J_i} = \frac{\partial h}{\partial Y_k} = \begin{bmatrix} \frac{\partial h}{\partial \gamma_{F_i}} \\ \frac{\partial h}{\partial \vartheta_{F_i}} \end{bmatrix} \tag{4.11}$$

Applying equation 4.11 on equation 4.10, we get

$$H_{J_i} = \begin{bmatrix} \frac{x_k - x_{F_i}}{\gamma_{F_i}} & \frac{y_k - y_{F_i}}{\gamma_{F_i}} & 0 \\ \frac{y_{F_i} - y_k}{\gamma_{F_i}^2} & \frac{x_{F_i} - x_k}{\gamma_{F_i}^2} & -1 \end{bmatrix} \tag{4.12}$$

Where i vary from 1 to M and M indicates all the associated features at the scanning instant k.

The noise that accompanies the model and sensor is assumed to be zero mean, Gaussian, white, mutually exclusive and independent with respect to sampling instances k. The noise covariance is initialized after conducting test bench experiments for the MR and its associated sensors. The noise covariance matrix for the system (Q) and sensor (R) model is given by equation 4.13.

$$Q = \begin{bmatrix} {\sigma_{\omega x}}^2 & 0 & 0 \\ 0 & {\sigma_{\omega x}}^2 & 0 \\ 0 & 0 & {\sigma_{\omega x}}^2 \end{bmatrix}; R = \begin{bmatrix} {\sigma_{\nu\gamma}}^2 & 0 \\ 0 & {\sigma_{\nu\vartheta}}^2 \end{bmatrix} \tag{4.13}$$

Derivation of Q and R

The mobile robot is made to run in a known environment, path and features for a fixed distance as seen in the figure 4.2. Now the data acquired from the laser range finder and its deviation attributes for R and difference between the total distance actually travelled by the vehicle and the data displayed by the encoder attributes for Q.

4.3 Feature Extraction and Data Association

Data preprocessing

Basically the data from wheel encoder and LRF are taken in this experiment to address the issue of localization. The data from the LRF is preprocessed before it is used for feature extraction. The very purpose of preprocessing is to remove the spurious data that is generated because of the environmental noise such as dust particles from which the laser may get reflected. The first process is to remove all the points that are greater than the best observable distance B_{od} (chosen to be 4 m) by the LRF. Also due the inherent property of discrete sampling of the environment, there is a possibility of the sharper edges being missed. Nonconventional filters such as moving average filters and median filters are used to remove these noises and retain the sharper edges. Moreover the environment is not sampled evenly, the features that are closer to the LRF are sampled more than the farther one. This over sampling of the nearer objects is an unwanted burden in the sense of data processing as the same efficiency can be achieved with lesser samples. The farther objects which see lesser samples may reduce the efficiency of the process, so data embedding is done appropriately. Thus data re−sampling is done to achieve better processing speed with the improved efficiency. An adaptive data re−sampling technique based on the distance of the object from the LRF is proposed for the above purpose. Thus the distance between two samples where an obstruction was sensed is adjusted to be 15−30 mm after all preprocessing. The raw data sampled by the LRF is shown in Figure 4.3a and the data after preprocessing is shown in Figure 4.3b.

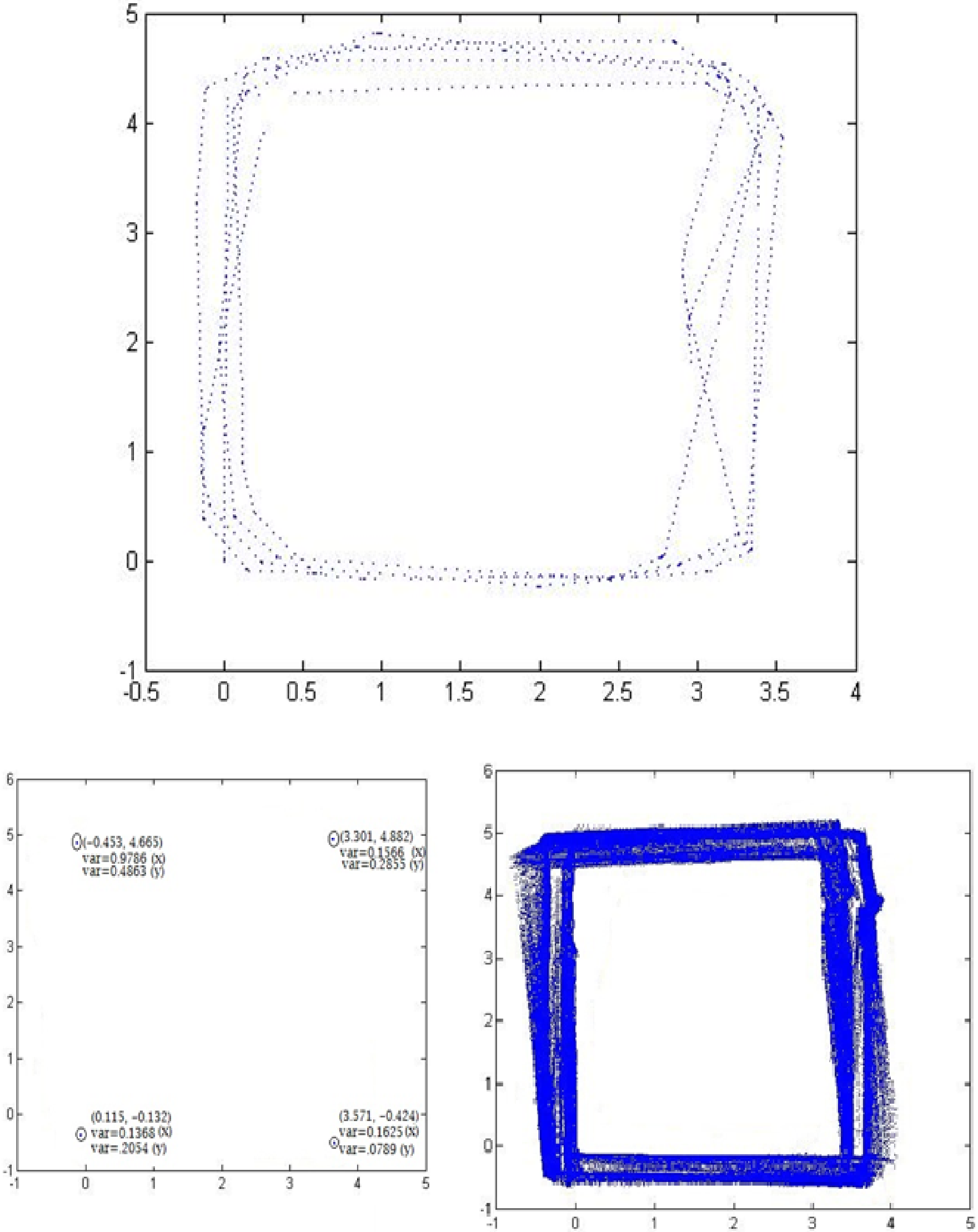
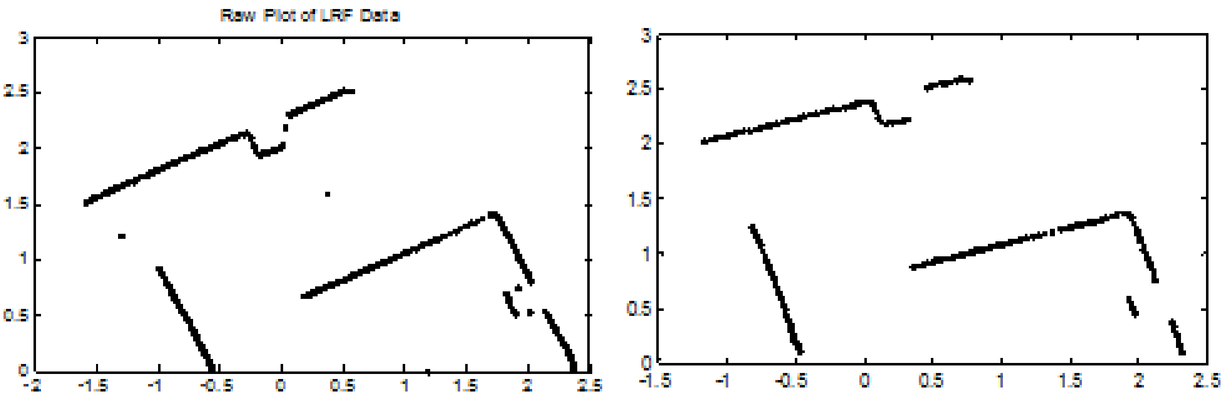


