AN IMPLEMENTATION OF 3D SLAM WITH PLANAR SEGMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÇAĞRI TURUNÇ

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2012

Approval of the thesis:

AN IMPLEMENTATION OF 3D SLAM WITH PLANAR SEGMENTS

submitted by **ÇAĞRI TURUNÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. İlkay Ulusoy _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Kemal Leblebicioğlu _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. A. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Fatih Kamışlı _____
Electrical and Electronics Engineering Dept., METU

Faruk Mengüç, M.Sc. _____
Manager, ASELSAN

Date: 12.09.2012

I hereby declare that all information in this document has been obtained and resented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Çağrı Turunç

Signature :

ABSTRACT

AN IMPLEMENTATION OF 3D SLAM WITH PLANAR SEGMENTS

Turunç, Çağrı

M. S., Department of Electrical and Electronics Engineering

Supervisor : Assoc. Prof. Dr. İlkay Ulusoy

September 2012, 222 pages

Localization and mapping are vital capabilities for a mobile robot. These two capabilities strongly depend on each other and simultaneously executing both of these operations is called SLAM (Simultaneous Localization and Mapping). SLAM problem requires the environment to be represented with an abstract mapping model. It is possible to construct a map from point cloud of environment via scanner sensor systems. On the other hand, extracting higher level of features from point clouds and using these extracted features as an input for mapping system is also a possible solution for SLAM.

In this work, a 4D feature based EKF SLAM system is constructed and open form of equations of algorithm are presented. The algorithm is able to use center of mass and direction of features as input parameters and executes EKF SLAM via these parameters. Performance of 4D feature based EKF SLAM was examined and compared with 3D EKF SLAM via monte-carlo simulations. By this way; it is believed that, contribution of adding a direction vector to 3D features is investigated and illustrated via graphs of monte-carlo simulations.

At the second part of the work, a scanner sensor system with IR distance finder is designed and constructed. An algorithm was presented to extract planar features from data collected by sensor system. A noise model was proposed for output features of sensor and 4D EKF SLAM algorithm was executed via extracted features

of scanner system. By this way, performance of 4D EKF SLAM algorithm is tested with real sensor data and output results are compared with 3D features. So in this work, contribution of using 4D features instead of 3D ones was examined via comparing performance of 3D and 4D algorithms with simulation results and real sensor data.

Keywords: Feature Based EKF SLAM, Center Point and Direction SLAM, Plane Segment Extraction, Range Scanner Sensor Modeling

ÖZ

DÜZLEM KESİTLERİ İLE 3D SLAM UYGULAMASI

Turunç, Çağrı

YüksekLisans, ElektrikElektronikMühendisligiBölümü

TezYöneticisi: Doç. Dr. lkay Ulusoy

Eylül 2012, 222 sayfa

Kendini konumlandırma ve haritalama mobil robotların önemli özelliklerindendir. Bu iki özellik birbiriyle ciddi biçimde ilintilidir ve her ikisini birlikte çalıştırma işlemi Eş Zamanlı Konumlandırma ve Haritalama (SLAM) şeklinde adlandırılır. SLAM problem çevrenin soyut bir modelle haritalandırılmasını gerektirir. Tarayıcı sensor sistemleri tarafından oluşturulmuş bir nokta kümesi haritalama amacıyla kullanılabilir. Bununla birlikte nokta kümesinden daha üst seviyede işaretler çıkarılması ve bu işaretlerin haritalama sisteminde girdi olarak kullanılması da SLAM problemine çözüm olarak kullanılabilir.

Bu çalışmada 4B işaret tabanlı EKF SLAM sistemi oluşturulmuş ve sistemin denklemleri açık biçimleriyle verilmiştir. Yazılan algoritma nesnelerin kütle merkezi ve yön bilgisini girdi olarak kullanmakta ve bu şekilde EKF SLAM algoritmasını koşabilmektedir. 4B işaret tabanlı EKF SLAM algoritması detaylı biçimde incelenmiş ve monte-carlo analiz yöntemi kullanılarak 3B EKF SLAM algoritmalarıyla kıyaslanmıştır. Böylece 3B işaretlere yön bilgisi eklenmesinin ne tür bir gelişme sağladığı incelenmiş ve monte-carlo analizi grafikleriyle görselleştirilmiştir.

Çalışmanın ikinci bölümünde, bir tarayıcı sensör sistemi tasarlanmış ve üretilmiştir. Sistemin oluşturduğu nokta kümesinden düzlemsel işaret çekilmesi için bir algoritma sunulmuş, uygun bir hata modeli geliştirilmiş ve bu modelle sensörden alınan veri kullanılarak 4B EKF SLAM algoritması koşturulmuştur. Son adımda 4B EKF

SLAM algoritmasın gerçek veri ile çalıştırılarak sonuçları 3B algoritma sonuçlarıyla karşılaştırmalı olarak sunulmuştur. Böylece bu çalışmada, 4B işaret kullanılarak sağlanan gelişme simülasyon ortamı ve gerçek veri ile 3B sonuçlarla karşılaştırılarak incelenmiştir.

Anahtar Kelimeler: İşaret Tabanlı EKF SLAM, Orta Nokta ve Yön ile SLAM, Düzlem Kesitleri Çıkarma, Uzaklık Tarayıcı Sensör Modelleme.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr. İlkay Ulusoy for her guidance, advice, criticism, encouragement and insight throughout the completion of the thesis.

I am indebted to all of my friends and colleagues for their support and encouragements. I am also grateful to ASELSAN Inc. for the facilities that made my work easier.

Finally, I am grateful to my family for their continuous support and encouragements.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Problem Definition and Motivation

Using autonomous or remote controlled systems can remove humans from dangerous conditions; accomplish operations in places that humans cannot even survive and decrease the cost or rate of the defects in several processes. This property makes the remote systems a good choice for industry, defense systems, aerospace systems, underground and underwater applications. Efficient use of mobile agents needs them to be localized in the environment that they are used. For the cases that the map of the environment is unknown, a remote system must also be able to construct the map of its environment, using its sensors.

Building the map of an unknown environment and localizing a robot in a known map are two fundamental issues in mobile robotics. Looking the solution of both of these two issues is called Simultaneous Localization and Mapping, in short SLAM. To solve this problem the robot must incrementally construct the map of the environment while localizing itself in this recently constructed map. In other words, SLAM problem asks if it is possible to construct a consistent map of the environment with iterative methods while localizing the place of the robot in this semi-constructed map.

SLAM can be used both at indoor and outdoor environments. Constructing the map of a structured environment -such as the interior of a building- is a good example for indoor SLAM. Such application can be used in different areas, for example

Minerva[1] is an autonomous robot that is used as a tour guide in American History Museum. This kind of autonomous robots can be used to construct the map of buildings. Besides the usage of tour guiding, SLAM can be used in different kinds of constructed or unconstructed closed environments, which are dangerous for humans. Building the maps of mines, caves, sewer systems; searching damaged constructions or places that humans cannot survive can be achieved by autonomous mobile robots. In addition, mobile robots can be used to map outdoor areas. Exploration of outdoor terrain with autonomous land or air vehicles and underwater applications of SLAM are used in several areas.

Various types of solutions for SLAM problem are proposed throughout the last decades. Each SLAM approach has advantages and disadvantages for different kinds of sensors and environmental conditions. Different equations, algorithms and sensors are used for different requirements.

Probabilistic methods are widely used in SLAM. Probabilistic robotics is relatively a new approach to robotics which pays tribute to uncertainty in the placement and actions of robot. The main idea in probabilistic robotics is to represent the uncertainty explicitly using mathematical functions.[2] In probabilistic perception, the belief of positions of surroundings and the belief of the agent are not kept as a single point existence, but they are kept as probability distribution functions the in whole space.

This thesis presents an EKF SLAM algorithm which uses center of mass and direction information as feature parameters. Point clouds are processed to extract planar features, just like done in many other works about feature based EKF SLAM. In addition to this, another EKF SLAM algorithm which uses 3D point landmarks is presented. Each algorithm is executed both in simulation environment and with real data. By this way several analysis are presented for 3D EKF SLAM and contribution of adding a direction vector to features is emphasized. A custom range scanner system –named IRSCAN- is designed and fully constructed for this work. The experiments in real environment are conducted with IRSCAN, thus the thesis

presents feature extraction method and a noise model for this custom scanner system. The following two sections (Section 1.2 – Section 1.3) provide deeper information about the state of the art and contributions of this thesis.

1.2 Literature Survey

State-of-the-art approaches for metric localization and mapping are probabilistic methods, explicitly considering uncertainty information modeling sensor noise and imperfections in robot motion. The most important representations used are grid-based, raw data based or feature-based methods.[3]

Grid based methods, represent the environment with metric grids and assign a number to each grid that represents the probability of being filled or not. Since grids directly represent occupancy information, they are known useful methods for navigation operations. In their work, Burgard and Thrun[4] proposed a typical solution for grid based mapping method. They constructed a tour guiding agent with 1 DoF sonar scanner, which constructs 2D grid maps of close door environments. When the agent needs to localize itself in a relatively large environment, grid based methods suffer from complexity of *cross-correlation search* algorithms. In addition, if the cross-correlation result is multi-model (has more than one maximum point) a maximum likelihood search may converge to wrong local maxima. These two problems might be addressed by employing a particle based localization method in the grid map.[5] A multi robot grid based mapping SLAM implementation is proposed by Birk[6] and mapping of an indoor environment with six mobile robots is presented in his work. One of the most significant weaknesses of grid based mapping SLAM is cycle detection. For large the cycles, the minimal search-space may become too great for real-time cross-correlation; on the other hand using a tight search space may lead the agent to fail detecting the cycles. Additionally, given a large search-space for scan matching, the possibility of multiple correlation modes is high, and data association failure becomes increasingly likely.[7]

The majority of 3D SLAM approaches are implemented with raw data based scan alignment method related to *Iterative Closest Point* algorithm. It has the advantage that it solves the data association problem automatically as it directly tries to find corresponding points in the raw data.[7] Such method needs no feature extraction algorithm since it directly uses scanner data. In his work Surmann[8] created a precise 3D map of an indoor environment using a 2 DoF laser scanner system integrated with a mobile agent. Similarly Nüchter[9] implemented a variant of *Iterative Closest Point* method to build a mine mapping robot, which use raw sensor data with a 6 degrees of freedom agent. However due to its iterative nature the above methods are relatively slow and may lead inconsistent maps for cycle detection.[7]

For the feature based methods the assumption is made the environment can be modeled by abstract geometric features. Due to the small quantity of parametric data required to represent these features, the resulting maps are compact and the associated algorithms are efficient in comparison to other approaches. However such methods needs a data processing algorithm that extract features and associate them with landmarks in the map.[3] The Extended Kalman Filter (EKF) is a widely used estimation tool applicable to feature based methods. A drawback of EKF based SLAM is its restriction to the Gaussian noise model, but Arras[10] proposed a multi hypothesis EKF SLAM model which allows tracking more than one Gaussian noise model in the same map. In addition to this, Montemerlo[11] presented a solution called FAST SLAM, which integrates particle filter to SLAM problem. By this way, position tracking localization and global localization methods are combined to provide stronger localization and mapping techniques.

Several kinds of features can be used for mapping of the environment. Scanner based sensor systems creates a point cloud of the environment and different kinds of features can be extracted from this point cloud. Guivant[12] used a laser scanner sensor to create point features of the interesting points. Lemaire[13] used a monocular camera to extract line segments and used these segments as features to represent indoor environment. Different kinds of features can already be used

depending on the task of the agent. For example Guivant [12] use trees as features to create the map of a park. Sunghwan[14] use center points of objects as features. In addition to that different kinds of geometric shapes such as cylinders or spheres can also be used as features to represent the map.[15] For example Berger[16] use a similar method used in this work and they represent the environment with plane segments which are integration of a 3D point and three normal vectors. In his work, Newman[17] represent the environment with planes via representing the uncertainty with SP-model. For visual SLAM Gee[18] collects visual point landmarks and to obtain a higher level structure, he place the landmarks on a plane which is represented by a 9D state vector that is composed of an origin and two vectors that lies on the plane. Similarly Kwon[19] used a mono camera system and added a normal vector to point landmarks to execute SLAM. Finally Weingarten[3] presented an influencing example of utilization of 3D features that, 3D infinite planes and 3D planar segments are used as features via SP-model for feature based 3D EKF SLAM. When taking the step from 2D to 3D what was a line in 2D often becomes a plane in 3D.[20] Actually using higher level of features for EKF SLAM is a straight forward extension of 2D EKF SLAM, but added significant complexity due to the vehicle and mapping models.[21]

1.3 Thesis Contribution

The major contributions of this thesis are related with constructing and executing an EKF SLAM algorithm with 3D planar features, constructing a scanner based sensor system named IRSCAN, presenting an appropriate feature extraction algorithm and a valid noise model for IRSCAN and executing EKF SLAM algorithm via data collected by this sensor system. By this way it is believed that a mathematical solution to SLAM problem is developed similar to the solutions based on Extended Kalman Filter in the literature. The relation between parameters of SLAM and algorithm performance is analyzed and illustrated with graphical representations. Two algorithms are coded and compared for EKF SLAM. The first algorithm uses direction vectors in addition to 3D center of mass information, while the latter one does not use any direction vector or plane orientation information. Since both

algorithms run in similar simulation environment, the contribution of adding a direction vector to EKF SLAM is examined via monte-carlo analysis.

As stated previously a custom scanner based sensor system named IRSCAN is constructed for this work. A feature extraction algorithm and a valid noise model are presented for IRSCAN and the extracted features are used for real experiments. Since IRSCAN provides a noisy point cloud relative to the other laser scanner systems, the thesis also provides a solution for 3D planar feature extraction and EKF SLAM implementation with a noisy sensor system. The principal contributions of the thesis are as follows:

- An IR scanner system that –named IRSCAN- constructs distance image and point cloud of the surroundings is designed, fully constructed and a noise model that can be used in EKF SLAM is presented for the sensor system.
- Widely used edge detection methods were executed (via MatLab) on the data collected by IRSCAN. By this way, points belonging to the surface of different objects are decomposed from other points in the cloud. A plane extraction algorithm was executed on the data produced by IRSCAN to extract planes from the decomposed data. Additionally, a corner detection method using edges of objects is presented for IRSCAN. By this way IRSCAN is able to provide data for feature based 3D EKF SLAM with planar segments and corner point features.
- A 3D feature based EKF SLAM algorithm that uses center of mass and orientation of planar features as landmarks is presented. The algorithm is able to run both in simulation environment and with real sensor data. The presented algorithm is an extension of feature based EKF SLAM algorithm presented by Thrun[2]. To execute such algorithm, a direction vector is added to 3D point features and open forms of EKF SLAM algorithm is presented to clarify the method used for this work.

- Another EKF SLAM algorithm that uses 3D positions of the point features is presented. This algorithm is also able to run both in simulation environment and with real sensor data.
- Two methods of EKF SLAM stated above (using plane and point features) are executed with real data. To achieve this, data collected by IRSCAN is processed to the extract features from the environment. The resulting parameters of the features are inserted into EKF SLAM algorithm with the presented sensor noise model. Finally the algorithm is executed for each data input and the map of the environment is constructed. By this way, it is believed that, mechanical, electronics and software design of IRSCAN; the presented model of sensor noise for IRSAN; point cloud construction and feature extraction algorithms and finally the MATLAB code written for EKF SLAM implementation is tested in real environment. Resulting outputs of stated process is presented with background truth.
- A simulation system, that runs the algorithms indicated above, is constructed. In the simulation system, variance of control parameters of the agent, variance of the sensor system parameters of the agent, environmental conditions such as the size of the map, number or type of the landmarks can be altered; skills of agent such as scanner range, scanner field of view, and locomotion speed can be adjusted. The simulation can be executed manually for each set of parameters or able to run monte-carlo analysis for different sets of parameters. Resulting outputs of the monte-carlo runs can be recorded and graphed via MATLAB.
- Several monte-carlo simulations for different set of parameters were executed and the output results of simulations are illustrated. It is believed that, output graphs of monte-carlo runs provide valuable information to examine effect of parameters of the agent on the performance of EKF SLAM.

CHAPTER 2

THEORETICAL BACKGROUND

2.1 Extended Kalman Filter

R. E. Kalman published his famous paper in 1960. In his paper he described a recursive solution to discrete-data linear filtering problem. Since that time, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. The Kalman filter provides mathematical equations for computational solution of the least-squares method. The filter is powerful since it supports estimations of past and present states. The Kalman filter is a set of mathematical equations that provides an efficient computational solution. It can do so, even when the precise nature of the modeled system is unknown.[22]

The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process, which is governed by the linear stochastic difference equation;

$$x_{k+1} = A_k x_k + B u_k + w_k \quad \text{2-1}$$

with a measurement equation;

$$z_k = H_k x_k + v_k \quad \text{2-2}$$

The random variables w_k and v_k represent the process and the measurement noise. They are independent, white, and normal probability distributions.

The $n\ x\ n$ matrix A, relates the state at time step k to the state at $k+1$. The $n\ x\ 1$ matrix B relates the control input to the state x. The $m\ x\ n$ matrix H in the measurement equation relates the state to the measurement z_k.[22]

In short, the Kalman filter is a set of mathematical equations that updates a Gaussian distributed probabilistic state to another Gaussian distributed state. The filter updates most possible position and distribution parameters for each step. In a localization algorithm the filter does such operation in two recursive steps: *time update* and *measurement update*.

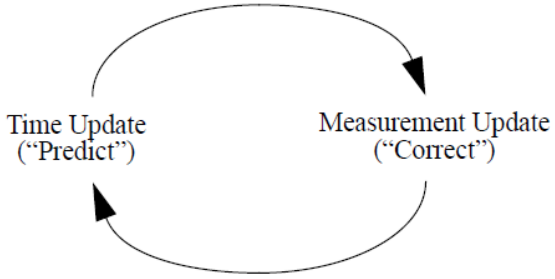


Figure 1 – Diagram representation of Kalman filter

Equations of time update part are given below;

$$\hat{x}_{k+1}^{-} = A_k \hat{x}_k + B u_k \quad \textbf{2-3}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + Q_k \quad \textbf{2-4}$$

Equations of measurement update part are given below;

$$K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + R_k)^{-1} \quad \textbf{2-5}$$

$$\hat{x}_k = \hat{x}_k^{-} + K_k (z_k - H_k \hat{x}_k^{-}) \quad \textbf{2-6}$$

$$P_k = (I - K_k H_k) P_k^{-} \quad \textbf{2-7}$$

With a very friendly explanation (2-3) is update of maximum likely point of state, relative to control input. This step is linear addition of previous state and change in the present state. By this way, new maximum likely point is found for given control input.

When the state is changed relative to the control input, uncertainty of state is expected to increase because of added uncertainty from control input. (2-4) is the update step comes from uncertainty of control input. It is a linear operation that sums previous uncertainty and additional uncertainty.

(2-5) is Kalman Gain which represents the relation of variance between previous step and the next step in terms of the measurement noise. This relation will be used in the next equations.

By (2-6) the final pose of the maximum likely points of the state is found and (2-7) updates uncertainty of the final pose. By this way, given the control and measurement inputs; Kalman Filter converts previous state and previous uncertainty to a new state and a new uncertainty.

The Kalman filter updates states via linear equations. However, the relation between the new state and the control input or the measurement input may be nonlinear. Actually for this work, the relation between state and inputs are nonlinear for some parts of the equations. So a nonlinear form of the Kalman filter is needed to be used.

An extension of the Kalman filter is called Extended Kalman Filter (EKF), which can be applied to nonlinear set of equations. EKF linearizes a set of nonlinear equations via Jacobian's.

Let *f* and *h* be nonlinear function that shows the following relation:

$$\hat{x}_{k+1} = f(\hat{x}_k, u_k) \quad \textbf{2-8}$$

$$\hat{z}_k = h(\hat{x}_k) \quad \textbf{2-9}$$

By Jacobians;

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}} \quad \textbf{2-10}$$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}} \quad \textbf{2-11}$$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}} \quad \textbf{2-12}$$

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}} \quad \textbf{2-13}$$

Here; $w_{[j]}$ and $v_{[j]}$ are used to represent the effect of control and measurement noise. The Kalman filter equations can be updated as follows.

$$\hat{x}_{k+1}^{-} = f(\hat{x}_k, u_k) \quad \textbf{2-14}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + W_k Q_k W_k^T \quad \textbf{2-15}$$

Equations of the measurement update are as follows;

$$K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + V_k R_k V_k^T)^{-1} \quad \textbf{2-16}$$

$$\hat{x}_k = \hat{x}_k^{-} + K_k(z_k - h(\hat{x}_k^{-})) \quad \textbf{2-17}$$

$$P_k = (I - K_k H_k) P_k^{-} \quad \textbf{2-18}$$

(2-10) is a linearization of the transition between the previous state and the next state.

(2-11) is a linearization of the effect of the control input and the next state. It is used to add the effect of control noise.

(2-12) represents a linearization of the transition between the measurement and previous states.

Finally (2-13) is a linearization of the transition between the measurement and next state.

By this way all nonlinear equations are transformed to linear ones and Kalman Filter as shown in (2-14) – (2-18) can be executed.

Mathematical functions shown (2-8) – (2-18) will be extensively used in this work. Given probabilistic control input for the locomotion system of the agent and measurements of the sensor system; above equations provide strong transformation between previous pose of the robot and the map to next pose of the robot and the map. The reader shall see work of Welch[22] for more detailed information about derivation and application of EKF.