Figure 4.2: Experiment for Q and R calculation.



(a) Raw data before processing.

(b) Data after preprocessing.

Figure 4.3: Data from laser range finder.

Line fitting & feature extraction

Two types of line fitting approach were used for this experimental study.

- Scaled split and merge technique (SSM)
- Split only technique (SO)

Scaled split and merge technique (SSM): The data U containing the processed data from LRF is divided into subsets S_i, where i vary from 1 to N. The S_i is created such a way that it contains data points from U if the distance between two consecutive data points in U is less than or equal to (E_d) 30 mm. If the distance is greater than (E_d) 30mm then a new subset S_i is created with the same condition. A subset S_i is valid only if the number of continuous points in the set is at least (N_p)=5 , assuming the validity of a line to exist is ($15 \times N_p$)= 75 mm. Different S_i indicates a break in the continuous line segment which may suggest end of a wall, a door opening or invisibility due to angle of view of the environment by LRF. S_i is again divided into small subsets $W_{i,j}$ containing 5 points each and j varies from 1 to M. Normal line fitting is done with $W_{1,1}$ and the perpendicular distances of the points from $W_{1,2}$ is checked. If at least 3 points distance from set $W_{1,2}$ to the line from set N_p is less than or equal to the threshold T_d then the set $W_{1,1}$ and $W_{1,2}$ is combined to form a new line and this process is repeated with the successive sets. If the threshold condition T_d fails then an edge is detected and a new line fitting is started. The algorithm 4, 5 and 6 explains the line fitting approach in a much simpler way and a schematic of the same is shown in Figure 4.4.

Algorithm for scaled split and merge technique.

```
k=1,r=1
if $| U(r) + U(r+1) | \leq E_d$ then
    $S_i(k) = U(r)$
    k=k+1
end
else
    $S_i(k+1) = U(r-1)$
    k=1
    i=i+1
end
```

Algorithm 4: Algorithm to identify breaks in data set which signify the existence of doors or open space.

```
for k=1 to i do
    if length($S_k$) $< N_p$ then
        $S_k = NULL$
    end
    if length($S_k$) $> N_p$ then
        j=1
        for t=1 to length($S_k$) do
            $W_{k,j} = S_k(t : t + N_p - 1)$
            j=j+1
        end
      end
    end
```

Algorithm 5: Algorithm to identify edges or corners.

```
m = 1, k = 1, t = 1;
$L_{k,m} = linefit(W_k, t)$
for k=2 to i do
    for k=2 to i do
        if 50 % (| $L_1, W_{k,t}$ |) $\leq T_d$ then
            $W_{k,t} = W_{k,t-1} \cup W_{k,t}$
            $L_{k,m} = linefit(W_{k,t})$
        end
        else
            $L_{k,m} = linefit(W_{k,t})$
            m=m+1
        end
      end
  end
```

Algorithm 6: Algorithm for line fitting.

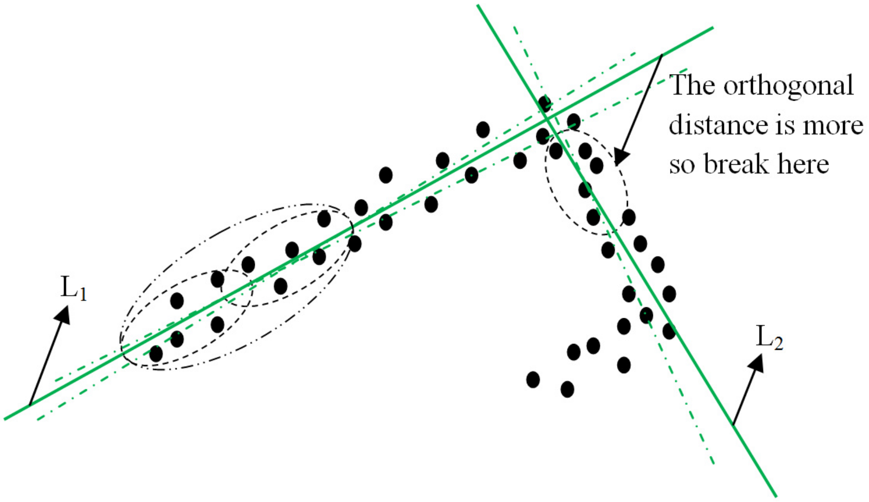


Figure 4.4: Schematic of scaled split and merge.

Split only technique (SO): The subset S_i is created in the same way as in the previous technique from set U. The end points in the subset S_i for $i = 1$ is joined by a straight line L_1 and the point from the same set which has the maximum deviation from the line is identified. If this deviation exceeds the allowable deviation T_d then S_1 is broken at this point into two subsets S_{11} and S_{12}. Straight line fitting is done for S_{11} by line L_{11}. The initial and final value of S_{12} is joined by a straight line L_{12} and the point from set S_{12} which has maximum orthogonal deviation is identified. Again S_{12} is broken into subsets S_{121} and S_{122}. Straight line fitting L_{121} is done for S_{121} and the procedure is repeated until all the possible lines are fitted in the set S_1. The same is carried out for all S_i for i varying from 1 to N. This is the simplest method in identifying straight lines with a set of points from LRF. The algorithm 7 is used for this approach and the schematic of the same is shown in Figure 7.

```
Initialize T$_d$
N = length(S$_k$)
L=linefit(S$_k$(1), $S_k(N)$)
d=max(| L + S$_k$|)
while $d \geq T_d$do
    $I_m = Identify\_the\_index\_for\_max(| L + S_k |)$
    break L into 2 lines at $I_m$
end
```

Algorithm 7: Algorithm for split only technique.

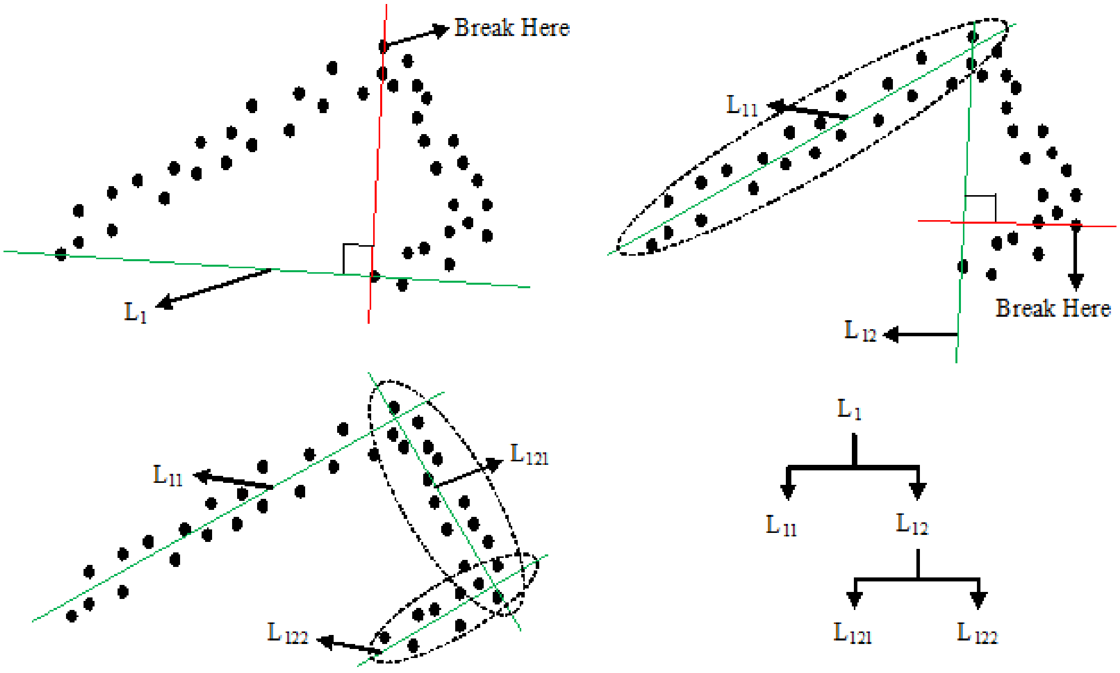


Figure 4.5: Schematic of split only technique.

Data association

The localization of a mobile robot (MR) either in a known or unknown environment is largely dependent on effective feature extraction and data association. The task of fixing a particular node in the environment as a feature and associating the features correspondences between scans taken at different spatial instances is very typical and challenging. So the data association problem can be identified as the information carried between two successive scans. Assuming the features identified till time instance k to be set $A_k = \{F_1, F_2, F_3, F_4, ...\}$ and at time instant k+1 to be $A_{k+1} = \{F_2, F_3, F_4, F_5\}$, then the information carried is $I_C(k+1|k) = A_k \cap A_{k+1} = \{F_2, F_3, F_4\}$. The complexity of the data association problem basically depends on the distribution of the features in the environment. For better understanding the distribution can be classified as spread distribution (SD) and cluttered distribution (CD). The SD poses less complexity as the features are largely distributed over space and any spurious data can be easily singled out. But in a CD, features may be placed closely and hence data association may not be unique between two scans. Any localization techniques which are self evolving in nature like Extended Kalman filter (EKF), Unscented Kalman Filter (UKF), and Particle Filters (PF) which largely depends on data association will diverge over time on wrong association. Recent work suggests techniques like Minimum Mahalanobis Distance (MMD) based Nearest Neighbor (NN), Joint Compatibility Test (JCT), Sequential Compatibility Nearest Neighbor (SCNN), Joint Compatibility Branch and Bound (JCBB), and neuro-fuzzy based data association.