2.2 Probabilistic SLAM

SLAM is a process that, a mobile agent can build the map of its surroundings while localizing its pose in this map. An illustration of SLAM simulation is given below:

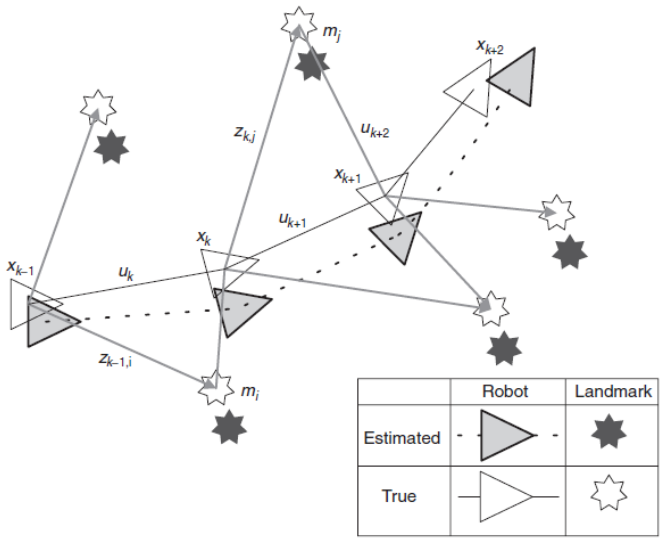


Figure 2 – An illustration of an agent in SLAM simulation. Steps of the agent is superposed. Figure from [23]

In Figure 2;

- $\mathbf{x_k}$: the state describing the pose (both location and orientation)
- $\mathbf{u_k}$: the control vector, applied at time k – 1. Drives the vehicle from x_{k-1} to x_k
- $\mathbf{m_i}$: the actual location of landmark i
- $\mathbf{z_{ik}}$: observation vector taken from the vehicle of the location of the i^{th} landmark at time k.

In explicit form the following vectors may also needed to be used:

- $\mathbf{X_{0:k}} = \{x_0, x_1, x_2, \dots, x_k\}$

  Used to describe the history of vehicle locations
- $\mathbf{U_{0:k}} = \{u_1, u_2, u_3, \dots, u_k\}$

  Used to describe history of control inputs
- $\mathbf{m} = \{m_1, m_2, m_3, \dots, m_n\}$

  Used to describe the pose of all landmarks
- $\mathbf{Z_{0:k}} = \{z_{11}, z_{21}, z_{31}, \dots, z_{n1}, z_{12}, z_{22}, z_{32}, \dots, z_{n2}, \dots, z_{1k}, z_{2k}, z_{3k}, \dots, z_{nk}\}$

  Used to describe the history of all observations for all landmarks

As the reader can verify from Figure 2, there is a gap between the real and the belief pose of the landmarks and the agent. Probabilistic SLAM must be able to deal with such erroneous mis-posed landmarks and agent.

In probabilistic SLAM problem, the main task of the system is to calculate

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \qquad \textbf{2-19}$$

for each *k* times.[7] In other words, probabilistic SLAM is a process that calculates the pose of the robot (x_k) and the landmarks (m) via sensor readings from surroundings ($Z_{0:k}$), control/sensor inputs of locomotion system of the robot ($U_{0:k}$) and of course initial position of the robot (x_0). For each step of robot and for each sensor reading, the pose of the robot (x_k) and pose of landmarks (m) shall be updated. Throughout this process, inputs of the system are sensor readings and

control input (and of course initial position). The outputs of the system are belief of the robot pose, belief of the landmark poses.

As it can be verified by the reader, (2-19) has two outputs: robot pose (x_k) and landmark poses (m). The first output x_k, is the solution of localization problem and the latter one m is the solution of the mapping problem. On the next chapters, these two problems will be discussed.

2.3 Localization

SLAM algorithm is mainly composed of two parts: Localization and Mapping. Mobile robot localization is the problem of determining the pose of a robot relative to a given map of its surroundings.

This task can be thought as a coordinate transformation operation between the map and the agent. The map is a global frame which is independent from the pose of robot. Localization is the process of establishing correspondence between the map coordinate system and the robot's local coordinate system. Knowing this coordinate transformation enables the robot to express the location of objects of interest within its own coordinate frame.[2] In many cases knowing the relative position $(x, y, z)^T$ and orientation *(pitch, yaw, roll)* angles is enough to find coordinate transformation parameters. Actually localization task itself is finding these position and the orientation transfer parameters relative to global map. Once these are obtained, all other parameters of transformation matrixes can be calculated via simple transformation operations. A graphical representation of localization problem for EKF SLAM is illustrated bellow.

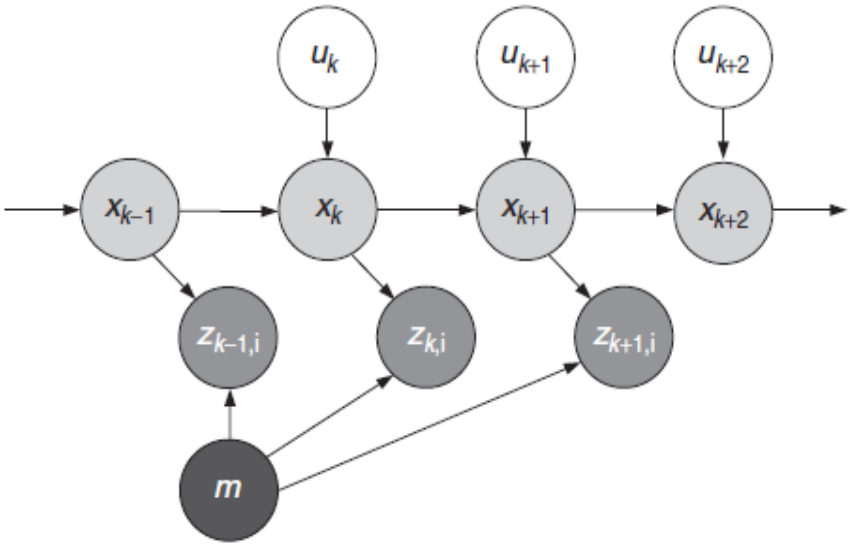


Figure 3 – Graphical representation of ekf localization. Figure taken from [2]

There are two main approaches for localization; incremental localization and global localization. Former one assumes that the initial position is known. The robot position is then updated using the measurements, while the robot is navigating in the environment. This type of localization is also called “position tracking”.

Global positioning problem is more complicated compared to the former one since the initial pose is unknown. In this approach the localization has to be done by running a localization algorithm that covers the entire map. Although the latter approach does not require the assumption of a pre-known initial position, it has high memory and processing requirements. Both of these approaches require the map of the environment to be known. Throughout this work incremental location approach will be used to execute EKF SLAM.

Some well-known solutions for localization problem will be discussed in the following sections.

2.3.1 Dead Reckoning Localization

Dead reckoning is the simplest localization technique. It use motion model and control inputs of the agent and decide a new location for the robot. Even though it is easy to execute this localization technique in an algorithm; Dead Reckoning is known as useless. Because, it does not provide any feedback for locomotion error. So accumulating errors will possibly lead the algorithm to diverge in time.

Rugged surface, chaotic structure of terrain, uncertainty in vehicle dynamics, wheel slip, skidding, sliding, disparity of wheel sizes and model biases etc. lead error in belief of vehicle position and direction. Error rate can significantly be degraded via an Inertial Navigation System support (INS), however accumulating behavior of errors and low frequency faults of motion models cannot be prevented.[2] Due to these reasons Dead Reckoning is not a useful option in probabilistic robotics.

2.3.2 Markov Localization

The key idea of Markov localization is to compute a probability distribution over all possible locations in the environment. $Bel(L_t = l)$ denotes the robot's belief of being at position l at time t. $Bel(L_0)$ represents the initial state of knowledge: if the robot knows its starting position, $Bel(L_0)$ is centered on the correct location as an impulse; if the robot has not any knowledge about its initial location, $Bel(L_0)$ is uniformly distributed to reflect the global uncertainty of the robot (latter case is illustrated at Figure 4).

The belief Bel is updated when one of the following two actions occurs. First, when the agent moves. The robot motion is modeled by conditional probability, denoted by $p_a(l|l')$. $p_a(l|l')$ is used to update the belief upon robot motion, where $Bel(L_t = l)$ denotes the resulting belief at time t:

$$Bel(L_t = l) \leftarrow \sum_{l'} p_a(l \mid l')\, Bel(L_{t-1} = l') \qquad \text{2-20}$$

Here, $p_a(l \mid l')$ must be obtained from the model of the robot's kinematics.

The second case of update occurs when the agent use its sensors. Let s denote a sensor reading and $p_a(s \mid l)$ is the likelihood of perceiving s at position l. When sensing s, the belief is updated according to the following rule:

$$Bel\,(L_t = l) \leftarrow \frac{p(s \mid l)\; Bel'(L_t = l)}{p(s)} \qquad \textbf{2-21}$$

Here $p(s)$ is the normalizer that ensures that the integral of belief all over the space is 1.[24]

An illustration of Markov Localization for a 1D environment can be seen bellow:

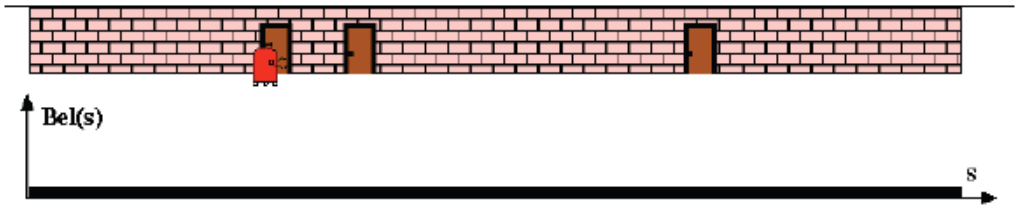


Figure 4 – Markov localization. Robot pose is completely unknown.

The figure above shows the belief of a robot at initial position. The map of the surroundings is given to the robot for this case and the three doors are the only landmarks of this map. Since initial state is unknown, probability density function (pdf) of robot position is uniform through the map. Whenever the robot starts to move and collects data the graph is updated as shown below:

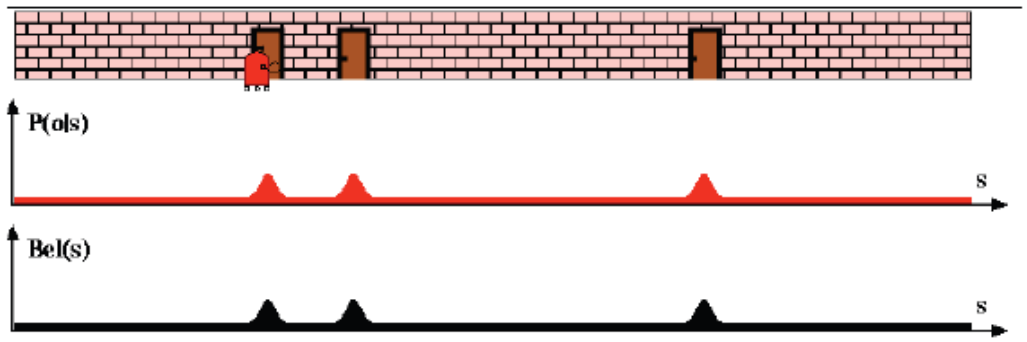


Figure 5 – Markov localization. Robot senses one of the doors.

As stated in (2-21) new belief of robot position is multiplication of previous belief and the belief comes from the observation. After the sensing step, the robot keeps moving as follows:

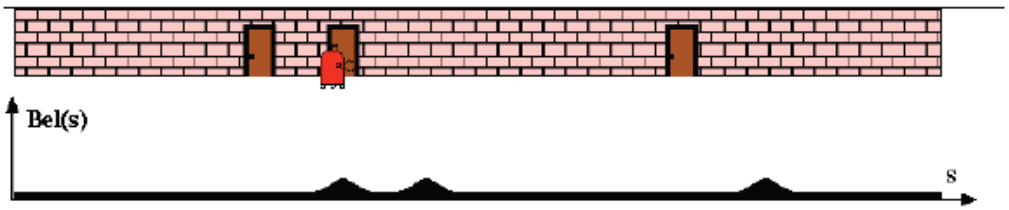


Figure 6 -Markov Localization. Uncertainity increases as the agent moves.

As the robot moves, belief of robot's position graph also moves. However at this step (2-20) holds and certainty of belief of position decreases because of the noise of the robot's locomotion system. Uncertainty will increase until it senses a new door:

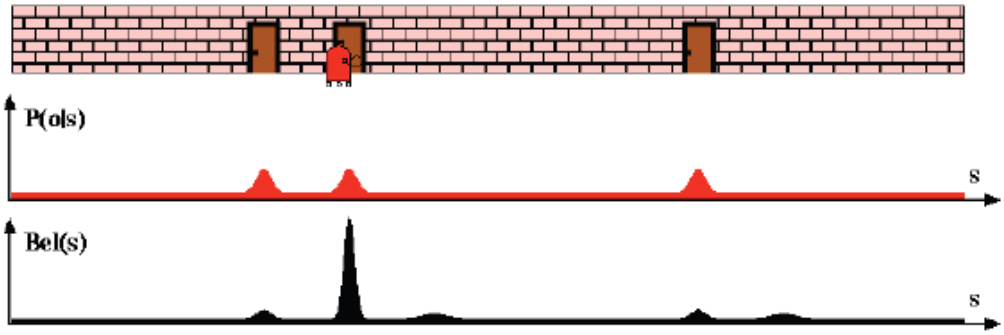


Figure 7 – Markov localization – When the agent senses the second door.

Just when the robot senses the new door, the belief will be updated again according to (2-21). Final belief will be updated as shown in Figure 7.[25]

2.3.3 Extended Kalman Filter Localization

Extended Kalman Filter Localization (EKF Localization) is a special variant of Markov Localization. This localization technique is used to find the position and the orientation of the agent in a given map. EKF localization represents the belief of pose with a Gaussian function in which the mean of belief is μ_t and covariance of belief is Σ_t. In other words, instead of several types of graphs, which are harder to parameterize, a set of mean and covariance matrixes are used in EKF localization. Equations used in EKF localization are similar to the ones used in Markov localization. However forms of pdf used in these equations are restricted with Gaussian function with representation $N(\mu_t, \Sigma_t)$.

An illustration of EKF localization is as follows:

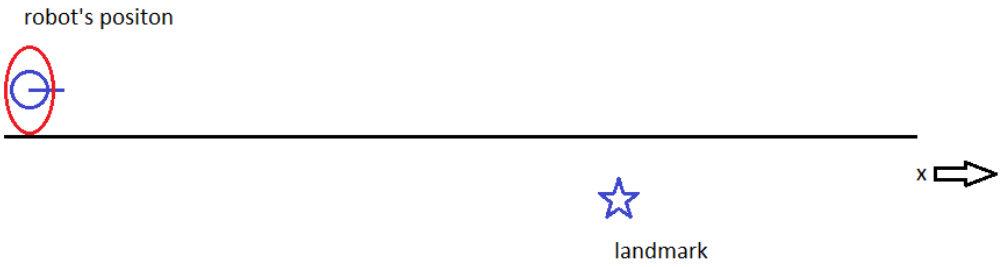


Figure 8 – EKF Localization. Initial pose of agent.

In the figure above the Gaussian variance is superposed on the mean of the robot position.

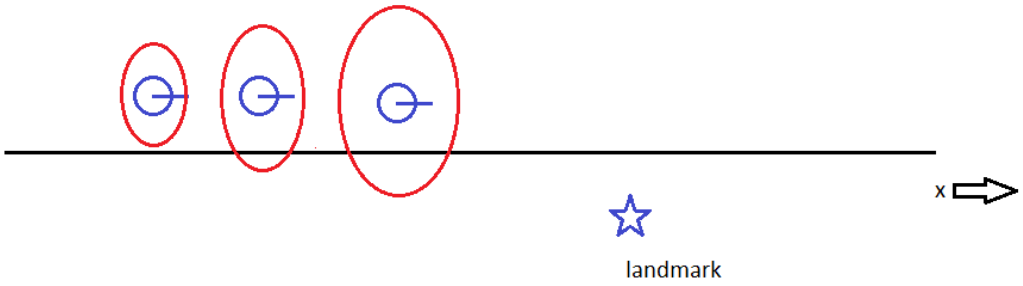


Figure 9 - EKF localization. Uncertainty grows as the agent moves.

(2-20) states variance of the robot position is increasing according to its motion model so throughout each step of movement, the uncertainty of robot's position is increasing, as it can be seen on Figure 9.

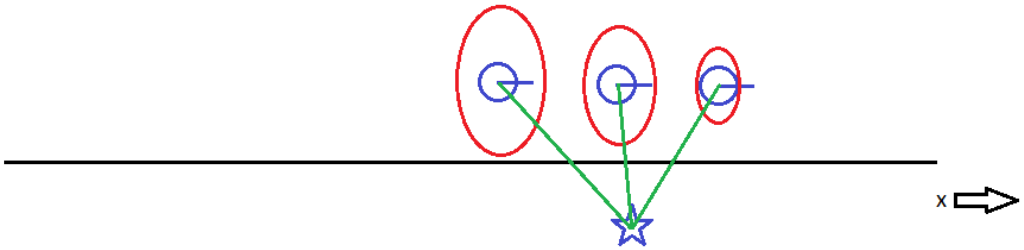


Figure 10 – EKF localization. The agent established contact with landmark.

When the robot senses the landmark, it will increase the precision of the belief of its location since the true pose of the landmark is known in the given map. Here the location cannot be determined with complete certainty, since landmark measurement system also has an uncertainty.

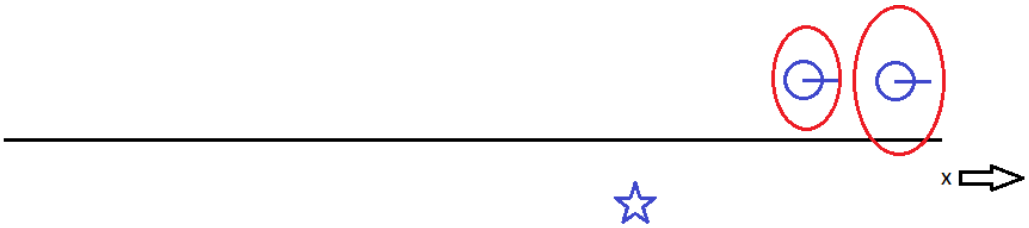


Figure 11 – EKF localization. The agent loose contact with landmark and keeps moving.

Whenever the robot loses contact with the landmark, variance of the agent keeps grooving while it moves.

A variant of EKF localization will be used through this work for localization of agents and surroundings. The equations of used variant of EKF localization will be given in open form in related sections of this work.

2.4 MAPPING

As stated previously localization is the act of finding coordinates of the agent given the coordinates of surroundings. Mapping in EKF SLAM is a similar action. This time the coordinate of agent is given and the coordinates of surroundings are asked.

Maps allow the vehicle to plan its movement within the environment in order to achieve its goals. The automatic creation of maps allows the vehicle to model the surroundings using its sensors. In some instances, the creation of an accurate map of the environment is the goal itself. While in many other circumstances the maps are used as a tool for performing other actions.

Once the position of the agent is given, creation of a map of the environment is a simple transformation operation. Given the current position estimate and an observation of the environment; the observation can be fused into the existing map to create an updated map estimate.[26]

Feature Based Maps

Feature maps represent the environment by the global locations and the orientations of parametric features. Localization is performed by extracting features from the sensed data and finding correspondence with the elements in the map. Just after finding correspondence between the sensor data and the previously known map landmarks, position of the belief map shall be updated according to variance of the sensor and the variance of the previous belief. By this way, localization operation becomes a multiple target tracking problem but, unlike normal target tracking, the targets are static and the observer is in motion.

In SLAM a feature can be used as a form of hypothesis about the existence and the location of a geometric shape or point in the environment. In feature based maps only the features and the accumulated evidence about them are stored. This leads to an abstraction of the environment. For example, in a feature map, a point cloud of a cylindrical shape can be stored as parameters of the cylinder (center of mass, orientation, height and radius). By this way point cloud information collected by sensor system is stored into a geometrically meaningful parameter set. Thus,

building a feature map can be considered as a form of data compression method. The abstraction of the environment could also lead to an understanding of the data on a deeper level. This method may be preferred for many cases, since the features are more recognizable by humans.[20]

Ultrasound sensors and laser scanner sensors can be used to create 3D point clouds to extract features. The work of Weingarten[27] is a significant example of utilization of distance sensors to execute 3D feature based EKF SLAM, that planar features are extracted from point clouds to represent environment via infinite planes. Mono or stereo camera systems can also be used to extract features to represent the environment. Kalay[28] used both mono and stereo camera systems to extract interesting points, thus represented the environment via these 3D point features. For a stronger representation of the environment, Lemaire [13] used a mono camera system to extract line features in a 3D map. An alternative method to extract features from the environment is fusing the laser (or sonar) scanner data with camera image. Work of Arras[29] is a significant example of fusing a laser scanner data and camera data to extract lines in a 2D map. Similarly Biber[30] used a 2D laser scanner system and a camera to represent indoor environment with 3D vertical planes. Finally Puente[15] used a laser scanner system to create a point cloud; segmented the point cloud with visual image segmenting techniques; extracted planar, cylindrical and spherical features from the point cloud and placed these features in a 3D map to represent the environment (with features those are represented with 3D positions and orientations). In all the works stated above, a method to extract features from a camera or a scanner system is presented and point, line, plane or some higher level features such as spheres or cylinders are used to represent the environment in a 2D or 3D map.

Each feature is entirely defined by its parameter set. For example, a cylindrical feature in a 2D map might be defined by $f = (t, r, x, y)$, where t is the feature type (cylinder), r is the cylinder radius, and x and y define its 2D position. Only the location information is directly useful for localization but the other information serves to assist landmark recognition for data association.[7]

Feature based maps does not show free space. Instead they only represent existence of surroundings with some parameters and this property of feature based maps makes them easy to work with. Because, every feature is represented by a set of mathematical parameters and once parameter set is chosen effectively, it is efficient to work with feature based maps. The maps composed by occupancy information usually need more memory since they also store information for free space. In addition to this, directly using raw data as a map construction tool (such as point cloud) needs excessive amount of memory compared to other methods.[7] Different mapping techniques will be compared with feature maps at the following section.