The mathematical model for the MR and sensor is given by the equation 4.14 from where the rest of the implementation of EKF begins,

$$\begin{aligned} X_k &= f(X_{k-1}, U_{k-1}) + \omega_{k-1} \\ Y_k &= h(X_k) + \nu_k \end{aligned} \tag{4.14}$$

where X is the estimated state vector $X_k = [x_R, x_{F1}, x_{F2}, x_{Fn}]^T$, U the input vector and ÏL’ the system noise with covariance $Q(\sigma_\omega^2, 0)$ and mean zero. Y is the output vector and ν the measurement noise with covariance $R(\sigma_\omega^2, 0)$ and mean zero with ω and ν completely independent. It is very important to know how the state estimate propagates with respect to its mean and the dependency between the states. This relationship is represented using the error covariance matrix P_k. The prior error covariance P_k^- is given by equation 4.15.

$$P_k^- = A_J P_{k-1}^+ + A_J^T + Q \tag{4.15}$$

where P_k is of the form $P_k = \begin{bmatrix} P_R & P_{RF_1} & \cdots & P_{RF_n} \\ P_{F_1R} & P_{F_1} & & \vdots \\ \vdots & & \ddots & \vdots \\ P_{F_nR} & & \cdots & P_{F_n} \end{bmatrix}$ and A_J represents the Jacobian of the system model.

The Kalman gain is given by K_k=$P_k^- H_{J_i}^T S^{-1}$, where S given by equation 4.16 is the innovation covariance and H the measurement Jacobian. This innovation covariance is

$$S = H_{J_i} P_k^- H_{J_i}^T + R \tag{4.16}$$

This innovation covariance given by equation 4.16 is used to find the Mahalanobis distance through equation 4.17 for best joint compatibility.

$$C_k \subset D_k^2 = (Y - \hat{Y})^T S^{-1} (Y - \hat{Y}) < \chi_{d,\alpha}^2 \tag{4.17}$$

where $\hat{Y}$ is the measurement vector after observation from the sensor and C_k contains all the associated features which satisfies the chi-square hypothesis with α being the desired confidence level and d the degree of freedom of Y which is 2 in this case.

4.4 Implementation of EKF for SLAM

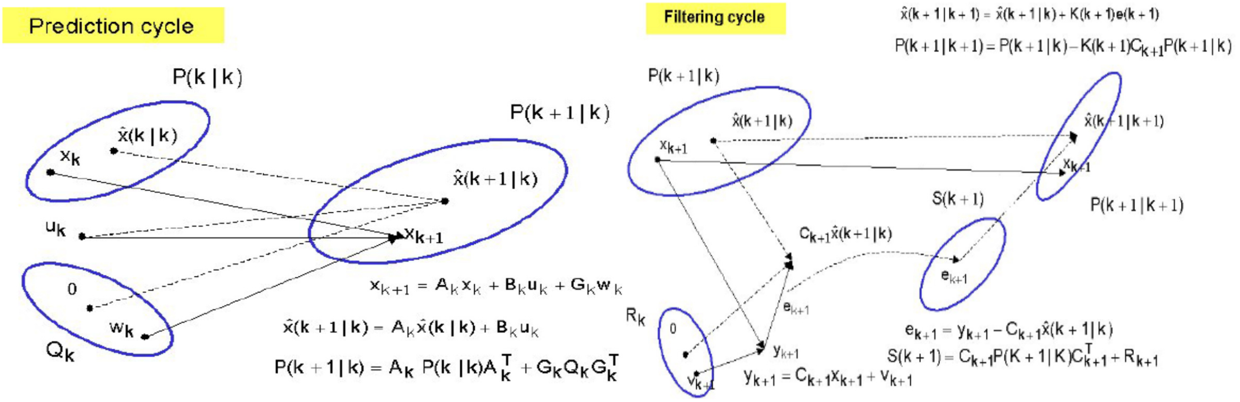
Using equation 4.16 the Kalman gain K_k is calculated using 4.18 and this K_k is derived for reflecting minimum cost function from the predicted states, current system model, sensor model and observed features.

$$K_k = P_k^- H_{J_i}^T (H_{J_i} P_k^- H_{J_i}^T + R)^{-1} \tag{4.18}$$

The estimated state from equation 4.1 is finally corrected using equation 4.19 and this is the most optimal estimate of the MR states given the observed feature positions.

$$X_k^+ = X_k^- + K_k(Y_k - h(X_k^-)) \tag{4.19}$$

X_k^+, signifies the posteriori state estimate of the MR at k_{th} instant, in the sense that it is an updated estimate of X_k^- after observation through vision sensors and EKF implementation. Y_k is the polar coordinates of the associated features between the current observation (k_{th} instant) and all the past identified features (1 to $(k-1)^{th}$ instances). $h(X_k^-)$, is the prediction of the distance and pose of the features identified till $(k-1)^{th}$ instant with respect to the estimated robots priori position. For computational reasons this prediction was done only for the associated features in this experiment. If no associated features are visible then this update step is skipped, full belief is given to the model and the visible features at k^{th} instant is accounted as new features. At least two associated features were recommended for the estimated state to be updated. If only one feature gets associated that update step was skipped after just adding any new features that was observed.


Prediction cycle
$P(k|k)$
$\hat{x}(k|k)$
x_k
$P(k+1|k)$
$\hat{x}(k+1|k)$
u_k
x_{k+1}
0
w_k
Q_k
$x_{k+1} = A_k x_k + B_k u_k + G_k w_k$
$\hat{x}(k+1|k) = A_k \hat{x}(k|k) + B_k u_k$
$P(k+1|k) = A_k P(k|k) A_k^T + G_k Q_k G_k^T$
Filtering cycle
$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k+1)e(k+1)$
$P(k+1|k+1) = P(k+1|k) - K(k+1)C_{k+1}P(k+1|k)$
$P(k+1|k)$
$\hat{x}(k+1|k)$
x_{k+1}
$\hat{x}(k+1|k+1)$
x_{k+1}
$S(k+1)$
$P(k+1|k+1)$
$C_{k+1}\hat{x}(k+1|k)$
e_{k+1}
R_k
0
e_{k+1}
y_{k+1}
v_{k+1}
$e_{k+1} = y_{k+1} - C_{k+1}\hat{x}(k+1|k)$
$S(k+1) = C_{k+1}P(K+1|K)C_{k+1}^T + R_{k+1}$
$y_{k+1} = C_{k+1}x_{k+1} + v_{k+1}$

(a) Prediction cycle.

(b) Filtering cycle.

Figure 4.6: Schematic block diagram.

Finally the posterior covariance P_k^+ is updated using K_k using equation **??**.

$$P_k^+ = (I - K_k H_{J_i}) P_k^- \tag{4.20}$$

Also an algorithmic flowchart summarizing the working of Extended Kalman Filter(EKF) is shown in Figure 4.7.