2.4.1 Comparison of Feature Based Mapping with other Mapping Methods and Contribution of Using Direction Vector

Recent work about 3D mapping and localization shows that, extending SLAM algorithm to 3D environment is very promising. Such maps can be used for visualization of architects, fire-fighters or virtual reality applications. On the other hand, even though 2D approaches are easier to deal with and need less power of computation, they may fail if the environment is non-planar, sensors are masked by a group of objects or if the robot is not restricted to ground plane.[39] Actually it can be said that, 3D maps are useful compared to 2D ones since real world itself is a 3D environment and realistic simulation of the world requires 3D localization and mapping for many cases. There are several approaches for 3D mapping of the environment. Weingarten[3] states that, most important mapping categories of representations are feature-based, grid-based and raw data-based mapping of the world.

Occupancy grids represent a region as a matrix of cells. Each cell describes a small rectangular area in the environment, and indicates the probability that, the area is occupied or not. Occupancy grids are effective localization method in relatively small environments. Filling values for grids does not require significant data processing operation and the data collected by sensor system is usually directly usable to fill the map. Thus grid maps are effective for multi sensor fusing

operations. Occupancy grids also offer an explicit representation of both occupied and free space, which is useful for navigation and path planning.[7]

A significant difficulty about occupancy grids mapping is data association. The *cross-correlation search*, within the region of the vehicle pose uncertainty, requires great amount of processing power if the search-space is large.[7] The above difficulties, makes cycle detection a challenging issue for grid based maps. If the agent just goes through a corridor and returns from the same path, the agent can correct its odometric error incrementally. However once a cyclic environment is required to be mapped, finding a true cross-correlation between the existing point and starting point in efficient way is a problematic issue for the algorithm.[2]

The most significant difficulty with occupancy grids is the tradeoff between grid resolution and the computational complexity. Ideally, to capture environmental detail and to facilitate accurate pose estimation, the grid size must be as small as possible, whereas for feasible computation a larger grid size may be necessary. Also; for high resolution grid maps, tasks like path-planning become computationally expensive. Methods to obtain variable resolution, and focus CPU resources at regions of environmental complexity, have been presented[31] but these implementations possess difficulties of their own.[7]

An alternative approach for mapping for SLAM is using unprocessed raw data. Such approach is also called *scan correlation* or *range-image registration*. The idea of using raw data is to align consecutive scans taken by sensors from the robot and estimate the agent's trajectory while creating a consistent map. A popular variation of scan matching method is known as *Iterative Closest Point* (ICP). ICP looks for closest point pairs between two different scans and iteratively minimizes their relative transform.[3]

Feature based maps need the environment to be parameterized, however in some cases, the geometric shapes in the environment is difficult to parameterize. This property of feature based EKF SLAM makes it weak in unconstructed environment such as mines or underwater applications. Thus, using raw data is superior to

feature based methods at unstructured environments, because it does not need to form sensor data, but directly uses it.[7] On the other hand, -just like grid based mapping- scan-matching is a tracking model and due to this reason it does not provide strong solution for global localization and creates several problems for cycle detection.[3] Additionally, feature based mapping methods stores data in compact packages, however the raw data maps stores individual scans and due to this reason mapping needs significant amount of memory environments.[20]

For feature-based approaches, it is assumed that the physical environment can be modeled by geometric features. This is the case especially holds for man-built environments like building interiors or cities which can be represented by a meaningful set of points, lines or planes. Due to the compact structure of data required to represent these features, the resulting maps are compact and the associated algorithms are efficient in comparison to other approaches using more memory like occupancy grids or raw data mapping. On the other hand, they require a reliable feature extraction mechanism, which may be a challenging task for several features. Secondly, in feature-based approaches, the data association or correspondence problem has to be solved.[3]

The Extended Kalman Filter (EKF) is a widely used estimation tool applicable to feature-based localization and mapping. It has the advantage that it provides an analytical solution to the SLAM problem, which leads to efficient implementation with high accuracy. A significant disadvantage of the EKF-based SLAM approach is its restriction to Gaussian error distributions which have a single peak and therefore can represent only a single hypothesis about the robot pose. In addition to this, complexity of the EKF-SLAM scales with square of the number of features.[3] Before implementing feature based EKF SLAM in real environment, the type of data abstraction method must be decided. As stated previously the sensor data is needed to be processed and several features must be extracted.

There are several methods of representing the features in space. The selection of features is based on the environment and the sensor system. Additionally the agent

must be able to be localized in the feature map. Without the ability to localize the robot, using the map, it would be impossible to execute EKF SLAM algorithm.[20] As a result; one requirement of a map feature is that it must be able to store necessary pose parameters of the robot.

The need of extracting information from the sensor data is another parameter for selected feature geometry. The sensor measurements must be able to constrain some of the geometry of features. Thus, the available sensors limit the feature selection. In addition to this, a feature may not be fully observable in a single sensor reading. This leads to the situation of partially observed features. In some cases, the partial observability depends on the frame of reference of the sensor data.[20] As the robot moves, it can fully observe the features and the ability to merge these measurements may needed to achieve several tasks. For example the walls of a corridor are not fully observable for many cases, but the task of the agent may require the algorithm to represent the wall as a single feature.

Environment is a significant constraint for feature selection. For example planar features cover a great part of indoor of a building or a city which are known as constructed environments.[32] On the other hand, a tree is a significant feature for mapping non-structured or semi-structured outdoor environment. [12] A point, a line or a plane may be used as feature for EKF SLAM.[20] In addition to these ones, different kinds of 3D features such as cylinders or spheres can be used to represent indoor environment.[15] Representing the plane segments is a vital issue for indoor feature based SLAM. Such representation covers surfaces of many objects in a building such as walls, tables, cabinets, doors, couches and so on.[32] According to Folkesson[20] a vertical planar segment such as a wall can be represented with following parameter sets:

Slope and Intersection: It is a way of representing a wall with simple line equation $y = mx + c$. Even though it seems to cover requirements of a wall, it is infinite in length; parameter m is nonlinear and has a singular behavior around $90°$.[20]

Distance and Direction: One of the most simple map origin based representations is to use the perpendicular distance to the wall and the orientation of the wall. However this method results an infinite wall. Additionally a second problem called “lever-arm” effect[33] occurs; which points that, if the wall is passing close from origin point of the map, small errors in orientation angle leads the wall located in wrong position. An improved version of this representation also parameterizes starting and ending points of the wall, however it also suffers from lever-arm effect.

Center Point, Length, and Orientation: Since each wall is positioned according to its own center of mass, this method does not suffer from level-arm effect. Such representation needs the planar object to be fully observable, since it uses center of mass of the wall as an input parameter.[20] However a wall may not be fully observable for many cases as stated previously. For example, when one requires constructing map of a closed door environment, mapping the walls of the room may be a significant problem. Because the need of measuring true place of CoM of walls requires that the wall must be fully observable in one single observation and this may be hard or impossible for several cases depending on sensor range, sensor field of view or interference of other objects that stands on the floor. On the other hand, since infinite planes are used for many different plane representations, the methods that use different kinds of transformation or plane representation systems (such as SP-Model) may suffer from correspondence problem as Weingarten[3] states in his article. Whereas using CoM information may provide a significant tool to avoid miscorrespondence for landmarks.

SP-Model: The SP-model offers a solution to the lever-arm effect too. In this model, a local coordinate frame is attached to the center of the wall and the wall is described by the transformation to this local frame. This allows for a representation of the uncertainty of the wall location in this local frame. When compared with center point length and orientation method a drawback of the SP-model is that it does not handle information along the direction of the wall as easily. However it

creates less coupling between the parameters of the landmarks and does not require full observability for the feature. [20]

In this work planar segments will be represented by center point and orientation. Such representation requires the plane segments to be fully observable; due to this reason, in experimental part, small planar objects will be used, instead of wall like large plane segments. In addition to this representation, planar segments will also be represented by four corners, which is a 3D point feature map representation method. Using planar segments provides strong feedback to robot's angular pose while dealing with feature based 3D EKF SLAM. Contribution of using orientation for features will be examined in Section 5.8.

As stated previously Feature based EKF SLAM is known as an efficient SLAM algorithm in terms of required processing power and used memory for mapping. In addition to this, it is possible to provide an improved computational complexity by increasing the level of abstraction for feature based SLAM.

Computational Complexity for Feature Maps

A contribution of feature based EKF SLAM is about computational complexity. Raw data is processed and several features are extracted from it. Most popular solution to SLAM, considers it as a stochastic process in which EKF is used to compute an estimation of a state vector, together with the covariance matrix. Most of the processes associated to the move-sense update cycle of EKF SLAM are linear in complexity. As Paz [34] states; the exception is the update of the covariance matrix, which is $O(n^2)$. The EKF solution to SLAM has been used successfully in small scale environments, but $O(n^2)$ limits the use EKF-SLAM in large environments. Feature data may be represented with several different methods; for example at experimental part of this work, rectangular planar segments are used as objects. At this point, instead of representing objects with several points or lines, executing this operation with a 6D feature that indicates position and orientation may be a beneficial option. By this way, the data may be stored with more compact structure and complexity shall be reduced by decreasing number of landmarks. With a more

formal expression; if data structure of landmarks is altered in the way that number of landmarks is decreased by *k* times, while number of parameters represents a landmark is increased *m* times (*k* > *m*); complexity of covariance matrix will decrease from $O(n^2)$ to $O((n * m/k)^2)$. By this way complexity of EKF SLAM algorithm shall be reduced via more compact data structure that represents landmarks. An implementation of decreasing complexity of EKF SLAM by increasing parameter size can be seen in Section 5.8.

Data Association Problem in Feature Maps

As stated in section 2.3.3, EKF localization and mapping method will be used throughout this work. A drawback of EKF localization is, it models uncertainty with Gaussian form and this property requires initial pose to be known. In other words, once the algorithm is executed to update location of landmark, it combines previous position and new sensor reading to localize final position. The need of usage of previous position brings the requirement of correspondence between previously known map and the new sensor reading. In short, the sensor reading of each feature must be associated to a landmark in the map.

Correspondence problem is arguably the main weakness of feature map localization.[7] Correct pose estimation relies on finding correct correspondence between a feature observation and its associated map feature. Whenever a misassociation occurs, the real error -possibly- increases. This effect may lead the vehicle get lost or even get the map collapsed. Most feature map localization implementations are susceptible to data association failure because they rely on the association methods developed for target tracking, which treat each measurement in isolation. However one must realize that, other mapping methods such as raw data mapping or grid mapping also suffers from associating the initial location to existing location while closing a loop. In addition to this, a stronger solution for data association problem can be obtained by increasing the abstraction level of features. In other words, fusing point features to obtain planes or fusing planar features to obtain objects as a feature type may provide significant improvement for data association problem.

2.4.2 Plane Model

Planar segments will be used at necessary parts of this work. A plane in a 3D space can be represented with formulation:

$$Ax + By + Cz + D = 0 \quad \text{2-22}$$

A, B, C, D are unique parameters that represent the plane. There are also other formulations:

$$nx - d = 0 \quad \text{2-23}$$

in which;

$$n = \left(n_x, n_y, n_z\right)^T \quad \text{2-24}$$

Above formulation represents a plane with a normal vector *n* and distance to origin, *d*. For a similar useful plane representation one can change *d* with a point in 3D space:

$$(p', n)^T \quad \text{2-25}$$

in which;

$$p' = (x, y, z) \quad \text{2-26}$$

$$n = (n_x, n_y, n_z) \quad \text{2-27}$$

The formulation above is composed of a normal vector that shows direction of plane and a point that the plane passes. This formulation is useful to represent the plane segments, since it consists a specific point (such as center of mass) and a direction vector. Because of this property the formulation (2-25), (2-26) and (2-27) will be one of the main plane segment representations used in this work. The reader shall see literature, for different kinds of representations and properties of these formulations.

Plane Segments

A plane segment can be represented by a number of parameters. The center of mass (CoM) of the plane can be represented by (2-26) while the normal of the plane

can be represented by (2-27). Since it is a normal vector, once two of the parameters in (2-27) are known, the third parameter can be found. So it can be said that; in 3D space, two parameters are needed to represent angular orientation of a plane.

In addition to these parameters, size of the plane segment may also be used to improve correspondence and provide a stronger illustration of the map. Three more parameters can be used for this task:

$$other\ parameters\ of\ plane = (w, h, t) \quad \textbf{2-28}$$

The three parameters shown in (2-28) represents width (*w*), height (*h*) and roll (*tilt*-*t*) angle of the plane. So they can be used as an additional tool to improve the solution of correspondence problem of features in the map and they will also be used as a tool to visualize the map in experimental part.

So it can be said that eight parameters can be used to represent a planar segment in a 3D map:

$$Planar\ Segment = (x, y, z, n_x, n_y, w, h, t) \quad \textbf{2-29}$$

2.4.3 Extracting Planar Segments from Point Cloud

Laser scanners, IR scanners and sonar scanners can be used to create a distance image of the environment from their point of view. In a scanner sensor, the distance sensing part measures the distance to closest object on the beam. This beam is used to scan the area and a distance is measured for each angle set of the scanner. By this way, the sensor constructs a point cloud representation of the environment.

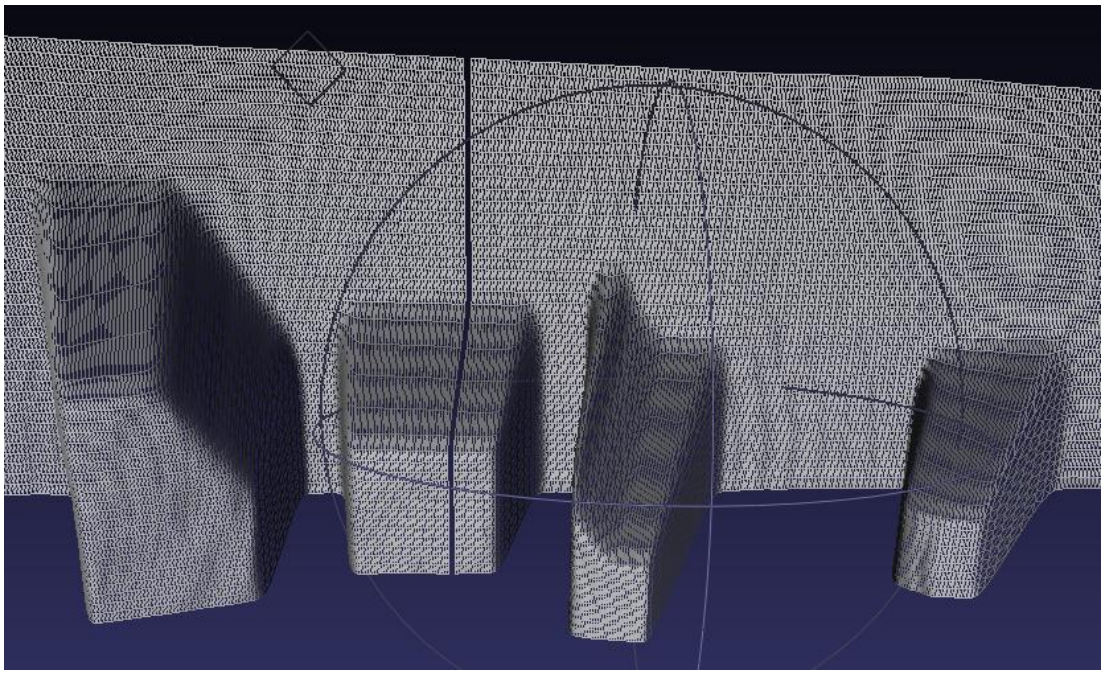


Figure 12 – A point cloud of four objects standing on the floor. Background is the wall. Data is collected IRSCAN.

There are various types of feature extraction/plane extraction methods. A feature extraction algorithm, first processes and labels each point as a part of feature. Every single point that belongs to the same feature is required to be labeled with the same label. After all points are labeled, the ones with *same label* must be processed and several characteristics and parameters of features shall be extracted.[3] The output of feature extraction algorithm may be a set of 3D points or even may be the positions, orientations and complicated information for a set of higher level features like planes spheres or arcs.

A plane extraction algorithm extracts parameters of a plane $(x, y, z, n_x, n_y, w, h, t)$. There are various types of plane extraction methods. In order to get a better overview of the numerous existing planar segmentation algorithms, a classification is useful. There are two main ways of segmenting entities in a picture, dominating the literature. The former is split technique. This method starts from the entire image and checks if it is homogeneous according to some criterion. If not, the

object is split into sub objects. The action is applied until each sub object becomes homogeneous.[35]

The latter method is a merging sub objects to a greater one. This method is also called region-growing. A region-growing algorithm starts from single entities of an input range image. After processing these small entities, they are merged and grown into larger regions with matching neighbors according to some parameter. This operation is stopped, when a certain stopping criteria is achieved.[35] An example for latter method can be seen in Figure 13.

Figure 13 – An example of surface merging method. Upper left image is the photograph of real environment. Upper right image shows the point cloud representation of the environment via a laser scanner. Bottom left image shows planer segment which are created by a plane fitting method. Finally, similar neighboring planes are merged and label as a feature as shown in bottom right image. Figure taken from [3]

Another categorization for plane extraction methods is region based extraction and edge based extraction. Region based methods use the regions of point clouds to

detect the features. On the other hand edge based methods extract the edges of features according to some criteria and label the points inside closed edges.[3] In this work, edge based methods will be used for sensor experiments.

There are various types of edge detection methods. A scanner sensor with a specific resolution creates a distance image of environment. An illustration of a distance image of Figure 12 is as follows:

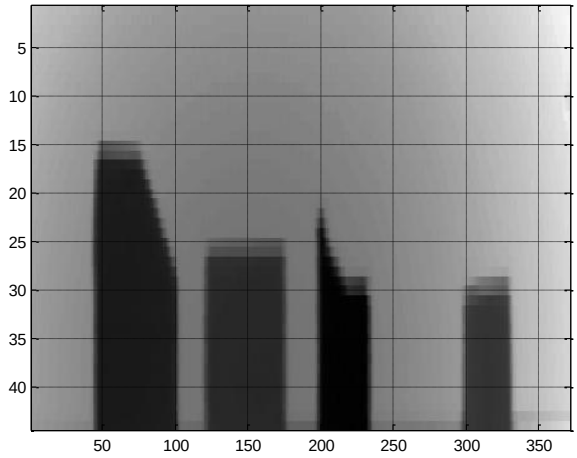


Figure 14 – Distance image of 4 objects. Darker pixels indicate close distance while brighter pixels indicate farther. Data is collected by IRSCAN.

In distance image illustrated at Figure 14, edge finding methods can be used to detect surrounding edges of the image. A few methods used in this work are briefly explained in

APPENDIX A. For each method explained in

APPENDIX A, MATLAB 7.12.0 code is be implemented to Figure 14 and resulting edge map will be illustrated. By this way the edges of features can be marked and feature extraction algorithms can be executed via edge based methods.

CHAPTER 3

FEATURE BASED EXTENDED KALMAN FILTER SLAM WITH 3D POSITION AND ORIENTATION FEATURES

Extended Kalman Filter Simultaneous Localization and Mapping (EKF SLAM) is the process of incrementally building a map of the environment while tracking robot's pose at this map. As stated previously EKF SLAM is composed of two problems; localization and mapping. Localization problem was discussed in Section 2.3 while mapping was discussed in Section 2.4. In localization part, true map of the environment was given and the task is locating the agent probabilistically in this map. On the other hand in mapping part, transformation of feature parameters from true position of robot is performed.

Solving the problem of localization requires an a priori map of the environment. However, such a map is not always available. Blue prints may not exist or represent real pose of surroundings, furniture in an office environment for example may have shifted over time, which makes the map out of date.

Mapping on the other hand requires a known robot pose. GPS (Global Positioning System) makes it possible with some uncertainty at outdoor environments. However, GPS cannot be used in conditions like forests or cities, that GPS signal is weak or in places such as caves and indoor environments. The robot pose can be provided by sensors like wheel encoders, gyros or accelerometers; however, these sensors accumulate errors and therefore can only be used reliably over short distances.

Due to these reasons both localization and mapping algorithms must be processed recursively in SLAM. Unlike the problems discussed in Section 2.3 and Section 2.4, but this time, both location of the agent and poses of the features are probabilistic.

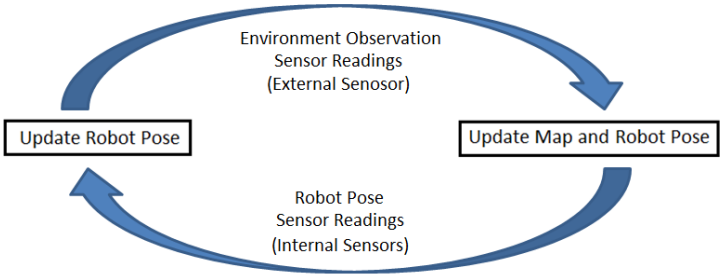


Figure 15 – Diagram that represent recursive operations of EKF SLAM.

A graphical representation of EKF SLAM is illustrated at Figure 15 which is a very similar representation with Figure 1. As it can be seen on the figure, EKF SLAM is a recursive algorithm that updates pose of robot and features. Internal sensors of the agent (wheel encoders, gyro, accelerometer etc.) detect movement and update belief of the robot. Once the environment is sensed via external sensors (laser scanner, IR scanner, camera etc.) both the pose of the robot and positions of its surroundings are updated.

The definition of the problem shall be formalized at this point. A sensor system provides data from the environment. Let the data set provided by the sensor system be *z*. Data collected by the sensor system is stored with a data structure and let this previously stored data be μ_{lm}. Finally there is another data structure that points the pose of the robot. Let robot pose be represented as μ_r. At this point, to make inferences about landmark pose (μ_{lm}) and robot pose (μ_r) via measurements (z), a function is needed to be defined that relates these three data types. Let this relation be named *innovation* (η). So η is a function of μ_{lm}, μ_r and z.

Jacobian matrix of innovation is stated by;

$$\left(J_{\eta z}\ J_{\eta \mu_{lm}}\ J_{\eta \mu_r}\right) = \left(\frac{\partial \eta}{\partial z}\frac{\partial \eta}{\partial \mu_{lm}}\frac{\partial \eta}{\partial \mu_r}\right) \quad \text{3-1}$$

EKF SLAM use Jacobian functions to calculate these relations of innovation.[3]

The algorithm of EKF SLAM executes necessary parts of these relation functions via control inputs, sensor observations and previously obtained landmark poses.

3.1 FEATURE BASED EKF SLAM WITH 3D POSITION AND ORIENTATION

In this work a code for 3D feature based EKF SLAM is presented and this code is fit into a simulation. After simulation part, the code written is used to execute data collected by IRSCAN. The pseudo code of full EKF SLAM, EKF SLAM simulation results, specifications of the sensor will be explained in the following sections. This chapter aims to explain EKF SLAM in 3D environment with plane segment features. Note that theoretic perception of EKF SLAM algorithm given here closely matches to the one described by Thrun[2]. However unlike the approach of Thrun the code in this work is extended to 3D space with planar features; instead of 2D space with point features.

The agent is assumed to work in a structured indoor environment with a flat ground (z=0). Landmark planes in the environment are represented by their 3D position and orientation. Even though only planar features are used in the real sensor code, the same algorithm can also work for other features such as non-planar segments once feature extraction algorithm can handle them.

3.1.1 Data Structure

In EKF SLAM all pdf's of the pose of the robot and its surroundings are assumed to be Gaussian. Structure of stored data keeps parameters of these Gaussian formed pdf's. As stated in Section 2.3.3 a Gaussian shall be represented as $N(\mu_t, \Sigma_t)$. Here μ_t represents maximum likely location and Σ_t stands for covariance of Gaussians.

Maximum likely point of the agent is kept in the form:

$$\mu_{robot,t} = [\mu_{robot,x,t}\ \mu_{robot,y,t}\ \mu_{robot,z,t}\ \mu_{robot,\alpha,t}\ \mu_{robot,\varphi,t}\ \mu_{robot,\omega,t}]^T \quad \textbf{3-2}$$

In above equation $\mu_{robot,x,t}\ \mu_{robot,y,t}\ \mu_{robot,z,t}$ represent maximum likely position of the agent while $\mu_{robot,\alpha,t}$, $\mu_{robot,\varphi,t}$, $\mu_{robot,\omega,t}$ represent maximum likely value of yaw, roll and pitch angle of the agent relative to the global coordinate system.

Since the agent moves on a flat ground, it has a changing yaw angle but pitch and roll angles of agent are constant (assumed zero). So pose of the robot can be specified with three parameters, two parameters for position (x, y) and one for orientation (α). Flat ground assumption shall be represented as:

$$\mu_{robot,z,t} = 0 \quad \textbf{3-3}$$

$$\mu_{robot,\varphi,t} = 0 \quad \textbf{3-4}$$

$$\mu_{robot,\omega,t} = 0 \quad \textbf{3-5}$$

So;

$$\mu_{robot,t} = [\mu_{robot,x,t}\mu_{robot,y,t}\ \mu_{robot,\alpha,t}]^T \quad \textbf{3-6}$$

Similarly maximum likely pose of the landmark *i* at time *t* is kept in the form;

$$\mu_{i,t} = [\mu_{i,x,t}\mu_{i,y,t}\mu_{i,z,t}\ \mu_{i,\alpha,t}\mu_{i,\varphi,t}\mu_{i,\omega,t}]^T \quad \textbf{3-7}$$

In which $\mu_{i,x,t}, \mu_{i,y,t}$, $\mu_{i,z,t}$ stand for the position of center of mass (CoM) of the landmark and $\mu_{i,\alpha,t}, \mu_{i,\varphi,t}, \mu_{i,\omega,t}$ represent the orientation of the landmark in 3D space. Since height, roll and pitch parameters of the robot are fixed, roll and pitch parameters of landmarks will not be used in EKF SLAM simulation. The reason why these two parameters are not used will be explained with more detail in Jacobian part of algorithm. However, these two unused parameters are useful while trying to find true correspondence information. Due to this reason they shall kept in data structure of the algorithm. So the pose of a feature in SLAM algorithm is composed of four parameters. The first three parameters are used to specify the position of the center of mass (x, y, z) and the last parameter will be used to show the

orientation ($yaw - \alpha$) of the feature. Other two parameters ($roll - \beta, pitch - \gamma$) will not be actively used in the algorithm. Thus maximum likely position and orientation of data is examined above and will be kept in the following matrixes:

$$\mu_t = [\mu_{robot,x,t}\mu_{robot,y,t}\ \mu_{robot,\alpha,t}\mu_{1,x,t}\mu_{1,y,t}\mu_{1,z,t}\ \mu_{1,\alpha,t}\ \mu_{2,x,t}\mu_{2,y,t}\mu_{2,z,t}\ \mu_{2,\alpha,t}\ \dots]^T \quad \textbf{3-8}$$

$$\mu_{landmark\varphi,\omega,t} = \begin{bmatrix} \mu_{1,\varphi,t} & \mu_{1,\omega,t} \\ \mu_{2,\varphi,t} & \mu_{2,\omega,t} \\ & \vdots \end{bmatrix} \quad \textbf{3-9}$$

So for *N* landmarks (3-8) is a *[4N + 3, 1]* matrix and (3-9) is a *[N, 2]* matrix.

In EKF SLAM variance and covariance of every parameter must be kept just as maximum likely points done, so;

$$\Sigma_t = \begin{bmatrix} \Sigma_{robot,robot,t} & \Sigma_{lm,robot,t} \\ \Sigma_{robot,lm,t} & \Sigma_{lm,lm,t} \end{bmatrix} \quad \textbf{3-10}$$

in which;

$$\Sigma_{robot,robot,t} = \begin{bmatrix} \sigma_{robot\ x,robot\ x,t} & \sigma_{robot\ x,robot\ y,t} & \sigma_{robot\ x,robot\ \alpha,t} \\ \sigma_{robot\ y,robot\ x,t} & \sigma_{robot\ y,robot\ y,t} & \sigma_{robot\ y,robot\ \alpha,t} \\ \sigma_{robot\ \alpha,robot\ x,t} & \sigma_{robot\ \alpha,robot\ y,t} & \sigma_{robot\ \alpha,robot\ \alpha,t} \end{bmatrix} \quad \textbf{3-11}$$

$$\Sigma_{lm,lm,t} = \begin{bmatrix} \sigma_{1x,1x,t} & \sigma_{1x,1y,t} & \sigma_{1x,1z,t} & \sigma_{1x,1\alpha,t} & \sigma_{1x,2x,t} & \sigma_{1x,2y,t} & \sigma_{1x,2z,t} & \sigma_{1x,2\alpha,t} & \\ \sigma_{1y,1x,t} & \sigma_{1y,1y,t} & \sigma_{1y,1z,t} & \sigma_{1y,1\alpha,t} & \sigma_{1y,2x,t} & \sigma_{1y,2y,t} & \sigma_{1y,2z,t} & \sigma_{1y,2\alpha,t} & \\ \sigma_{1z,1x,t} & \sigma_{1z,1y,t} & \sigma_{1z,1z,t} & \sigma_{1z,1\alpha,t} & \sigma_{1z,2x,t} & \sigma_{1z,2y,t} & \sigma_{1z,2x,t} & \sigma_{1z,2\alpha,t} & \\ \sigma_{1\alpha,1x,t} & \sigma_{1\alpha,1y,t} & \sigma_{1\alpha,1z,t} & \sigma_{1\alpha,1\alpha,t} & \sigma_{1\alpha,2x,t} & \sigma_{1\alpha,2y,t} & \sigma_{1\alpha,2z,t} & \sigma_{1\alpha,2\alpha,t} & \dots \\ \sigma_{2x,1x,t} & \sigma_{2x,1y,t} & \sigma_{2x,1z,t} & \sigma_{2x,1\alpha,t} & \sigma_{2x,2x,t} & \sigma_{2x,2y,t} & \sigma_{2x,2z,t} & \sigma_{2x,2\alpha,t} & \\ \sigma_{2y,1x,t} & \sigma_{2y,1y,t} & \sigma_{2y,1z,t} & \sigma_{2y,1\alpha,t} & \sigma_{2y,2x,t} & \sigma_{2y,2y,t} & \sigma_{2y,2z,t} & \sigma_{2y,2\alpha,t} & \\ \sigma_{2z,1x,t} & \sigma_{2z,1y,t} & \sigma_{2z,1z,t} & \sigma_{2z,1\alpha,t} & \sigma_{2z,2x,t} & \sigma_{2z,2y,t} & \sigma_{2z,2z,t} & \sigma_{2z,2\alpha,t} & \\ \sigma_{2\alpha,1x,t} & \sigma_{2\alpha,1y,t} & \sigma_{2\alpha,1z,t} & \sigma_{2\alpha,1\alpha,t} & \sigma_{2\alpha,2x,t} & \sigma_{2\alpha,2y,t} & \sigma_{2\alpha,2z,t} & \sigma_{2\alpha,2\alpha,t} & \\ & & & \vdots & & & & & \ddots \end{bmatrix} \quad \textbf{3-12}$$

$$\Sigma_{robot,lmi,t} = \begin{bmatrix} \sigma_{robotx,lmix,t} & \sigma_{robotx,lmiy,t} & \sigma_{robotx,lmiz,t} & \sigma_{robotx,lmi\alpha,t} \\ \sigma_{roboty,lmix,t} & \sigma_{roboty,lmiy,t} & \sigma_{roboty,lmiz,t} & \sigma_{roboty,lmi\alpha,t} \\ \sigma_{robot\alpha,lmix,t} & \sigma_{robot\alpha,lmiy,t} & \sigma_{robot\alpha,lmiz,t} & \sigma_{robot\alpha,lmi\alpha,t} \end{bmatrix} \quad \textbf{3-13}$$

$$\Sigma_{lm,robot,t} = \Sigma_{robot,lmi,t}{}^{T} \quad \textbf{3-14}$$

Finally variance of $landmark\ \varphi, \omega, t$ are kept with matrix;

$$\Sigma_{\ landmark\ \varphi,\omega,t} = \begin{bmatrix} \sigma_{1,\varphi,t} & \sigma_{1,\omega,t} \\ \sigma_{2,\varphi,t} & \sigma_{2,\omega,t} \\ & \vdots \end{bmatrix} \quad \textbf{3-15}$$

Diagonal elements of 3-10 represent variance for each parameter and other entries represent covariance of parameters.

In short, in this work parameters of EKF SLAM will be kept in the matrixes given below:

Table 1 – Size of parameters for 3D position and orientation EKF SLAM

	symbol	size of matrix
Mean of parameters	$\mu_{\ t}$	4N + 3 x 1
	$\mu_{\ landmark\ \varphi,\omega,t}$	4N x 2
Variance of parameters	Σ_t	4N + 3 x 4N +3
	$\Sigma_{\ landmark\ \varphi,\omega,t}$	4N x 2

Where N is the number of observed landmarks and:

$\mu_{\ t}$ stores maximum likely values of x, y, θ parameters of *robot* and maximum likely points of x, y, z, θ parameters of each landmark

$\mu_{\ landmark\ \varphi,\omega,t}$ stores maximum likely values of φ, ω parameters for each landmark

Σ_t stores covariance of each parameter of $\mu_{\ t}$

$\Sigma_{\ landmark\ \varphi,\omega,t}$ stores variance of each parameter of $\mu_{\ landmark\ \varphi,\omega,t}$

Every single parameter of $\mu_{\ t}$, $\mu_{\ landmark\ \varphi,\omega,t}$, Σ_t, $\Sigma_{\ landmark\ \varphi,\omega,t}$ is a real number.

3.1.2 Displacement of the Agent (Update Robot Pose)

In this work the agent is assumed moving and stopping to collect data. After collecting data, the robot is moved once more and stopped again. Robot motion model is assumed differential motion model and all the parameters will be updated according to this model.

Differential drive robot control input is:

$$u_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \quad \text{3-16}$$

Here v_t is translational motion and ω_t is rotational motion. The goal is to find final pose of the robot, given the initial pose (an illustration is shown in Figure 16).

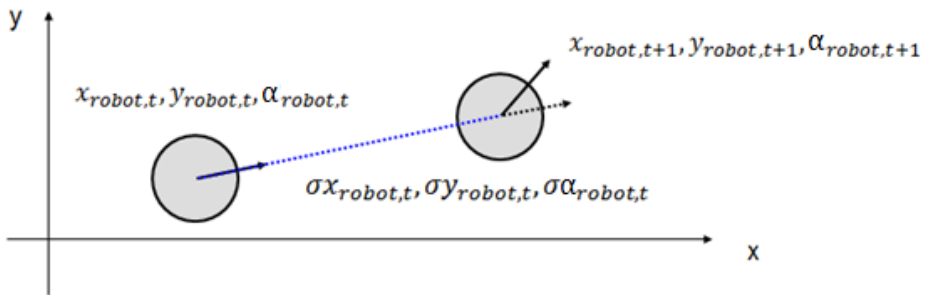


Figure 16 – An illustration that shows movement of a differential drive robot.

In this exact motion model: [2]

$$x_{robot,t+1} = x_{robot,t} + \sigma x_{robot,t}\,, \quad \text{3-17}$$

$$y_{robot,t+1} = y_{robot,t} + \sigma y_{robot,t}\,, \quad \text{3-18}$$

$$\alpha_{robot,t+1} = \alpha_{robot,t} + \sigma \alpha_{robot,t}\,, \quad \text{3-19}$$

Where:

$$\sigma x_{robot,t} = -\frac{v_t}{\omega_t}\sin(\alpha) + \frac{v_t}{\omega_t}\sin(\alpha + \omega_t\,\Delta t)\,, \quad \textbf{3-20}$$

$$\sigma y_{robot,t} = \frac{v_t}{\omega_t}\sin(\alpha) - \frac{v_t}{\omega_t}\cos(\alpha + \omega_t\,\Delta t)\,, \quad \textbf{3-21}$$

$$\sigma\alpha_{robot,t} = \omega_t\,\Delta t\,. \quad \textbf{3-22}$$

So, equations of exact motion model of agent are presented.

3.1.2.1 Uncertainty of Robot Motion

In EKF SLAM once the pose of robot is updated, covariance matrix must also be updated. The new uncertainty is a function of previous uncertainty and added from uncertainty of motion. The Jacobian update equations of robot pose uncertainty can be found by simple derivative operations as follows: [2]

$$G_t = \frac{\sigma\ new\ robot\ pose}{\sigma\ previous\ robot\ pose} \quad \textbf{3-23}$$

$$V_t = \frac{\sigma\ new\ robot\ pose}{\sigma\ control\ input} \quad \textbf{3-24}$$

And M_t represents the uncertainty comes from robot motion, then;

$$\Sigma_t = G_t\ \Sigma_{t-1}G_t^T + V_t M_t V_t^T \quad \textbf{3-25}$$

So, update equations of uncertainty of robot's pose are presented.

3.1.3 Integrating Sensor Reading (Update Map and Robot Pose)

Once the sensors of the agent are operated and collected data from the environment, observed pose of the features can be found. Previous belief of the robot and the feature poses may differentiate from newly observed poses. Due to this reason the belief position of the agent and the features must be updated according to the sensor observation.

One can say;

$$(\mu_t, \Sigma_t) = f(\mu_{t-1}, \Sigma_{t-1}, z_{observation,t}) \quad \textbf{3-26}$$

where, $z_{observation,t}$ represents sensor reading. An illustration of this update step is as follows.

Figure 17 - Black stars represent previous belief position of landmarks, while the red ones illustrates pose of recent measurement. Blue ones are the final belief after update step.

Because of imprecision of the sensors and uncertainty of robot pose both new measurement and previous belief may have some error as shown Figure 17 (left). The new belief pose of the landmarks and the agent are determined from the combination of the previous belief and the recent sensor observation as can be seen in Figure 17 (right). While combining these two maps, covariances of beliefs are used as a weighting factor to locate new poses. When the new map is compared to the old one, variances are expected to decrease and covariance between parameters are expected to get stronger.

In EKF SLAM variances of sensor system and robot belief must be converted to comparable forms. In our solution variance of the map will be converted to form of the sensor noise and these two set of variance will be compared via a factor called Kalman Gain (K_t).

Let's say *i* different landmarks are observed in a set of sensor readings. These observations will be kept in a matrix; $z_{observation,t}$. After that part, one needs to

obtain predicted observation matrix by using the map data. This matrix will be shown with $z_{predicted,t}$:

$$z_{predicted,t} = f(\mu_t) \quad \textbf{3-27}$$

$z_{predicted,t}$ is the output of a transformation function from robot coordinate frame to sensor reading model.

Once $z_{predicted,t}$ is obtained, a Jacobian function shall be created to convert variance of belief to variance of observation;

$$H_t = \frac{\sigma\ observation\ for\ each\ landmark}{\sigma\ each\ pose\ parameter\ of\ \mu_t} \quad \textbf{3-28}$$

Open form of (3-28) will be given at (3-55) and Appendix B.

So, Q_t (sensor noise model) and Σ_t can now be compared in the form;

$$K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1} \quad \textbf{3-29}$$

New μ_t will be indicated by $\dot{\mu}_t$

$$\dot{\mu}_t = \mu_t + K_t\left(z_{observation,t} - z_{predicted,t}\right) \quad \textbf{3-30}$$

And new Σ_t will be indicated by $\dot{\Sigma}_t$

$$\dot{\Sigma}_t = (I - K_t H_t)\Sigma_t \quad \textbf{3-31}$$

So, input output relationship stated in (3-26) is obtained.

3.2 Feature Based EKF SLAM Algorithm Implementation in Simulation Environment with 3D position and Orientation

In this section, EKF SLAM algorithm which is used in simulation will be given in more opened form. Open form of equations, matrixes and Jacobian's will be presented and explained. The algorithm given below is implemented with MATLAB 7.12.0. Discussing insignificant details of simulation is not the main issue of this part but implementation of theoretical equations will be explained.

Robot and the Environment

It is assumed that the robot is operating in a 3D environment. The features are located with 3D position and orientation in the map. Feature extraction and correspondence problem is assumed to be solved and features with true correspondences will be considered given to the algorithm. The robot is assumed to be moving on a flat ground in a closed door environment (like a building or a room). Translational speed of the robot is v_t and rotational speed is w_t. Height of the sensor part of the robot from the ground is assumed to be zero for all observations.

Representation of Features by Sensor Point of View

The EKF SLAM code written for simulation is also tested with IRSCAN. So scanner sensor modeled in simulation has similar output form with IRSCAN.

When a 3D scanner collects data from the environment, it outputs a 3D point cloud illustration of the surroundings. After being processed with a feature extraction algorithm, the point cloud can be transformed to a set of features. It is assumed that feature extraction algorithm obtains center of mass (CoM) and also outputs angular orientation information for each detected feature. CoM information is formed from a range and two bearing parameters. An illustration of representation of CoM is shown in Figure 18.

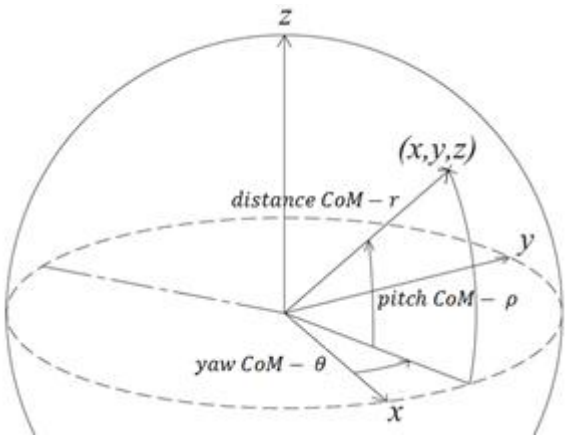


Figure 18 – Illustration of output of sensor system for CoM

As stated previously; for each feature, three parameters are used to represent CoM (r^i, θ^i and ρ^i from sensor coordinate system) and the other three parameters are used to represent angular orientation (α^i, β^i and γ^i from sensor coordinate system) of feature in 3D map. To clarify this issue, one can assume, that the features are plane segments scattered through the map. Since two orientation parameters will not actively be used, orientation of the sensor will be represented only by α^i.