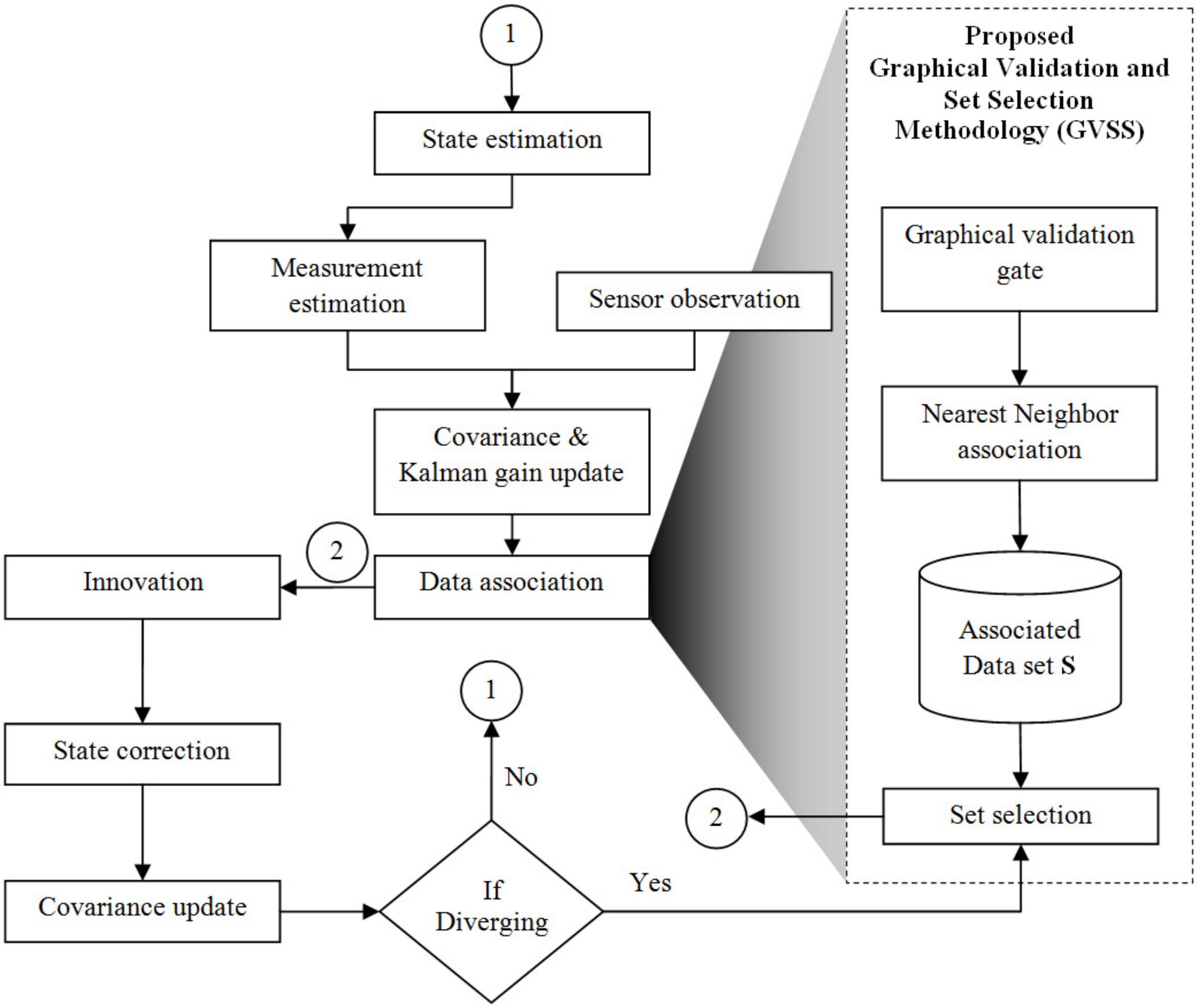


Figure 4.7: Algorithmic flowchart of Extended Kalman Filter.

4.5 Results and Discussions

The arena that was mapped using the EKF is shown in following figures. It is evident from the figure that the EKF has corrected the estimated state which otherwise would have followed the path indicated by red. Different arena was created for robust testing of the algorithm as depicted below in the following figures.

Data set 1

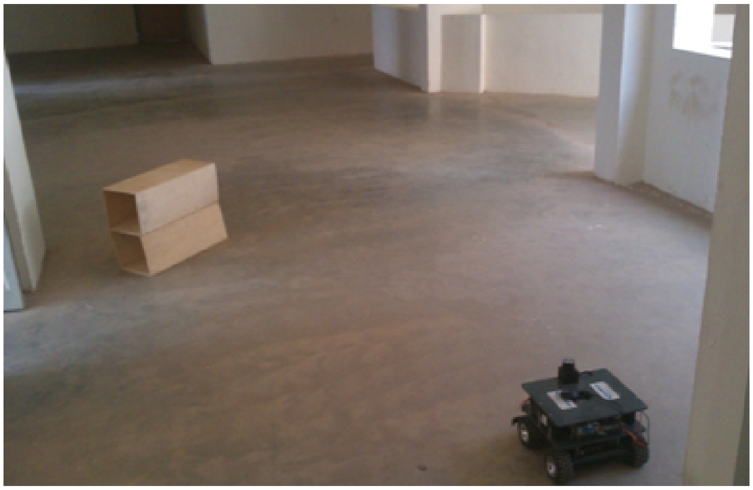


Figure 4.8: Picture of the arena recorded for data set 1.

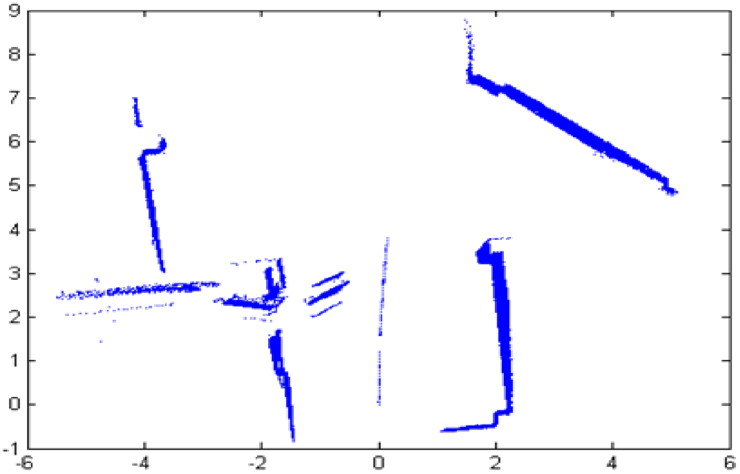


Figure 4.9: Global plot of range finder data using odometry.

Data set 2

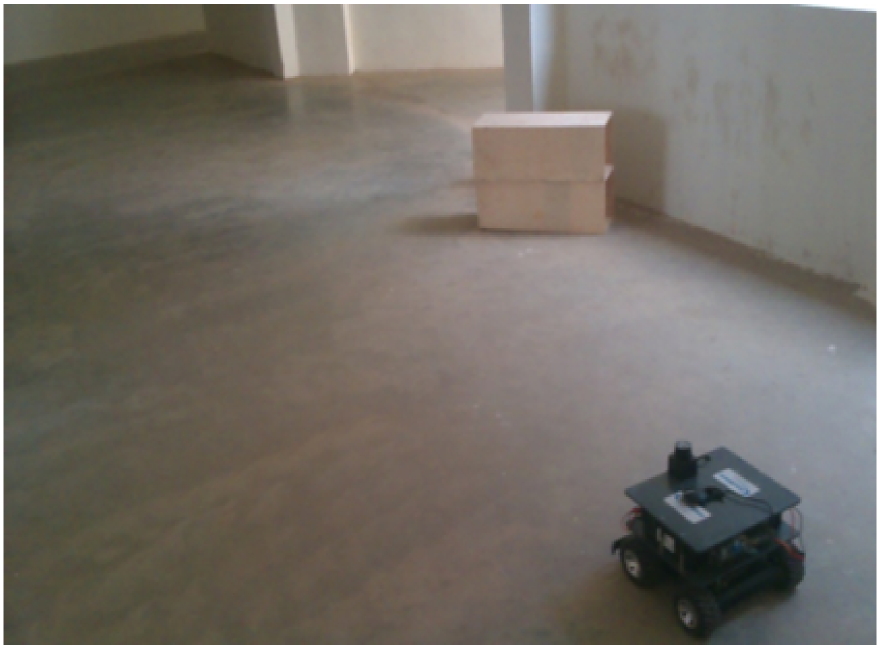


Figure 4.10: Picture of the arena recorded for data set 2.

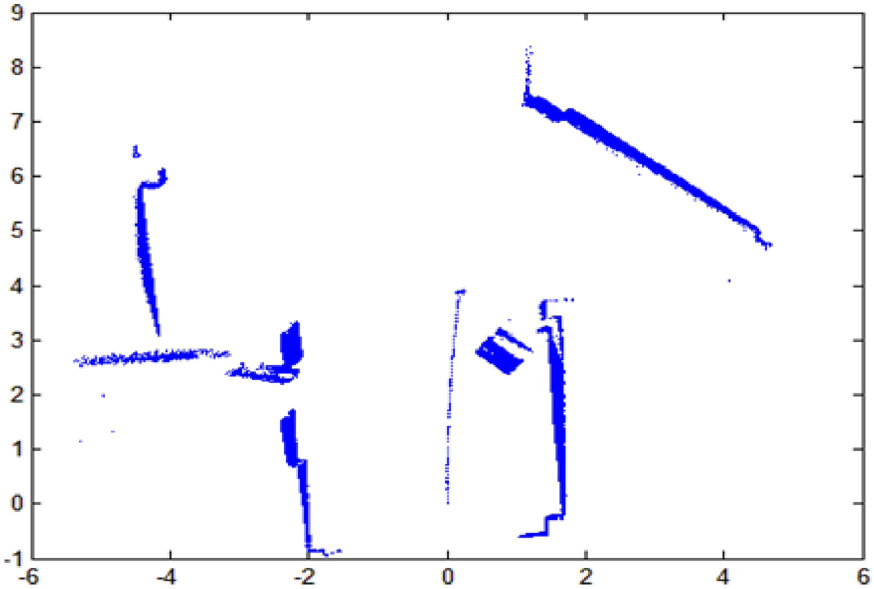
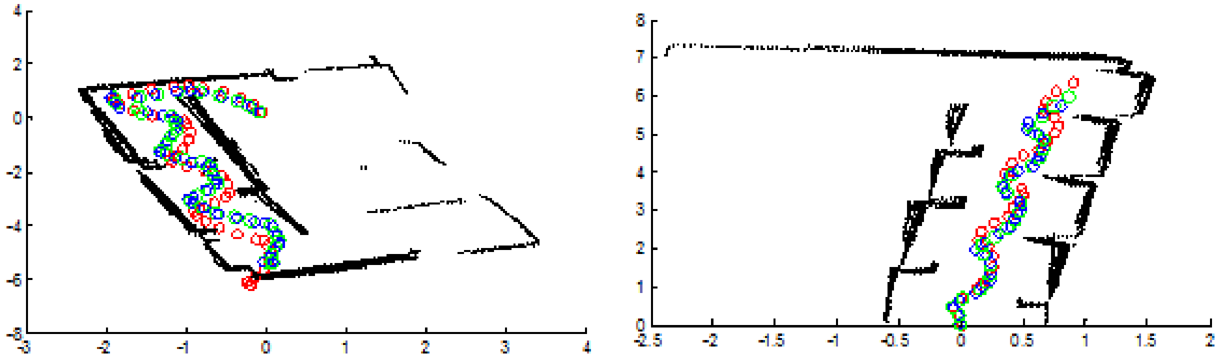


Figure 4.11: Global plot of range finder data using odometry.