Implementation

As stated in previous parts, robot and features will be represented according to global frame as presented by (3-7) and (3-8) as follows:

Maximum likely positions are represented as follows (3-8 modified);

$$\mu_t = [\mu_t^r \ \mu_t^1 \ \mu_t^2 \ \dots \ \mu_t^N]^T \qquad \textbf{3-32}$$
$$= [x_t^r y_t^r \alpha_t^r x_t^1 y_t^1 z_t^1 \alpha_t^1 x_t^2 y_t^2 z_t^2 \alpha_t^2 \dots \ x_t^N y_t^N z_t^N \alpha_t^N]^T$$

Covariance for *N* landmarks (3-10 modified):

$$\Sigma_t = \begin{bmatrix} \Sigma_t^{r,r} & \Sigma_t^{1-N,r} \\ \Sigma_t^{r,1-N} & \Sigma_t^{1-N,1-N} \end{bmatrix} \qquad \textbf{3-33}$$

The following steps are followed for feature based EKF SLAM:

1- When the robot moves, control input (u_t) is calculated.

$$u_t = \begin{bmatrix} -\frac{v_t}{w_t}\sin(\alpha_{t-1}^r) + \frac{v_t}{w_t}\sin(\alpha_{t-1}^r + \ w_t \, \Delta t) \\ +\frac{v_t}{w_t}\cos(\alpha_{t-1}^r) - \frac{v_t}{w_t}\cos(\alpha_{t-1}^r + \ w_t \, \Delta t) \\ w_t \, \Delta t \end{bmatrix} \qquad \textbf{3-34}$$

2- New maximum likely pose of the robot is calculated.

$$\mu_t^r = \ \mu_{t-1}^r + \ u_t \qquad \textbf{3-35}$$

3- The effect of previous robot pose to the next robot pose and the effect of control inputs to the next robot pose are calculated to update uncertainty. To achieve this, one must obtain the Jacobian's below.

$$G_t = \frac{\sigma\ \mu_t^r}{\sigma\ \mu_{t-1}^r} = \begin{bmatrix} 1 & 0 & -\frac{v_t}{w_t}\cos(\alpha_{t-1}^r) + \frac{v_t}{w_t}\cos(\alpha_{t-1}^r + w_t\,\Delta t) \\ 0 & 1 & -\frac{v_t}{w_t}\sin(\alpha_{t-1}^r) + \frac{v_t}{w_t}\sin(\alpha_{t-1}^r + w_t\,\Delta t) \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{3-36}$$

$$V_t = \frac{\sigma\ \mu_t^r}{\sigma\ (v_t, w_t)} = \begin{bmatrix} \frac{-\sin(\alpha_t^r) + \sin(\alpha_t^r + w_t\,\Delta t)}{w_t} & \frac{\sigma x}{\sigma w} \\ \frac{+\cos(\alpha_t^r) - \cos(\alpha_t^r + w_t\,\Delta t)}{w_t} & \frac{\sigma y}{\sigma w} \\ 0 & \Delta t \end{bmatrix} \quad \textbf{3-37}$$

where;

$$\frac{\sigma x}{\sigma w} = +\frac{v_t}{w_t^2}(\sin(\alpha_t^r) + \sin(\alpha_t^r + w_t\,\Delta t)) + \frac{v_t}{w_t}(\cos(\alpha_t^r + w_t\,\Delta t)) \quad \textbf{3-38}$$

$$\frac{\sigma y}{\sigma w} = -\frac{v_t}{w_t^2}(\cos(\alpha_t^r) - \cos(\alpha_t^r + w_t\,\Delta t)) + \frac{v_t}{w_t}(\sin(\alpha_t^r + w_t\,\Delta t)) \quad \textbf{3-39}$$

4- After obtaining necessary Jacobian's, new covariance is calculated

$$\Sigma_t^{r,r} = G_t \Sigma_{t-1}^{r,r} G_t^T + V_t Q_t V_t^T \quad \textbf{3-40}$$

assuming;

$$Q_t = \begin{bmatrix} \sigma v_t & 0 \\ 0 & \sigma w_t \end{bmatrix} \quad \textbf{3-41}$$

So, update operation for robot motion is completed.

5- Assume *M + L* features observed in a sensor data set. *M* of these features are already observed before and *L* of them are observed for the first time.

Step 5 to step 9 must be operated for all *M* features.

For feature *i*, the agent has the observation:

$$z_t^i = \left[r_t^i, \theta_t^i, \rho_t^i, \alpha_t^i\right]^T \quad \textbf{3-42}$$

Each landmark has a unique correspondence number denoted by *i*.

6- Predict the distance parameters using map data

$$\delta_t^i = \left[\delta_t^x \delta_t^y \delta_t^z \delta_t^\alpha\right]^T \quad \textbf{3-43}$$

where;

$$\delta_t^x = x_t^i - x_t^r \quad \textbf{3-44}$$

$$\delta_t^y = y_t^i - y_t^r \quad \textbf{3-45}$$

$$\delta_t^z = z_t^i - z_t^r = z_t^i \quad \textbf{3-46}$$

$$\delta_t^\alpha = \alpha_t^i - \alpha_t^r \quad \textbf{3-47}$$

$z_t^r = 0$, since the robot is at flat ground.

7- Predicted distance parameters must be transformed to predicted measurements with inverse geometric transformations

$$\hat{z}_t^i = \left[\hat{r}_t^i, \hat{\theta}_t^i, \hat{\rho}_t^i, \hat{\alpha}_t^i\right]^T \quad \textbf{3-48}$$

where;

$$\hat{r}_t^i = \sqrt{(\delta_t^x)^2 + \left(\delta_t^y\right)^2 + (\delta_t^z)^2} \quad \textbf{3-49}$$

$$\hat{\theta}_t^i = atan2\left(\delta_t^y, \delta_t^x\right) \quad \textbf{3-50}$$

$$\hat{\rho}_t^i = asin\left(\delta_t^z / \hat{r}_t^i\right) \quad \textbf{3-51}$$

$$\hat{\alpha}_t^i = \delta_t^\alpha \quad \textbf{3-52}$$

8- Error generated by sensor model must be inserted into a matrix. In this work Gaussian error model is assumed;

$$E_t^i = \begin{bmatrix} \delta r_t^i & 0 & 0 & 0 \\ 0 & \delta\theta_t^i & 0 & 0 \\ 0 & 0 & \delta\rho_t^i & 0 \\ 0 & 0 & 0 & \delta\alpha_t^i \end{bmatrix} \quad \textbf{3-53}$$

The above matrix points that every parameter of all variances for each feature is independent from each other.

9- Once true observation and predicted observation are obtained, the pose of the robot and the landmarks must be updated via covariance of the map and the sensor model. To achieve this, first Jacobian matrix H_t^i must be constructed.

$$F_i = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \quad \text{3-54}$$

3i-3 3N-3i

Here F_i provides the correspondence relation between sensor input and the landmarks and the required Jacobian function is as follows:

$$H_t^i = \begin{bmatrix} \sigma\hat{r}_t^i/\sigma x_t^r & \sigma\hat{r}_t^i/\sigma y_t^r & 0 & \sigma\hat{r}_t^i/\sigma x_t^i & \sigma\hat{r}_t^i/\sigma y_t^i & \sigma\hat{r}_t^i/\sigma z_t^i & 0 \\ \sigma\hat{\theta}_t^i/\sigma x_t^r & \sigma\hat{\theta}_t^i/\sigma y_t^r & \sigma\hat{\theta}_t^i/\sigma\alpha_t^r & \sigma\hat{\theta}_t^i/\sigma x_t^i & \sigma\hat{\theta}_t^i/\sigma y_t^i & 0 & 0 \\ \sigma\hat{\rho}_t^i/\sigma x_t^r & \sigma\hat{\rho}_t^i/\sigma y_t^r & 0 & \sigma\hat{\rho}_t^i/\sigma x_t^i & \sigma\hat{\rho}_t^i/\sigma y_t^i & \sigma\hat{\rho}_t^i/\sigma z_t^i & 0 \\ 0 & 0 & \sigma\hat{\alpha}_t^i/\sigma\alpha_t^r & 0 & 0 & 0 & \sigma\hat{\alpha}_t^i/\sigma\alpha_t^i \end{bmatrix} F_i \quad \text{3-55}$$

Where open forms of elements of 3-55 are given at Appendix B;

10- Operations in step 5 to step 9 must be processed and stored for each landmark that was observed at time *t* and also known previously. Newly observed features will be added to data package at the end. At this point stored matrixes will be merged in the same matrixes.

$$z_t = \begin{bmatrix} z_t^{i1} \\ z_t^{i2} \\ \vdots \\ z_t^{iM} \end{bmatrix} \quad \text{3-56}$$

$$\hat{z}_t = \begin{bmatrix} \hat{z}_t^{i1} \\ \hat{z}_t^{i2} \\ \vdots \\ \hat{z}_t^{iM} \end{bmatrix} \quad \textbf{3-57}$$

$$E_t = \begin{bmatrix} E_t^{i1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E_t^{iM} \end{bmatrix} \quad \textbf{3-58}$$

$$H_t = \begin{bmatrix} H_t^{i1} \\ H_t^{i2} \\ \vdots \\ H_t^{iM} \end{bmatrix} \quad \textbf{3-59}$$

On above equation, i^1, i^2 ... i^M denote the features observed and known previously.

11- *Kalman Gain* (*innovation*) is be calculated as follows:

$$K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + E_t)^{-1} \quad \textbf{3-60}$$

12- μ_t is updated for the observation part.

$$\bar{\mu}_t = \mu_t + K_t(z_t - \hat{z}_t) \quad \textbf{3-61}$$

13- Σ_t is updated for the observation part.

$$\bar{\Sigma}_t = (I - K_t H_t^T)\Sigma_t \quad \textbf{3-62}$$

14- Finally old data set shall be replaced with new one.

$$\mu_t = \bar{\mu}_t \quad \textbf{3-63}$$

$$\Sigma_t = \bar{\Sigma}_t \quad \textbf{3-64}$$

15- Now newly observed landmarks will be added to the landmark correspondence map and they will be placed to maximum likely pose in the map. In

addition to that, variance matrix must be extended for the new landmarks. As stated before there are *L* new landmarks and step 15 to step 19 must be processed for each of them. First μ_t^j must be merged to the end of μ_t.

$$\mu_t^j = \mu_t^r + \begin{bmatrix} r_t^j \cos(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \\ r_t^j \sin(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \\ r_t^j \sin(\rho_t^j) \\ \alpha_t^r + \alpha_t^j \end{bmatrix} \quad \textbf{3-65}$$

$$\bar{\mu}_t = \begin{bmatrix} \mu_t \\ \mu_t^j \end{bmatrix} \quad \textbf{3-66}$$

16- Now variance of new landmark must be calculated and merged to Σ_t. First Jacobians stated below must be obtained.

$$\frac{\sigma\mu_t^j}{\sigma\mu_t^r} = G_v^j = \begin{bmatrix} 1 & 0 & -r_t^j \sin(\alpha_t^r + \theta_t^j) \\ 0 & 1 & r_t^j \cos(\alpha_t^r + \theta_t^j) \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{3-67}$$

$$\frac{\sigma\mu_t^j}{\sigma z_t^j} = G_z^j = \begin{bmatrix} \sigma x_t^j/\sigma r_t^j & \sigma x_t^j/\sigma\theta_t^j & \sigma x_t^j/\sigma\rho_t^j & 0 \\ \sigma y_t^j/\sigma r_t^j & \sigma y_t^j/\sigma\theta_t^j & \sigma y_t^j/\sigma\rho_t^j & 0 \\ \sigma z_t^j/\sigma r_t^j & 0 & \sigma z_t^j/\sigma\rho_t^j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{3-68}$$

Where each element of (3-68) is presented at Appendix C;

17- Error matrix must be formed similar to (3-53)

18- Variance of new feature will be calculated and inserted

$$\Sigma_t^{j,j} = G_v^j \Sigma_t^{r,r} {G_v^j}^T + G_z^j E_t^j {G_z^j}^T \quad \textbf{3-69}$$

$$\Sigma_t^{j,r} = G_v^j \Sigma_t^{r,r} \quad \textbf{3-70}$$

$$\bar{\Sigma}_t = \begin{bmatrix} \Sigma_t^{r,r} & \Sigma_t^{r,lm1-N} & \Sigma_t^{j,r} \\ \Sigma_t^{lm1-N,r} & \Sigma_t^{lm1-N,lm1-N} & \Sigma_t^{m1-N,j} \\ \Sigma_t^{j,r^T} & \Sigma_t^{j,lm1-N} & \Sigma_t^{j,j} \end{bmatrix} \quad \text{3-71}$$

19- Finally data set must be recorded and by this way an iteration of EKF SLAM is completed.

$$\mu_t = \bar{\mu}_t \quad \text{3-72}$$

$$\Sigma_t = \bar{\Sigma}_t \quad \text{3-73}$$

Above equations are listed with similar order proposed by Thrun[2]. Pseudo code presented by Thrun is extended to higher dimensions for the purpose of this study.

3.2.1 Unused Parameters of Features

As stated previously; even though, it is possible to obtain 6 DoF pose from features, only four parameters were used in algorithm ($\hat{z}_t^i = \left[\hat{r}_t^i, \hat{\theta}_t^i, \hat{\rho}_t^i, \hat{\alpha}_t^i\right]^T$). In other words pitch and roll angles of the features were not used in algorithm. The reason of this issue is that since the robot is moving on a flat platform height, roll and pitch parameters of the agent are constant and known. Due to this reason; knowing roll and pitch angles of features does not provide any feedback for the pose of the robot. To explain this issue formally, one can claim that the effect of a parameter on other parameters can be defined by Jacobians and Jacobians of roll and pitch parameters are shown below.

$$\frac{\sigma\hat{\varphi}_t^r}{\sigma\varphi_t^i} = -1 \quad \text{3-74}$$

$$\frac{\sigma\hat{\omega}_t^r}{\sigma\omega_t^i} = -1 \quad \text{3-75}$$

However since $\hat{z}_t^r$, $\hat{\varphi}_t^r$ and $\hat{\omega}_t^r$ are constant and assumed to be *0*, the information derived from (3-75) is useless. So;

$$\frac{\sigma\hat{\mu}_t}{\sigma\varphi_t^i} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{3-76}$$

$$\frac{\sigma\hat{\mu}_t}{\sigma\omega_t^i} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{3-77}$$

Due to this reason $\sigma\varphi_t^i$ and $\sigma\omega_t^i$ are omitted from H_t and they are not used in EKF SLAM part of the simulation algorithm.

However it is invalid to claim that φ_t^i and ω_t^i are completely useless. As stated previously one of the most significant weaknesses of EKF SLAM is finding the true correspondence between the map and the observed data. At this point, φ_t^i and ω_t^i provide crucial information to find the true correspondence.

3.2.2 EKF SLAM with Point Landmarks

As stated previously a different version of EKF SLAM with 3D point landmarks will also be executed in this work. In the experiments part, once a rectangular plane is extracted from a distance map, four corners of this plane are used as landmarks and another run of EKF SLAM is executed with the corner landmarks. For EKF SLAM with point landmarks, the same equations explained in Section 3.2 were used; but the rows and columns of the matrixes that relate angular orientation of planes with other parameters are omitted. Unlike planes, points are composed of only 3 parameters. Sizes of the matrixes for N point features are presented in the table below.

Table 2–Data Structure of point feature EKF SLAM vs. plane segment feature EKF SLAM.

	symbol	size of matrix
Mean of parameters for point landmarks	μ_t	3N + 3 x 1
	$\mu_{landmark\ \varphi,\omega,t}$	N/A
Variance of parameters for point landmarks	Σ_t	3N + 3 x 3N +3
	$\Sigma_{landmark\ \varphi,\omega,t}$	N/A

CHAPTER 4

SENSOR AND SENSOR MODEL FOR FEATURE BASED EKF SLAM

A fundamental requirement for the reliable functioning of a mobile robot for SLAM is its ability to perceive its environment. This is achieved by using exteroceptive (or external) sensors (in contrast to proprioceptive or internal sensors). In other words, the agent is expected to collect data from the outer environment, it does not measure internal states.

A custom 3D IR distance scanner sensor is designed and produced for this work. This sensor is composed of a part that measures the distance on a beam and a mechanism that adjusts the direction of the beam to scan in 2 DoF. The mechanism and the sensor are controlled via PIC 16F877A. The sensor is designed and produced for this work and it will be referred with name IRSCAN through the rest of this work. A photograph of IRSCAN can be seen below.

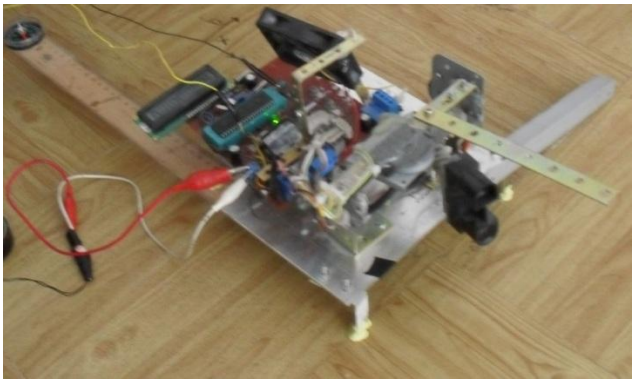


Figure 19 – A photograph of IRSCAN.

To create a better visualization for the reader, a set of data collected by IRSCAN from real indoor environment is illustrated at Appendix D.

4.1 Distance Measurement Tool

A SHARP GP2Y0A700 IR distance measuring sensor[36] is used to construct the distance map of IRSCAN. SHARP GP2Y0A700 measures the distance via triangulation method and it measures closer distance more accurate compared to farther. Response of the expected analog output of the sensor relative to the distance can be seen in Figure 20.

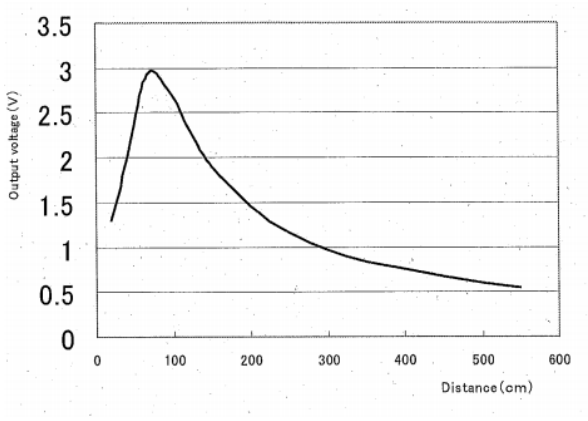


Figure 20 - Expected analog output of sensor relative to distance

The output graph may vary for each GP2Y0A700. Because of that, sensor needs calibration for more precise measurements. The sensor is slightly sensitive to color of measured objects. Due to this reason, calibration and tests in this work are done only with white objects. Sensor replenishes its output in every 20.2ms (maximum) and its total response time is about 25ms (maximum). This response time is the main restriction of the data collection speed.

4.2 Scanner

Direction of GP2Y0A700 is adjusted with two gearbox integrated stepper motors. The first motor controls the yaw angle and the second one controls the pitch angle. IRSCAN is able to create a 3D point cloud representation of the environment by acquiring *range* from SHARP GP2Y0A700 and controlling $pitch$ and yaw angles with stepper motors. Resolution angle in yaw direction is *0.1875°* and resolution in pitch direction is *0.635°*. The mechanical system is able to collect *660* samples in yaw direction and *130* samples in pitch direction; which means that a *130x660 (85.8 KP)* distance image can be constructed.

4.3 Control System

As stated previously IRSCAN is controlled via PIC 16F877A. A graphical structure of the electronic circuit of the control system of IRSCAN is illustrated below.

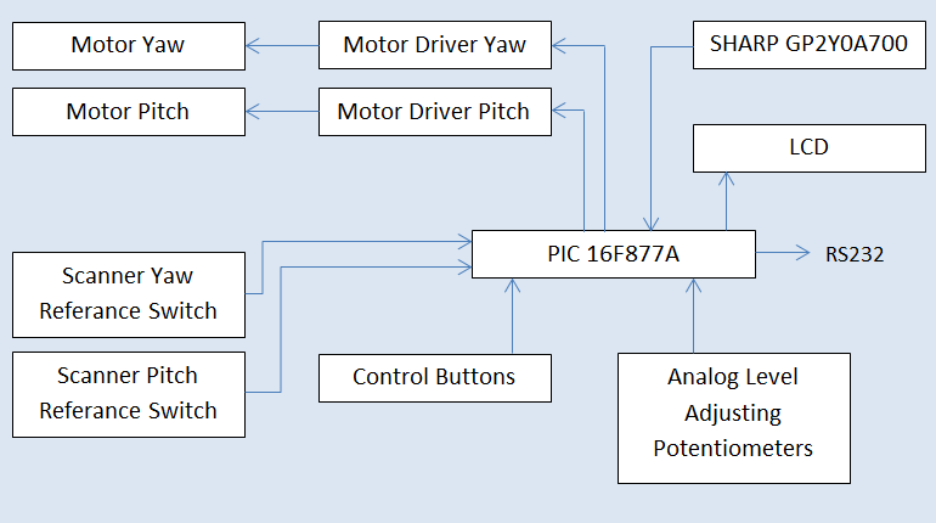


Figure 21 – Electronics diagram of IRSCAN.

16F877A drives the motors via a simple motor driver circuit (L293D). Scanner starting positions are located by scanner reference switches. GP2Y0A700 is connected to analog input of 16F877A via a coaxial cable. Two potentiometers

indicate lowest and highest possible voltage range and they are connected to analog voltage range setting pins of 16F877A. By this way resolution of analog reading shall be enhanced. Three control buttons and an LCD is used as an interface for controlling, testing and calibrating the sensor system. RS232 output of 16F877A is used for communication with PC. Sensor system sends output for each pixel to PC in real time; so throughout the operation time, sensor must be kept connected to PC via RS232 port and the PC must record RS232 data in real time.

4.4 Calibration

Intrinsic calibration refers to the process of setting the magnitude of the output (or response) of a measuring instrument or sensor to the magnitude of the input property within specified accuracy and precision. Extrinsic calibration refers to the process of relating the reference frame of the measurement instrument to another reference like the global coordinate frame.[37] Practically, it aims finding the location of the sensor coordinate frame with respect to some other reference frames. This is typically required in multi-sensor fusion where the data of different sensors has to be registered in a single coordinate frame. In this work, since there is no need for sensor fusion, intrinsic calibration methods will be used.