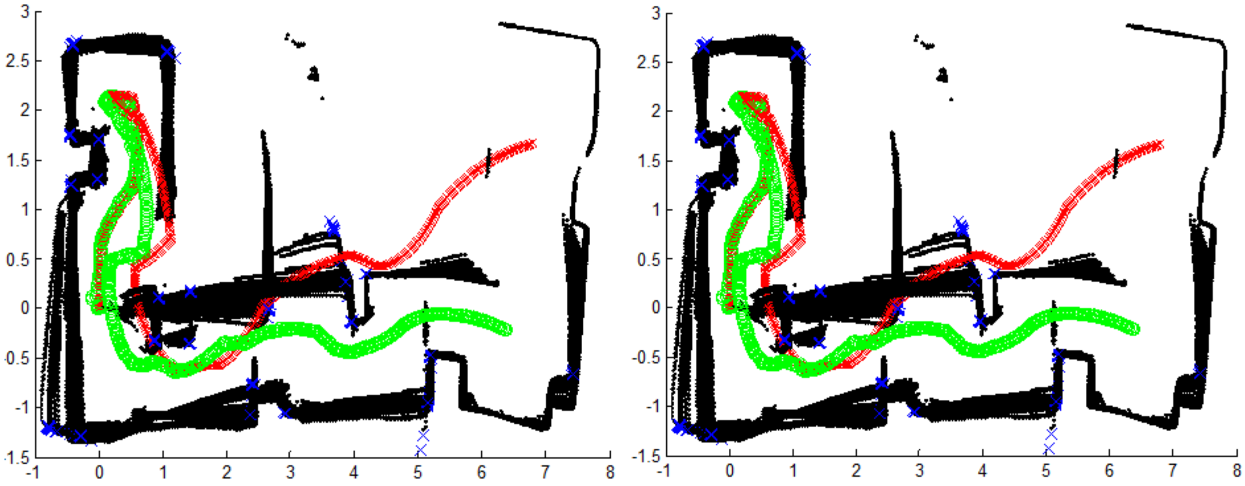


(a) Global map showing the raw path(red) and corrected path(green).

(b) Global map showing the raw path(red) and corrected path(green).

Figure 4.12: Global map of SLAM implementation

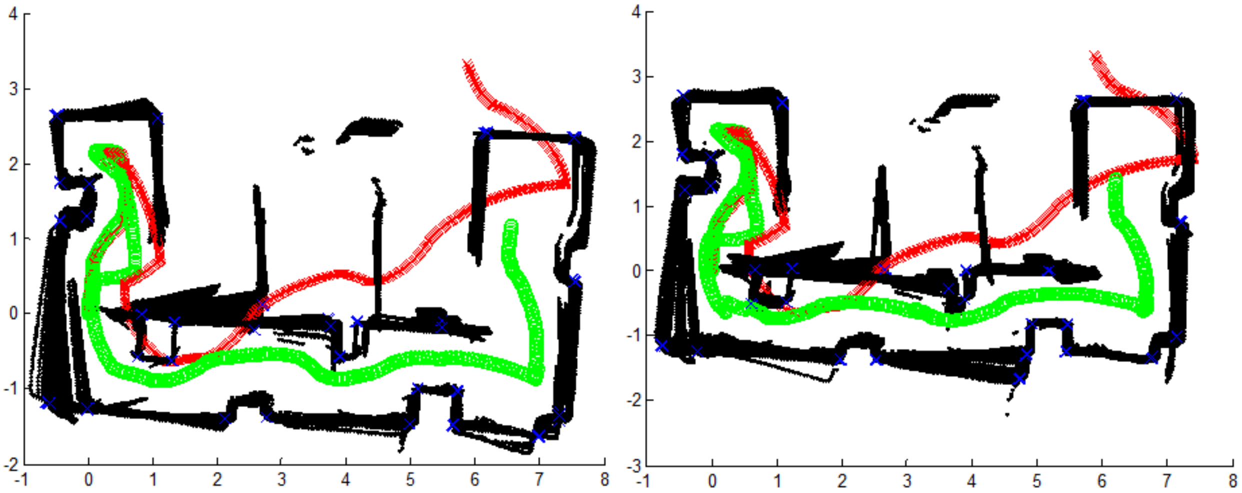
EKF implementation in various data sets of map of a maze.



(a) Maze data set 1.

(b) Maze data set 2.

Figure 4.13: EKF implementation in data set of a maze.



(a) Maze data set 3.

(b) Maze data set 4.

Figure 4.14: EKF implementation in data set of a maze.

Inferences

The real time implementation shown in Figure 4.13 and 4.14 demonstrates the effectiveness of the EKF algorithm, feature extraction program and data association in SLAM. The arena was created to have different kind of environmental features such as cluttered, sparse and distributed. The mobile robot proves to be highly efficient while navigating in all the environments with an accuracy of 30 cm. The red dots in the graph shows the actual predicted state of the mobile robot using the model 4.1 and 4.2. The green track shows the actual path traversed by the robot, which is the corrected state after implementing EKF. The features indicated by blue cross exhibits an accuracy of ± 2 cm, which shows the effectiveness of the data association algorithm. The implementation was tested in real time multiple times and the results showed a phenomenal performance in terms of state estimation and feature association. The intermediate loop closure problem was also addressed and it can be verified from the Figure 4.13 and 4.14 that the implementation is robust.

4.6 Recommendations for Future Work

- This algorithm can be extended for outdoor environment with other sensors such as GPS, camera and 3D LRF
- The feature extraction algorithm can be made more of environment adaptive.
- The failure rate of data association technique can be decreased with advanced tools.
- FPGA implementation of the same can be done to improve speed.

4.7 Papers Published and Communicated

1. K. Ramkumar, N. S. Manigandan, Stochastic Filters for Mobile Robot SLAM Problems - *A Review, Sensors Transducers*, 138(3), pp. 141-149, March 2012.

2. P. Prasanna, S. Sethuraman, N.S. Manigandan and K. Ramkumar, Fuzzy Based Obstacle Navigation and Effective Path Selection with Laser Range Finder, *Journal of Applied Sciences*, 14(14), pp. 1594-1599, 2014.

3. Manigandan Nagarajan Santhanakrishnan, John Bosco Balaguru Rayappan, Ramkumar Kannan, poXNOR Morphological Transform based Feature Extraction for Mobile Robot applications (communicated)

4. Manigandan Nagarajan Santhanakrishnan, John Bosco Balaguru Rayappan, Ramkumar Kannan, Repeatability of EKF based SLAM in a structured environment âĂŞ A real time implementation (to be communicated)

Bibliography

[1] J.A. Castellanos [27]. R. Martinez-Cantin. “Unscented SLAM for large-scale outdoor environments.” In: *Proc. of the 2005 IEEE Int. Conf. on Intelligent Robots and Systems.* RA - 3(3) (2005), pp. 3427–3432.