When hardware of IRSCAN is completed, analog level adjusting potentiometers are adjusted and fixed. Then, a white object is placed in front of the beam of the sensor and distance-voltage table is created for the sensor. Since the voltage response against distance is slightly different for each GP2Y0A700, this table must be renewed if the GP2Y0A700 unit is replaced with another one. The calibration graph for digital output is as follows:

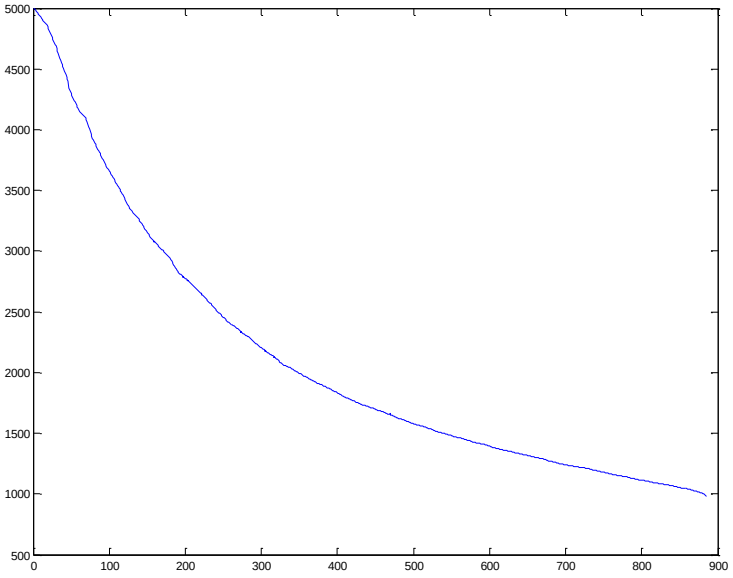


Figure 22 – Calibration graph of sensor – x-axis: Digital Output (10 bits). y-axis: Distance (mm)

After constructing analog conversion table, IRSCAN is placed in a room and point clouds of the walls of the room are constructed from several distances.

The point cloud of a wall is observed and a second set of calibration operation is performed depending on the output of IRSCAN. To do this IRSCAN is placed perpendicular in front of a wall and the wall is scanned at different distances such as 996mm, 997mm, 1396mm, 1696mm, 1996mm, 2296mm, 2596mm, 2896mm, 3196mm, 3496mm, 3796mm.

After scanning the wall, a function called Calibration Rate (CR) is created:

$$CR_{d_1}^{d_2} = \frac{1}{N}\sum_{n=1}^{N}\frac{Digitally\ Calculated\ Distance\ of\ Point\ n}{Measured\ Distance\ of\ Point\ n} \qquad \textbf{4-1}$$

In (4-1), CR denotes the Calibration Rate of points between d_1 and d_2 mm away from the sensor. N denotes the number of points between range of d_1 and d_2. This

calibration rate is used to calibrate output of the sensor with piecewise linear manner. To clarify, one can assume that there are 180 points between 1510mm and 1540mm and 195 points between 1540mm and 1570mm. So CR_{1510}^{1540} and CR_{1540}^{1570} are as follows respectively:

$$CR_{1510}^{1540} = CR_{1525} = \frac{1}{180}\sum_{n=1}^{180}\frac{Digitally\ Calculated\ Distance\ of\ Point\ n}{Measured\ Distance\ of\ Point\ n} \quad \textbf{4-2}$$

$$CR_{1540}^{1570} = CR_{1555} = \frac{1}{195}\sum_{m=1}^{195}\frac{Digitally\ Calculated\ Distance\ of\ Point\ m}{Measured\ Distance\ of\ Point\ m} \quad \textbf{4-3}$$

Once a measurement of 1530mm is collected, CR_{1530} can be found by simple weighting equation:

$$CR_{1530} = \frac{CR_{1525} * (1555 - 1530) + CR_{1555} * (1530 - 1525)}{1555 - 1525} \quad \textbf{4-4}$$

With more general formulation the linear weighting equation is as follows:

$$CR_x = \frac{CR_{x-k} * l + CR_{x+l} * k}{k + l} \quad \textbf{4-5}$$

Finally,

$$d_{calibrated} = d_{measured} * CR_{measured} \quad \textbf{4-6}$$

Resolution of this correction factor must be set to an optimum value. Even though increasing resolution rate seems a stronger method, it creates memorization of the calibration data and makes CR weaker. This noise will create a point cloud as illustrated bellow.

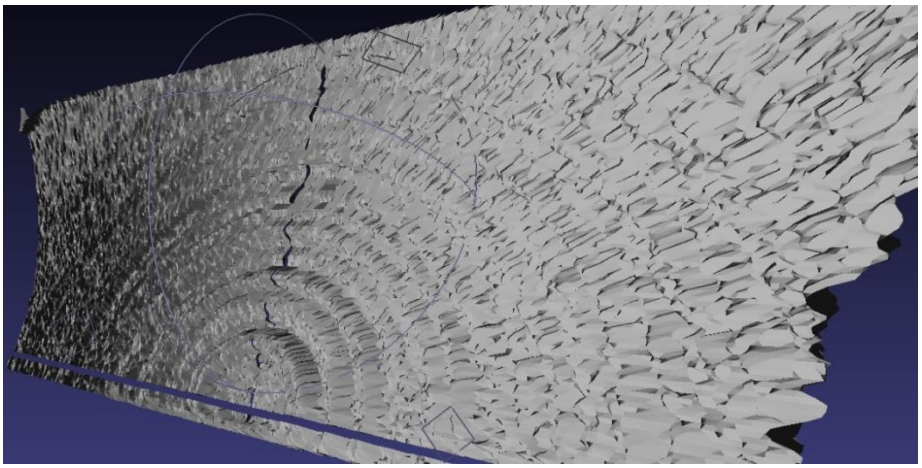


Figure 23 – CR function constructed with 5mm calibration resolution (constructed point cloud).

As the reader can verify, the planar surface of the wall is composed of rings because of calibration noise. The graph CR function for 5mm resolution is as follows:

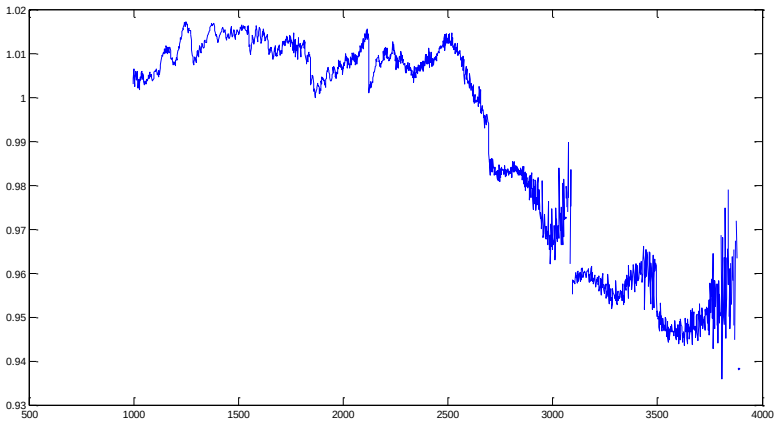


Figure 24 – CR function with resolution rate = 2mm

So it can be said that increasing calibration resolution to very dense values may create several problems as illustrated in Figure 23. In addition to this, one can sense

the noise of calibration data by looking at the CR function given in Figure 24 – CR function with resolution rate = 2mm.

On the other hand decreasing resolution weakens the effect of calibration process and leads jumps between crossing points to next resolution part. After several trials; 120mm resolution is decided to be used and a piecewise calibration method is implemented. The graph of piecewise CR is illustrated bellow.

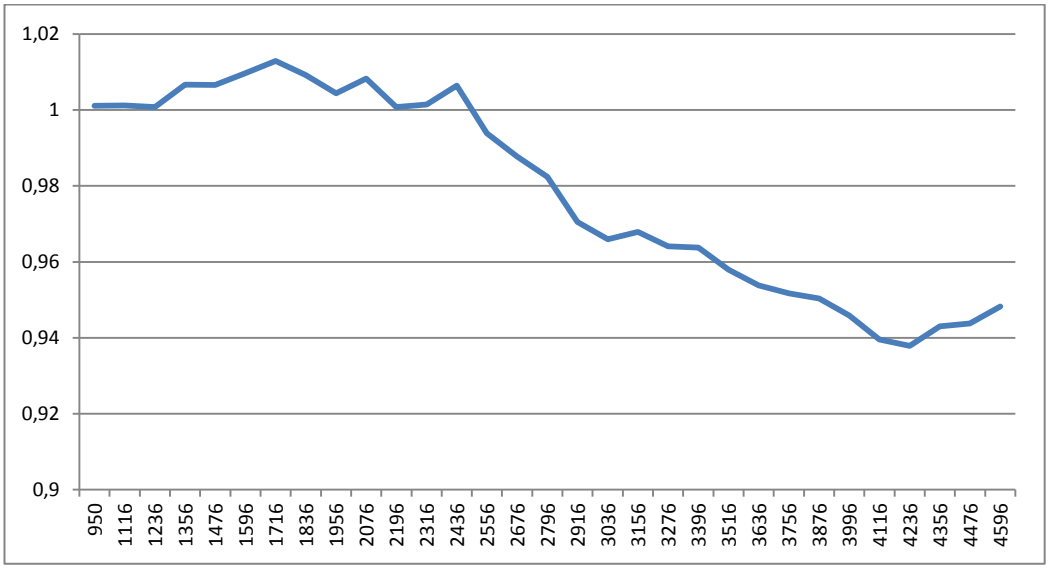


Figure 25 – Calibration Rate vs. distance (mm)

After calibration is completed the walls are reconstructed, some illustrations of the walls are as follows:

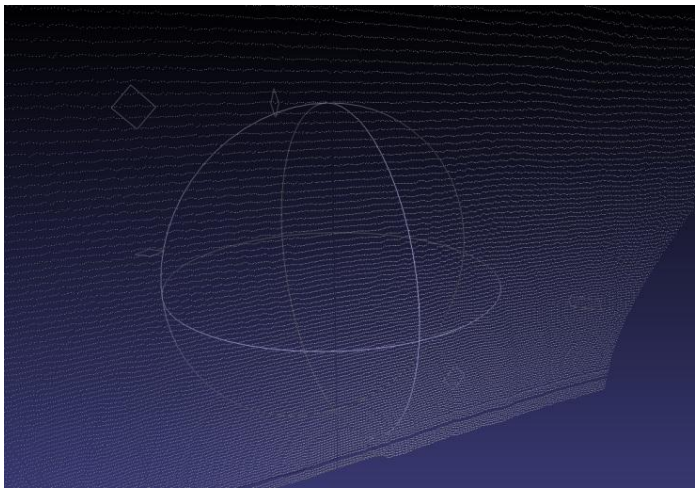


Figure 26 – Point Cloud of Wall from 997mm

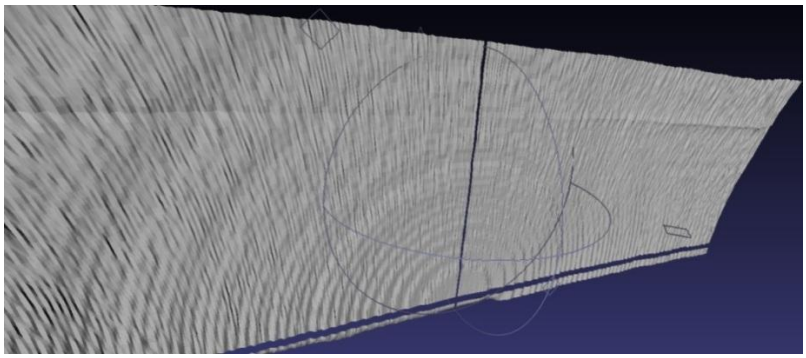


Figure 27 – Constructed Point Cloud of Wall from 997mm

Figure 28 – Constructed Point Cloud of Wall from 1396mm

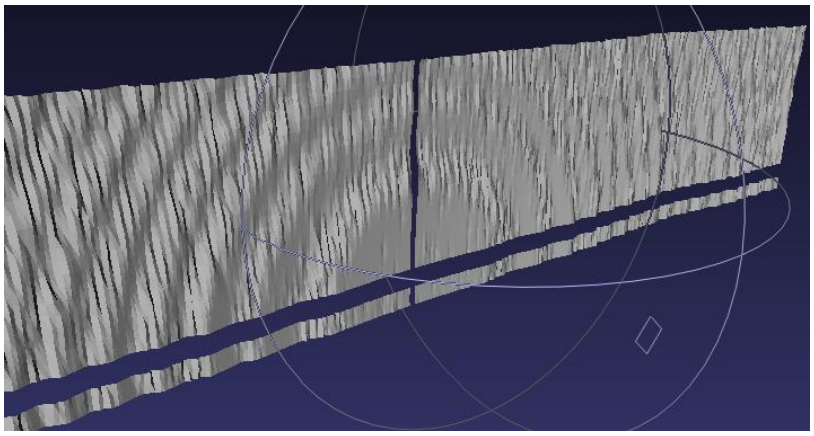


Figure 29 – Constructed Point Cloud of Wall from 1696mm

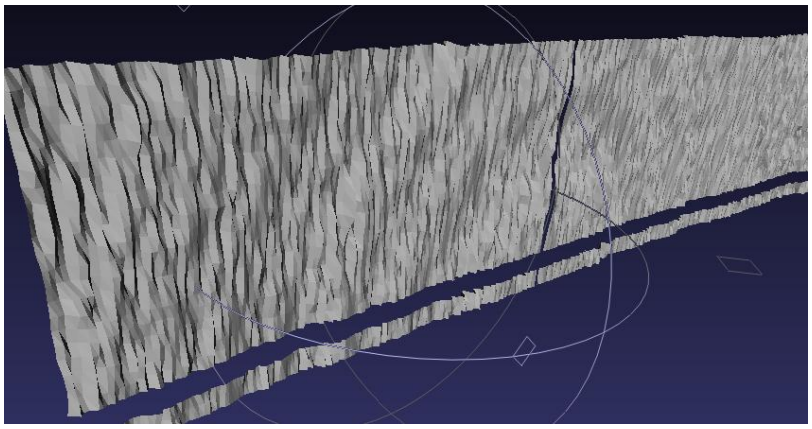


Figure 30 – Constructed Point Cloud of Wall from 2296mm

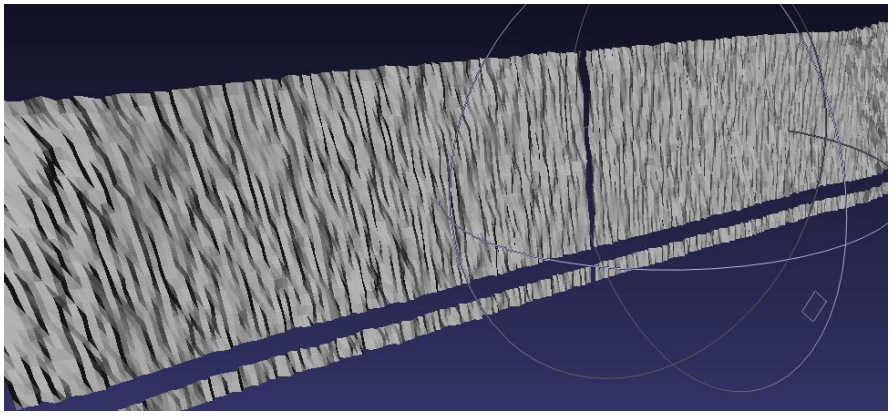


Figure 31 – Constructed Point Cloud of Wall from 2596mm

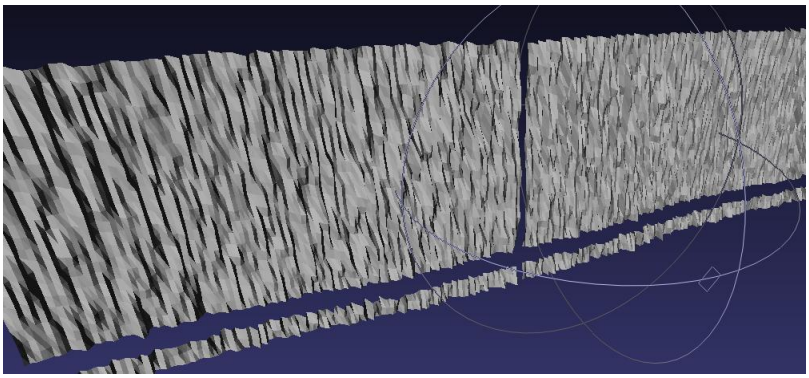


Figure 32 – Constructed Point Cloud of Wall from 2896mm

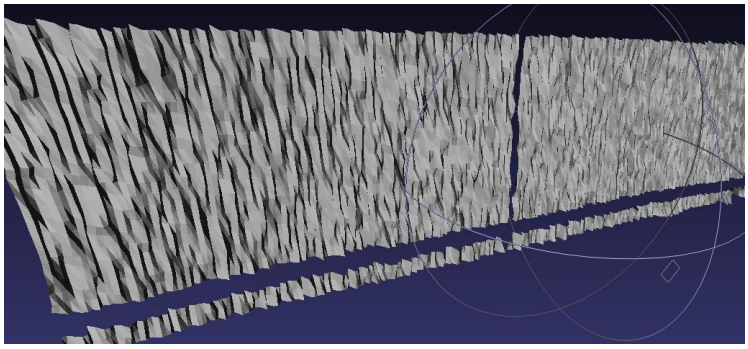


Figure 33 – Constructed Point Cloud of Wall from 3196mm

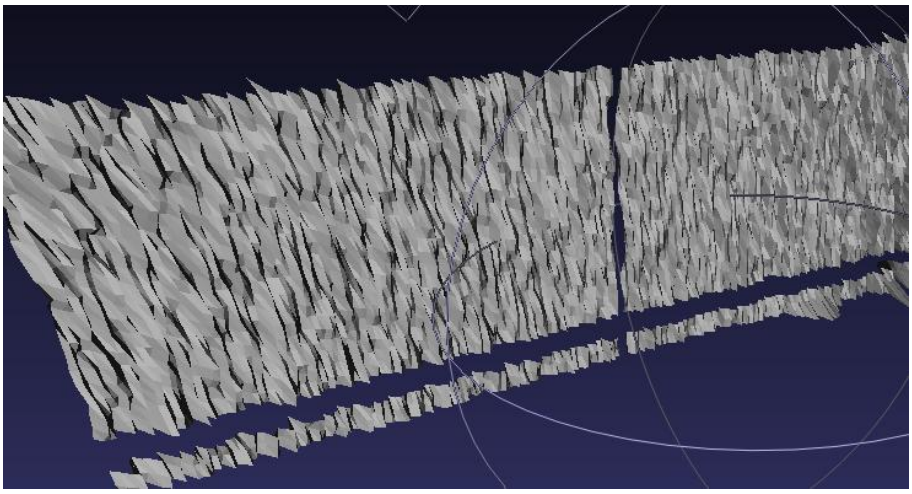


Figure 34 – Constructed Point Cloud of Wall from 3796mm

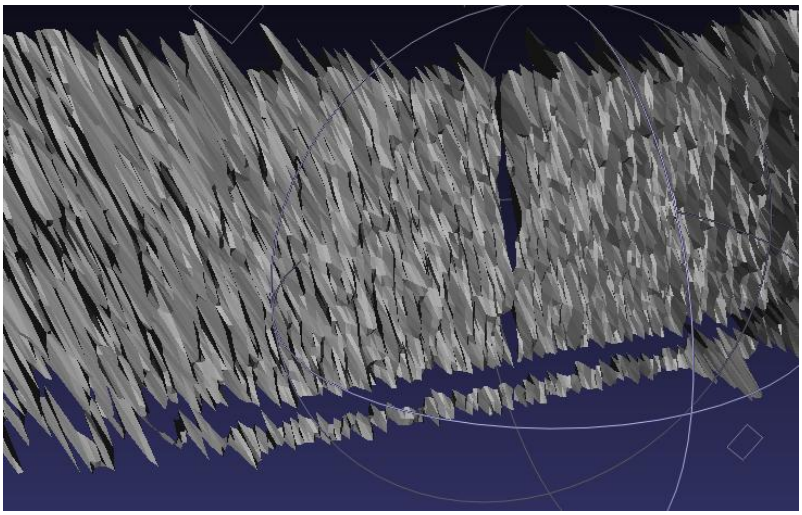


Figure 35 – Constructed Point Cloud of Wall from 4096mm

Figure 26 - Figure 35 show illustrations of a wall from several distances. As the reader can verify, even though calibration operation was implemented, there are still some patterns like circles on the wall (circular shapes are best visible in Figure 29). The reason of this circular pattern is that, GP2Y0A700 IR distance measuring sensor has a resolution for distance measuring operation. In other words GP2Y0A700 has a non-uniform distance vs. voltage output function and this effect make the sensor give specific output voltage values with higher probability. This impact is weakened by collecting more than one measurement for each point. For example, mean of 25 measurements is used for each point while calibrating the sensor and 25 sensor readings will be taken for other experiments too (unless otherwise stated). But even such a filter is used, there is still a non-uniform pattern in the point cloud of the wall and this effect will be tolerated in the conducted experiments with increased sensor variance.

Non-uniform pattern is getting weaker while the sensor system is getting farther from the wall, but this time a uniform noise is taking the place of non-uniform pattern. The reader can verify increased amount of noise at farther distances by

observing the figures above. Especially the change at the form of the noise between Figure 29 and Figure 35 can easily be visualized.

4.4.1 Variance of Observation Distance for a Single Point

IRSCAN will be used to extract several features for EKF SLAM, thus a noise model for the scanner system has to be presented. The first step of presenting a noise model shall be constructing a noise model for a single point of the sensor system. To achieve this IRSCAN is placed perpendicular in front of a wall and several sets of data were collected from different distances once more. Once the distance of the wall and the angular pose of the wall relative to IRSCAN are known, the true distance for each point can be calculated. Dividing the true distance to measured distance for each point, a set of histograms for different distances is obtained. The histogram set, which shows probability distribution function of error model, is presented at Appendix E. The resulting graphs that show average error rate and variance rate are as follows:

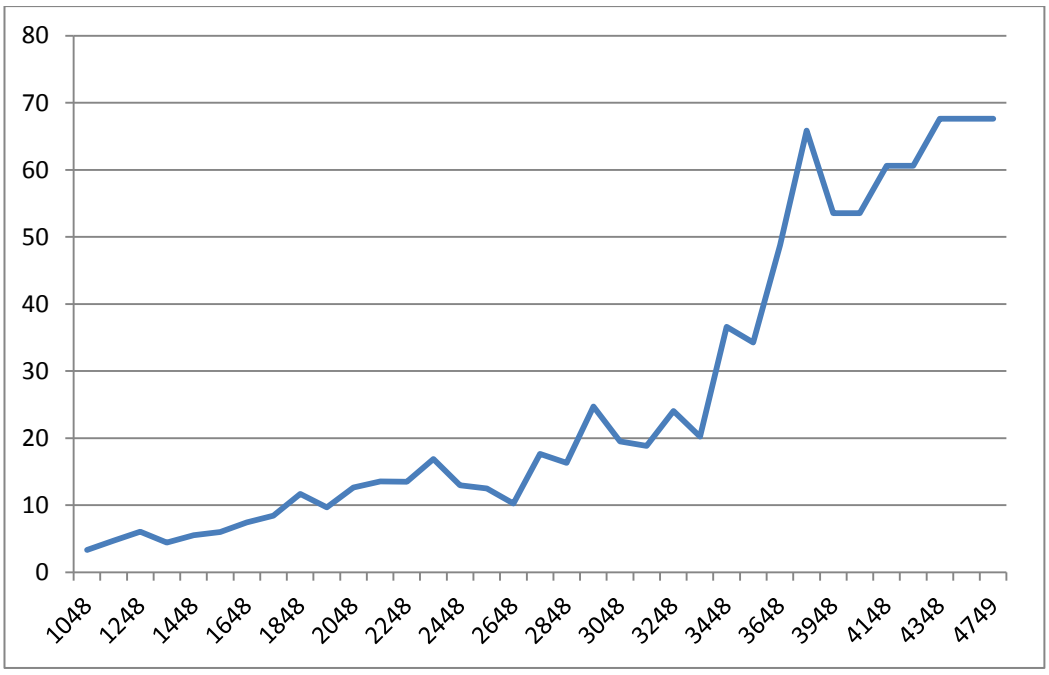


Figure 36 – Average Error Rate – x-distance (mm) vs. y-average error (mm)

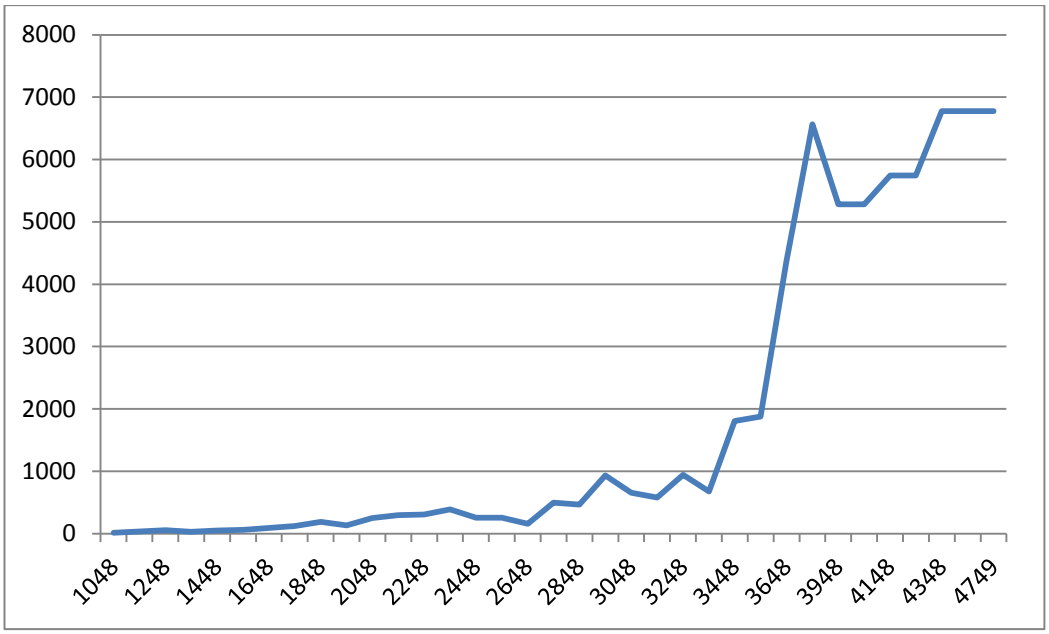


Figure 37 – Average of Square of Error– x-distance (mm) vs. y-square of average error (mm^2)

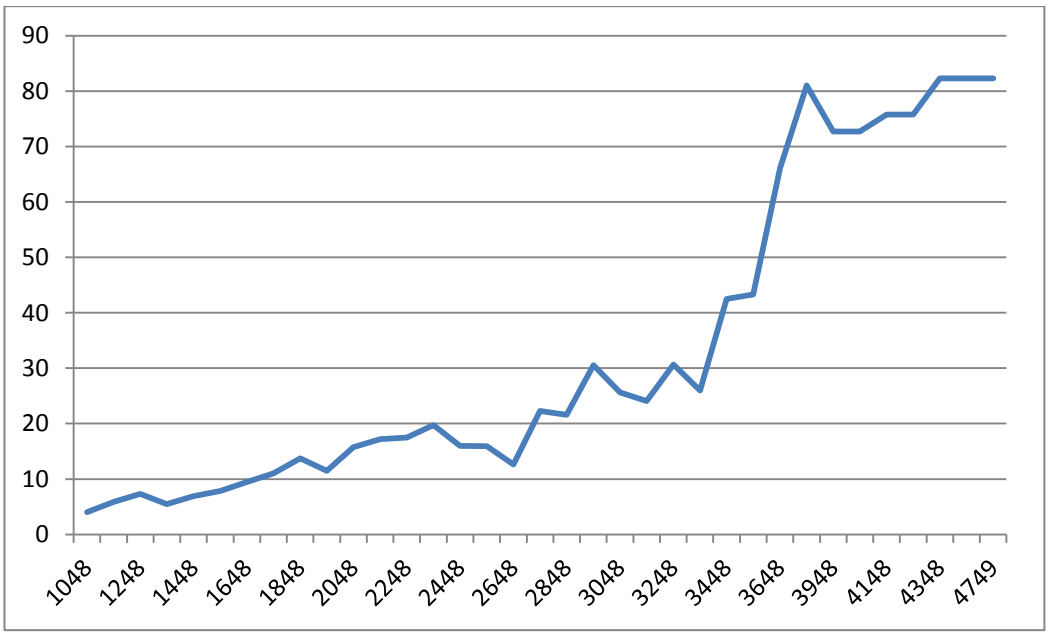


Figure 38 – Square Root of Figure 37– x-distance (mm) vs. y-error (mm)

Figure 38 is a sensible choice of standard deviation for a single point for IRSCAN. However; in this work, landmarks of 3D features will be chosen as edges of the object and it is not possible to choose the true pixel as edge for each case. Even though Figure 38 is a good choice of standard deviation for a random point, variance of range of CoM or edges will be calculated with different method at the following sections.

Variance of Angular Data for a Single Point:

There are two axes of rotations of the sensor: yaw and pitch axes. These two axes are controlled by two stepper motors and they can be directed independently. Due to this reason, variance of angular data of a point will be divided into variance of yaw and pitch axes and variance of these two axes will be assumed independent.

As stated before GP2Y0A700 measures the distance with triangulation method. In this method, the sensor emits a beam of light and the angle of reflection of light is measured via a sensor array. By this way the distance of reflection point can be calculated. A simple diagram that illustrates triangulation mechanism of sensor system is shown below.

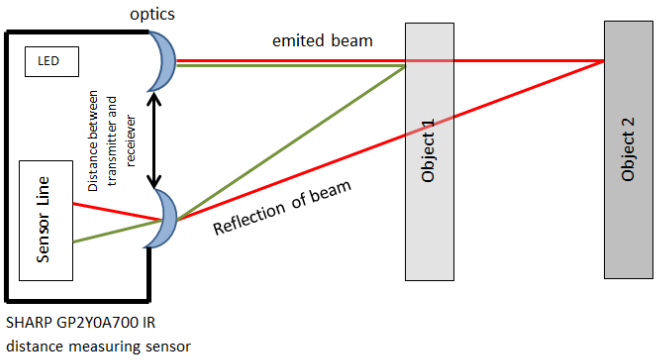


Figure 39 – Diagram that shows measureing mechanism of SHARP GP2Y0A700

As it can be seen in Figure 39, reflections from different distances illuminates different parts of the *sensor line*. Once the *distance between transmitter and receiver* is known, distance of reflection can be calculated via trigonometric operations.

At this point a problem arises because of the structure of optics. The emitter sends the beam with an offset and divergence rate. In other words, the beam has an offset radius comes from the radius of the emitter lens and has a divergence angle. A figure that illustrates an exaggerated form of radius of the beam is shown below.

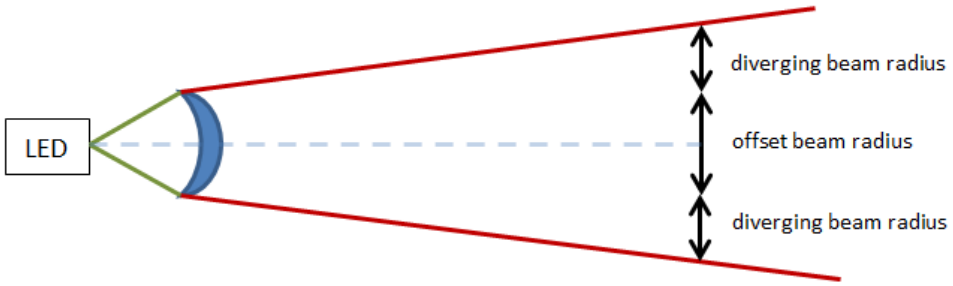


Figure 40 –Offset of GP2Y0A700.

Variance of a single point may be dominated by the mechanical error of the gearbox of IRSCAN. However, output of sensor needs to be processed to find interesting points (such as corners or edges) of features. In practical experiments, it is observed that the weakness of edges create an error while finding CoM or corners points of objects and this error dominates mechanical errors. Due to this reason variance of yaw angle will be calculated via this transition radius, which is an impact of divergence of the beam.

Variance of Yaw Axes:

As seen in Figure 40 radius of the beam linearly increases with distance. To find the parameters of this change, an object was placed in front of the sensor and a set of

data is collected. The sharp transition of edges is blurred because of the radius of the beam. Following figure shows the output of the experiment for the given object.

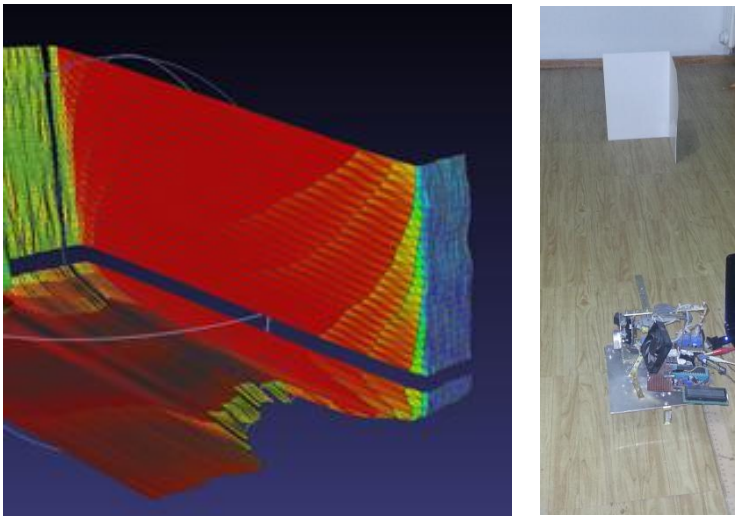


Figure 41 – Edge transition of calibration of IRSCAN from 1200mm.

Offset of radius is marginalized with increasing distance and the divergence angle dominates the blurring effect. In above illustration, points of measurement (each point in point cloud) are indicated with intersection of lines and there are 8 transition points at 1200mm object. Angular distance between points is 0.1875° and lens radius is *15mm*.

So for 1200mm, emitted beam radius is:

$$R_{1200} = \sin(8 * 0.1785) * 1200 = 29.90\ mm \quad \textbf{4-7}$$

By linear equation:

$$R_x = R_{offset} + R_{diverging\ effect} \quad \textbf{4-8}$$

$$R_x = 15mm + \sin(angle\ of\ diverging\ effect) * x \quad \textbf{4-9}$$

$$\sin(angle\ of\ diverging\ effect) \approx 0.0124 \quad \textbf{4-10}$$

$$R_x = 15 + 0.0124\,x \quad \textbf{4-11}$$

$$Ang(x) \approx \arcsin((15 + 0.0124\,x)/x) \quad \textbf{4-12}$$

(4-12) shows angular size of beam from *x* mm away. Since the edge is expected to be placed at center of transition part, actual divergence rate shall be chosen as half of (4-12).

$$Ang(x) \approx \frac{\arcsin((15 + 0.0124\,x)/x)}{2} \quad \textbf{4-13}$$

(4-13) will be used as standard deviation of yaw angle. This variance rate can be decreased via several data processing techniques. But, the goal of this part is finding a parameter that can be used as variance for edge detection algorithms. It is well known that standard deviation of angle is linearly related with angle of divergence of GP2Y0A700. However other parameters like the color of the object, color of background, angular pose of measurement surface and strength or bias of edge detection algorithms also dominates angular variance in both positive and negative directions. Since, using angle of divergence as standard deviation rate is believed to be a close value for possible worst case error angle; it is a sensible assumption that, this angle rate possibly covers the impact of different kinds of erroneous effects. In addition to this, choosing a defensive standard deviation is sensible for EKF SLAM. A monte-carlo analyzes will be executed in Section 5.9 to explain the reason of choosing a high value of standard deviation.

Variance for Pitch Axes:

A similar method will be used to find standard deviation of pitch axes. However - different from yaw axes- there is another effect that can be observed for this case. As stated previously, the sensor system works with triangulation. Triangulation line

is posed in the same direction with pitch axes in IRSCAN and this creates an undesired pattern at the bottom transition parts of the objects. A diagram to visualize the reason of such effect is as follows:

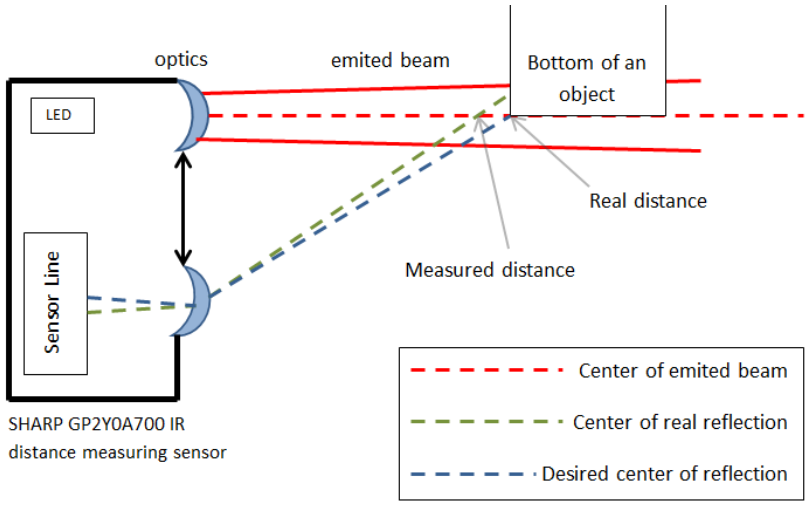


Figure 42– Diagram that show a bias effect of GP2Y0A700 for bottom part of obstacles.

In Figure 42 the intersection of dotted red and dotted blue line is the desired distance to be measured. However, there is a gap between center of emitted beam and center of reflection point; due to this reason the intersection of dotted green line and dotted red line is given as output (the point labeled *Measured distance*). Impact of such effect on output of point cloud is shown below.

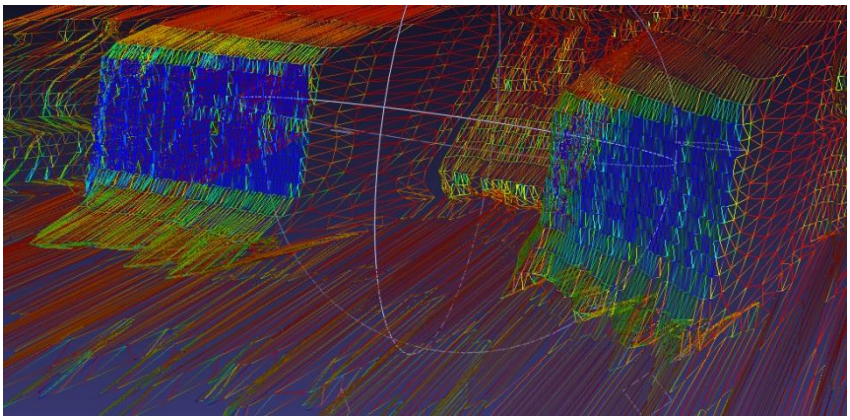


Figure 43 – Constructed point cloud of two object from 2427mm with errorneous bottom.

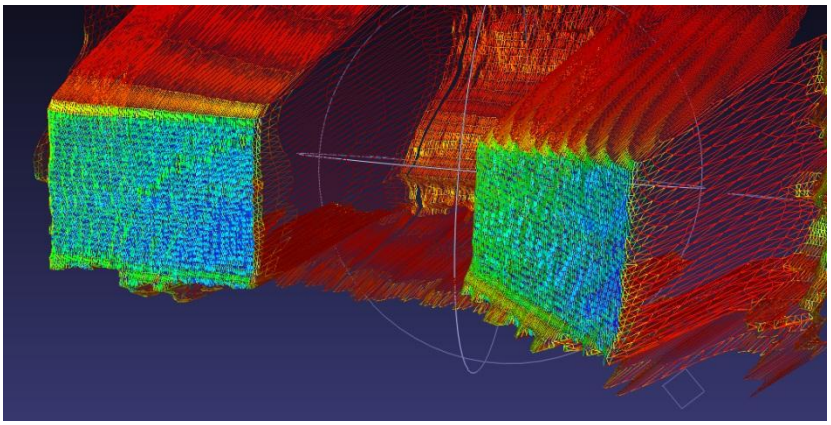


Figure 44 - Constructed point cloud of two objects from 1227mm with errorneous bottom.

Figure 43 and Figure 44 shows constructed point clouds of two planes standing at 115mm height from the ground. In both cases bottom side of the object is measured closer than the object itself.

Weakness of GP2Y0A700 for vertical transition is also stated in data sheet of sensor [36] and this wrong data may create several errors on measurement or mapping of the features standing high ground. Due to this reason this effect must be filtered in feature extraction algorithm. It is hard to determine a standard size for this wrongly

measured part because the size may change one or two pixels depending on the angular pose or color of objects. In addition to this, resolution of sensor system at pitch axis is lower than the resolution of yaw axis. At the following sections of the work, a convolution mask that removes the pixels close to edges will be presented. While choosing the size of such filter, above stated impact must be considered and the size of the filter must be set large enough to remove the erroneous parts.

In contrast of negative effects, the wrong measurement at bottom of the object can also be used in beneficial way to detect features. Features can be extracted via finding edges in the distance map. While executing such an algorithm outgrowth part at the bottom side of the objects creates a very strong pattern in the distance image. This pattern can be used to find the bottom part of high ground objects. In actual experiments edge detection algorithms never missed bottom parts of objects standing a few centimeters higher from ground. Distance image of two objects in Figure 43 is illustrated below.

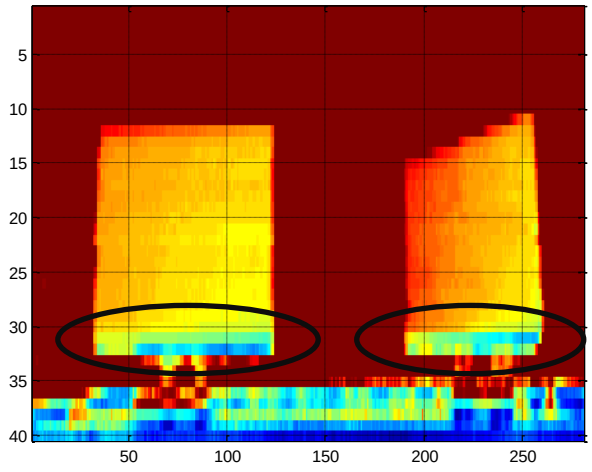


Figure 45 – There is a three pixel size pattern at bottom side of object. - Color is scaled as red is farther green is middle and blue is closer - The noisy zone below 35th line is ground and shall be omitted.

As seen in Figure 45, bottom parts of the objects are noisy and this high frequency part is never missed by edge detection algorithms throughout the experiments.

A similar effect holds for upper side of the object in reverse direction. In theory GP2Y0A700 tends to measure upper side of the objects farther than actual distance. However since background of object is farther too, this effect does not create a significant wrong measurement.

A method was proposed to determine a standard deviation value for yaw axes of IRSCAN. A similar method will now be used to find the variance of pitch axes. Following figure is the top part of an object measured from 1633mm away.