[2] K. Murphy S. Russel A. Doucet N. de Freitas. “Rao Blackwellized particle filtering for dynamic Bayesian networks.” In: *Inter. Proc. of the Conf. Uncertainty in Artificial Intelligence,UAI* (2000), pp. 176–186.

[3] R. Siegwart A. Martinelli N. Tomatis. “Simultaneous localization and odometry self calibration for mobile robot.” In: *Autonomous Robotics.* 22 (2007), pp. 75–85.

[4] M. Kaess F. Dellert. “Square Root SAM: Simultaneous localization and mapping via square root information smoothing.” In: *ICRA. Proceedings of the international conference on robotics.* 25 (2006), pp. 1181–1203.

[5] U. Frese. “A proof for the approximate sparsity of SLAM information matrices”. In: *Proc.IEEE International Conference on Robotics and Automation* (2005), pp. 331–337.

[6] J. Guivant. “Efficient simultaneous localization and mapping in large environments, PhD thesis, University of Sydney”. In: *Australian Centre for Field Robotics.* (2002).

[7] T. Bailey H.D Durrant-Whyte. “Simultaneous localization and mapping: Part I”. In: *IEEE Robot. Autom. Magazine* 13 (2006), pp. 99–110.

[8] A. Sanfeliu J. Andrade-Cetto T. Vidal-Calleja. “Unscented transformation of vehicle states in SLAM.” In: *Proc. of the 2005 IEEE Int. Conf. on Robotics and Automation* (2005), pp. 324–329.

[9] J. Neira J. Tardos J. Castellanos J. Montiel. “The SPmap: A probabilistic framework for simultaneous localization and map building”. In: *IEEE Transactions on Robotics and Automation* 15 (1996), pp. 984–953.

[10] J. Neira J.A. Castellanos J.D. Tardos. “Building a global map of the environment of a mobile robot: The importance of correlations.” In: *Proceedings of the 1997 Conference on Robotics and Automation, Albuquerque* (1997), pp. 1053–1059.

[11] H.F. Durrant-Whyte J.J. Leonard. “Directed Sonar Navigation.” In: *Kluwer Academic Press.* (1992).

[12] A.M. Shahri K. Shojaei. “Experimental study of Iterated Kalman filters for simultaneous localization and mapping for autonomous mobile robots”. In: *J. Intell. Robot. Syst* 63 (2001), pp. 575–594.

[13] R.E. Kalman. “A new approach to linear filtering and prediction problem”. In: *ASME Trans., J. of Basic Engg.* 82(D) (1960), pp. 35–45.

[14] M. D. Adams. L. D. L. Perera W. S. Wijesoma. “The estimation theoretic sensor bias correction problem in map aided localization.” In: *ICRA. Proceedings of the international conference on robotics.* 25 (2006), pp. 345–667.

[15] D. Koller B. Wegbreit M. Montemerlo S. Thrun. “Fa6stSLAM: A factored solution to the simultaneous localization and mapping problem.” In: *AAAI* (2002).

[16] D. Koller B. Wegbreit M. Montemerlo S. Thrun. “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”. In: *Proc. of Int. Conf. on Artificial Intelligence, IJCAI* (2003).

[17] J. Leonard M. Walter R. Eustice. “Exactly sparse extended information filters for feature-based SLAM”. In: *International Journal of Robotics Research.* 26 (2007), pp. 335–359.

[18] J.J. Leonard R. Eustice H. Singh. “Exactly sparse delayed-state filters”. In: *Proc. IEEE Int. Conf. on Robotics Automation (ICRA)* (2005), pp. 2428–2435.

[19] P. Cheeseman R. Smith M. Self. “Estimating uncertain spatial relationships in robotics”. In: *Springer Verlag, Autonomous Robot Vehicles* (1990), pp. 167–193.

[20] W.D. Renken. “Concurrent localization and map building for mobile robots using ultrasonic sensors”. In: *Proc. IEEE Int. Work-shop on Intelligent Robots and Systems (IROS).* 2 (1993), pp. 2192–2197.

[21] J.A. Castellanos R.M. Cantin. “Unscented SLAM for large-scale outdoor environments”. In: *Intelligent Robots and Systems, IROS 2005* (2005).

[22] J. Leonard R.M. Eustice H. Singh. “Exactly sparse delayed-state filters for view based SLAM.” In: *IEEE Transactions on Robotics* 22 (2006), pp. 1100–1114.

[23] D. Koller A.Y. Ng Z. Ghahramani H. Durrant-Whyte S. Thrun Y. Liu. “Simultaneous localization and mapping with sparse extended information filters”. In: *Int. Journal of Robotics Research.* 23 (2004), pp. 693–716.

[24] D. Koller A.Y. Ng S. Thrun Y. Liu. “Simultaneous localization and mapping with sparse extended information filters.” In: *Inter. Journal of Robotics Research* 23 (2004), pp. 693–716.

[25] J.K. Uhlmann S.J. Julier. “A counter example to the theory of simultaneous localization and map building.” In: *Proceedings of the 2001 IEEE International Conference on Robotics and Automation* (2001), pp. 4283–4243.

[26] H. Durrant-Whyte T. Bailey. “Simultaneous localization and mapping (SLAM): Part II State of the Art”. In: *IEEE Robot. Autom. Mag.* 13 (2006), pp. 108–117.

[27] M. D. Adams S. Challa W. S. Wijesoma L. D. L. Perera. "An analysis of the bias correction problem in simultaneous localization and mapping". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems.* (2005), pp. 747–752.

[28] J. Guo W. Zhou Ch. Zhao. "The study of improving Kalman filters family for nonlinear SLAM". In: *J. Intell. Robot. Syst* 56 (2009), pp. 543–564.

[29] G. Dissanayake Z. Wang S. Huang. "D-SLAM: A decoupled solution to simultaneous localization and mapping". In: *International Journal of Robotics Research* 26 (2007), pp. 187–204.

[30] G. Dissanayake Z. Wang S. Huang. "Tradeoffs in SLAM with sparse information filters." In: *Springer Tracts in Advanced Robotics.* (2008), pp. 339–348.