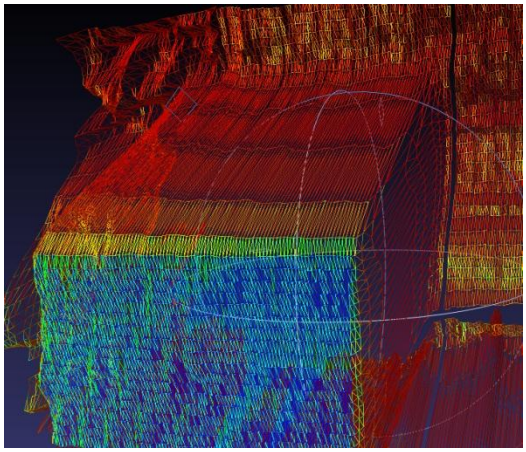


Figure 46 – top of the object is 1633mm away from the sensor

There are 5 transition points at the top of the object in Figure 46. The pitch angle resolution is 0.635°. So, for 1633mm, radius is:

$$R_{1633} = \sin(5 * 0.635) * 1633 = 90.43\ mm \quad \textbf{4-14}$$

$$R_x = R_{offset} + \sin(angle\ of\ divergence) * x \quad \textbf{4-15}$$

$$\sin(angle\ of\ divergence) \approx 0.0462 \quad \textbf{4-16}$$

$$R_x = 15 + 0.0462\ x \qquad \text{4-17}$$

$$Ang(x) \approx \arcsin((15 + 0.0462\ x)/x) \qquad \text{4-18}$$

The reason of transition of (4-12) to (4-13) holds for this case too. So (4-18) will be divided by two.

$$Ang(x) \approx \frac{\arcsin((15 + 0.0462\ x)/x)}{2} \qquad \text{4-19}$$

Just as done at yaw axis, (4-19) will be used as standard deviation in pitch axis.

4.4.2 Segmenting the Object and Fitting a Plane into Point Cloud

As stated previously, features will be extracted from point clouds for feature based EKF SLAM. This work focuses on mapping of indoor environments. Such environments are usually composed of planar structures such as walls, door, tables, closets and so on. Due to this reason working on planar features for EKF SLAM is a good choice, Thrun call this "structured environment assumption"[32]. In experimental part of this work, planes will be used as objects for several cases. So plane extraction algorithms will be used to find size and direction of the plane.

A set of data collected by sensor system is shown below.

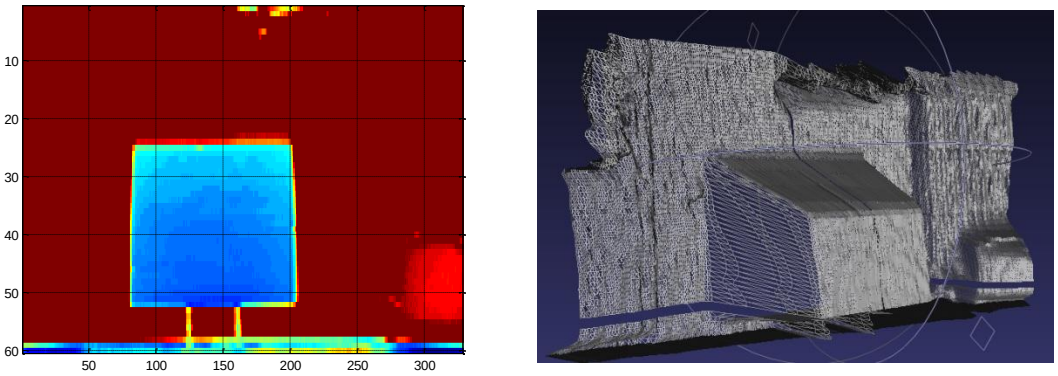
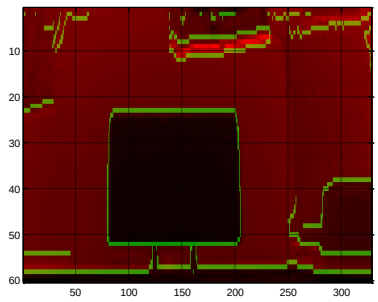
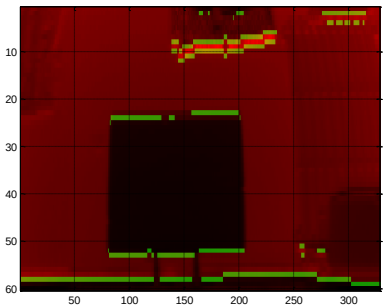
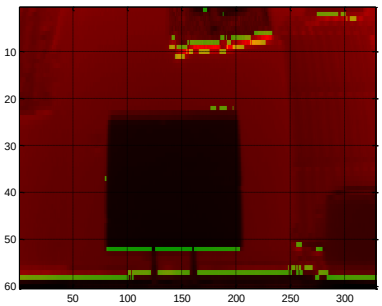
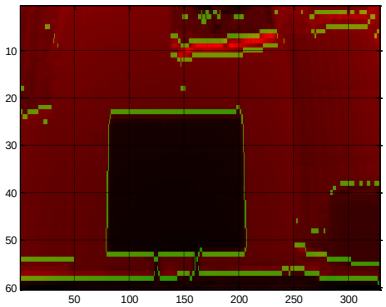
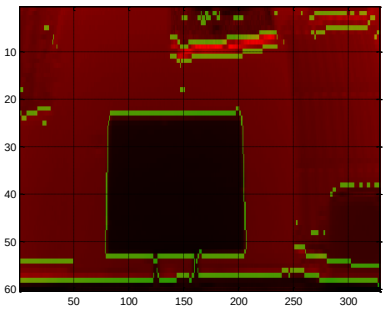
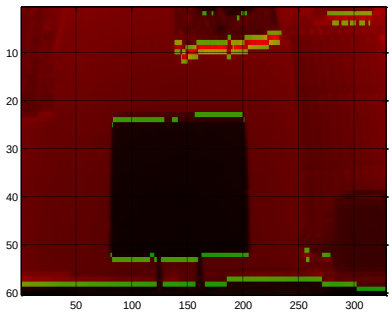


Figure 47 – Distance image of a planar segment (Right) – Constructed point cloud of the image (left)

When the point cloud and the distance map are obtained, edges of objects in distance map must be found. To do this standard edge finding codes of MATLAB 7.12.0 are used. These methods of finding edges were discussed at

APPENDIX A. Illustrations of output of several modes of edge finding codes of Figure 47 are given below:

Table 3 – 6 edge finding methods for Figure 47

	
Canny edge detector	Prewitt edge detector
	
Robert's edge detector	Zero cross edge detector
	
Laplacian of Gaussian edge detector	Sobel edge detector

When the figures in Table 3 are observed one can realize that Canny edge detector gives the best edge finding result for depth segmentation. In addition to this Zero cross and Laplacian of Gaussian edge detectors gives similar output and they are both strong for segmentation. In this work all of the above stated edge detectors will be superposed and the resulting edge image will be used. An illustration of final edge image is shown below.

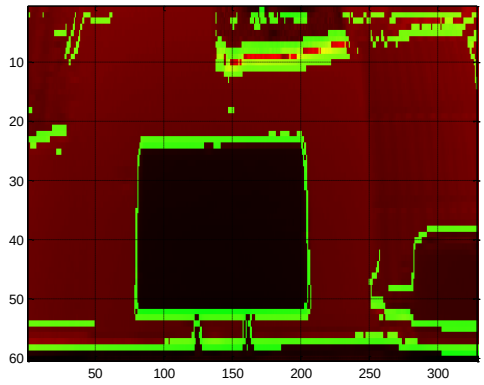


Figure 48 – Edge Image Superposed by Distance Image for Figure 47

Figure 48 is the overlapped edge image. It is known that, performance of sensor system may degrade at edges as stated previously (Section 4.4.1). Due to this reason, edges of the image are extended via a simple convolution algorithm and the extended version of edge image is obtained as follows:

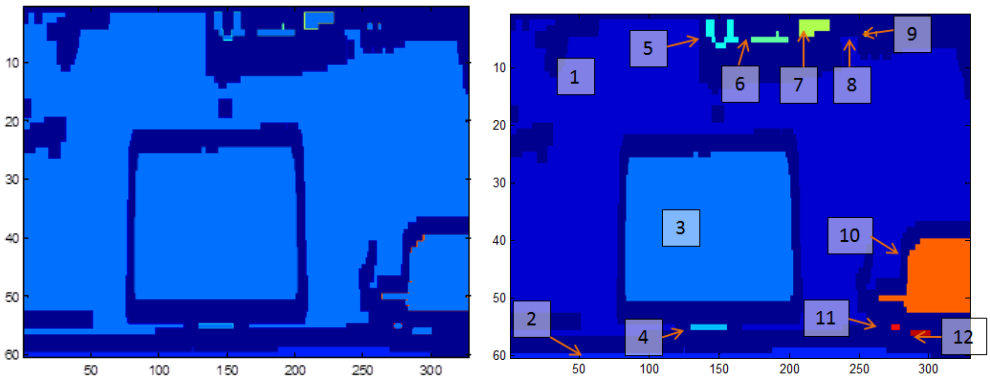


Figure 49 – Extended edge image of Figure 48 (right) – Watershed segmented version of the image (left)

Figure 49 (right) is extended edge map of distance image and areas enclosed by edges must now be segmented. To achieve this, watershed algorithm is used on extended edge image and Figure 49 (left) is obtained.

As discussed previously, EKF SLAM needs correspondence between the segments and the landmarks. In this work; since no auto-correspondence operation is executed, the correspondence between segment#3 and object#1 is manually inserted into the algorithm. After that point, segment#3 will be named as object#1 and the data of object#1 will be processed. The point cloud of object#1 will be formatted to a suitable data type for EKF SLAM. First of all, the point cloud of object#1 shall be extracted from distance image and reconstructed.

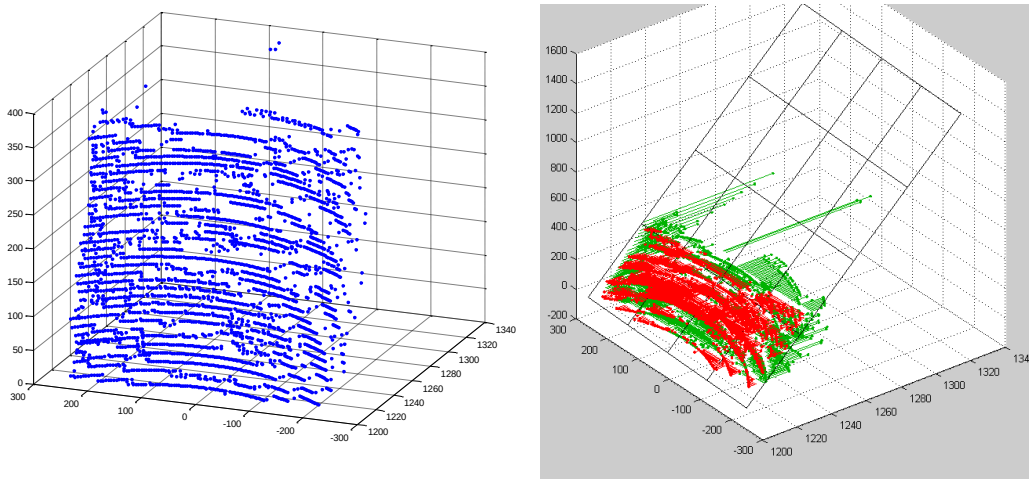


Figure 50 – Point Cloud of object#1 at Figure 49 (right) – A plane is fit into point cloud (left)

A plane must be fit into the point cloud as shown in Figure 50 (left). For this fitting operation, a code that uses least square of errors was chosen.[38]

Normal vector of the plane is *[0.9963 0.0089 -0.0851].* Yaw angle of direction of plane (direction#1), relative to sensor is found *0.5135°* (Actual angle is 0°). CoM#1 of plane relative to sensor axis is found as follows:

Table 4 – Position of CoM of object#1 at Figure 50

	measured	real
range of CoM#1 relative to sensor frame	1237.4mm	1241.9mm
yaw angle of CoM#1 relative to sensor frame	0.6182°	0°
pitch angle of CoM#1 relative to sensor frame	8.44°	8.89°

Here, CoM#1 represents center of mass of object#1. By this way three parameters for CoM#1 and one parameter for Direction#1 are found. To make these

parameters used in EKF SLAM, variance of the direction of the plane must also be obtained.

Variance for Plane Direction

Direction of the plane will be used as one of the main parameters of EKF SLAM. So, at this point the variance of the direction vector must also be presented. Graphs of standard deviation of the sensor system -for different ranges- are illustrated previously. Once the variance of each point is known, it is possible to calculate a variance for plane fit into the point cloud. There are several methods of fitting a plane to a point cloud. Weingarten[39] propose a method to fit a plane into a point cloud for EKF SLAM. Once variance for each point is known, it is also possible to determine a variance rate for the angular pose of the whole plane. However, response of GP2Y0A700 may depend on the color or the angular pose of the object. In addition to this, IRSCAN has a resolution of depth and this effect may create undesired circular shapes on the resulting distance image (an example of circular shapes is illustrated at Figure 29). Additionally, edges of the object are blurred by the divergence effect. Due to this reason instead of complete theatrical calculations an experiment based method will be chosen to determine a variance for angular pose and it will be calculated with the following formulation.

1- Error rate of the direction vector is inversely proportional with the average distance of points. Assume that direction vector will be created via two points.

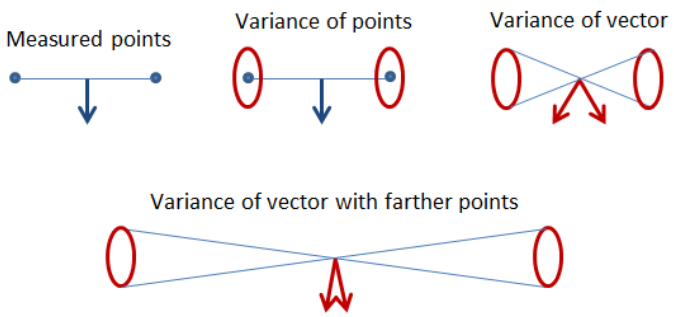


Figure 51 – Diagram that illustrates variance of direction relative to distance between points

Figure 51 shows the change of the variance of the direction vector of a line that was fit on two points (variance of points did not changed). As it can be seen from the figure, while the distance between points is increasing, the amount of change in angular direction relative to the variance of points is decreasing too. A similar situation also holds for plane fitting operation. Once the variance of single points does not vary; more precise angle of plane is expected with farther average distance of points. So it is expected that, the standard deviation of the angular direction is inversely proportional with the width of the plane.

2- One can realize that certainty of the range of the points is proportional to the certainty of the direction of the plane. A simple illustration of this issue is presented below.

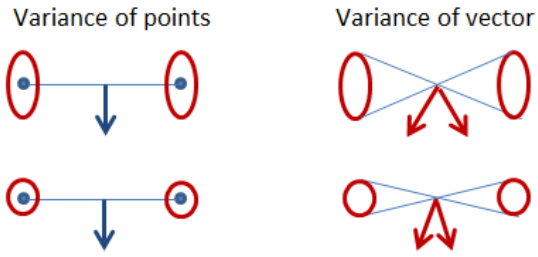


Figure 52 – Diagram that illustrates variance of direction relative to variance of range of points.

In many cases, resolution of the sensor dominates error of angular pose and resolution is –not completely but- closely linear with distance. So standard deviation of direction vector is closely proportional with distance of CoM.

So, standard deviation of direction angle (SDDA) for a rectangle is:

$$SDDA \propto \frac{r}{w} \qquad \textbf{4-20}$$

Let;

$$SDDA = c * \frac{r}{w} \quad \text{4-21}$$

Experimental results with IRSCAN shows that an average error of 0.5° is acceptable for a plane with *range = 1200mm* and *width = 503mm*.

However this is the result of several experiments conducted with one single white colored rectangular object. As stated previously, erroneous data collected from transition part of features, may interfere the direction of plane or may change CoM of plane in undesired way. In addition to this, size of the transition pixels makes it harder to mark the edges of the object. Because of that, the transition of background may also interfere with the extracted features for some cases.

When the amount of uncertainty is uncertain or may change depending on other conditions; it is a more sensible strategy to choose standard deviation at higher value, than expected uncertainty. In his work Thrun[40] states that adding a safety noise to sensor system may be used for increasing consistency of EKF SLAM for some cases. Such option may seem to degrade the performance of EKF SLAM, but it also avoids the algorithm from collapsing. In other words, if there is the risk of increasing uncertainty because of the environmental issues; using a safety margin for uncertainty is an acceptable strategy. Due to this reason, the standard deviation of object will be increased to twice value and *1°* will be used for the object given above. A monte-carlo analysis will be executed at Section 5.9, to analyze the effect of using a standard deviation value higher from the true model's value.

So, to find *c* for a set of data taken with *1200mm* range and width *503mm*;

$$SDDA = 1° = c * \frac{r}{w}$$

$$= c * \frac{1.2}{0.503} \quad \text{4-22}$$

$$c = \frac{1 * 0.503}{1.2}$$

$$c \cong 0.42 \quad \text{4-23}$$

The constant stated above will be used at experiments with following formulation;

$$SDDA = 0.42 * \frac{r}{w} \quad \text{4-24}$$

(4-24) is resulting output of experiments with a safety margin added on it. It is believed that standard deviation of direction angle can be decreased with stronger level of edge detection methods calibrated for sensor system and improved plane fitting operation which takes non-homogenous response of sensor into account (of course with the cost of increased need of processing power). However, at the experiments, the above equation produces standard deviations rates about 1°-2° and in practice, the resulting standard deviation rates provide beneficial contribution to EKF SLAM.

4.4.3 Finding Corner Points

Corner points are detected by an algorithm and the output results of the algorithm are never manipulated by supervisor. Below figure is an illustration of the point cloud of an object.

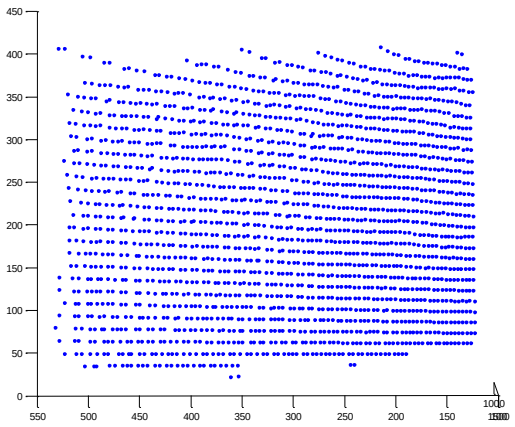


Figure 53 – Point cloud of an object collected by IRSCAN

Figure 53 is a point cloud illustration of an object. In the point cloud, each point has a yaw and pitch angle value. For each corner; a function is used to attach a number on each point (in total four numbers are attached to each point. Each number shows a candidate value to be one of the corners). The point with highest function output is chosen as corner.

$$Upper\ Right(Yaw\ Angle, Pitch\ Angle) = Yaw\ Angle - Pitch\ Angle \quad \textbf{4-25}$$

$$Bottom\ Right(Yaw\ Angle, Pitch\ Angle) = Yaw\ Angle + Pitch\ Angle \quad \textbf{4-26}$$

$$Bottom\ Left(Yaw\ Angle, Pitch\ Angle) = -Yaw\ Angle + Pitch\ Angle \quad \textbf{4-27}$$

$$Upper\ Left(Yaw\ Angle, Pitch\ Angle) = -Yaw\ Angle - Pitch\ Angle \quad \textbf{4-28}$$

The point that outputs maximum value for each function is chosen as candidate of related corner point. But to improve the performance of the algorithm two more corrections are carried out.

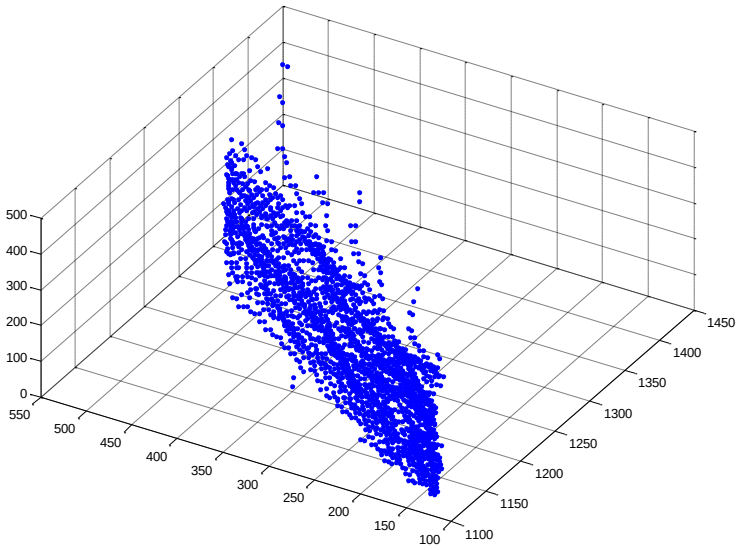


Figure 54 – Different view of Figure 53

Figure 54 is the same point cloud at Figure 53 but with different point of view. As the reader can verify the upper transition part is involved into the feature at top row. Similarly there are cases that transition of bottom row or left and right columns are included into the feature. Such interference will make the algorithm to find wrong values for the range of the edge or the corner points. Due to this reason the range of inner pixels is used as the range of corner point as shown below:

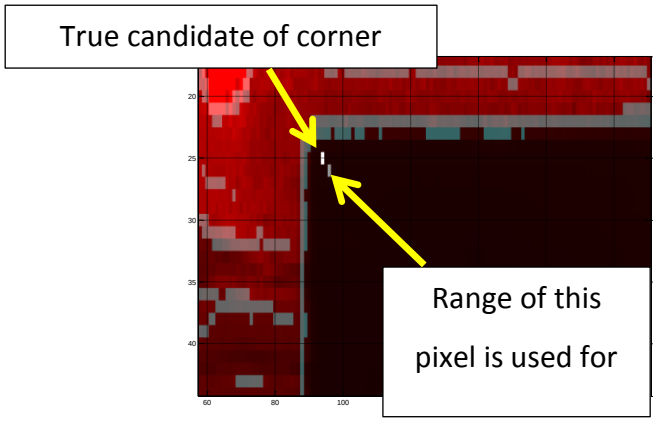


Figure 55 – True candidate of corner is at (25, 94). But the range of point at (26, 96) is used.

By the method illustrated at Figure 55, the range transition error close to edges is strongly degraded and a remarkable rate of performance improvement is obtained.

The second operation to improve the performance of corner selection is shifting the angular parameters of the pixels to outer part of object. As shown in Figure 48 and Figure 49, a convolution operation is carried on point cloud to remove transition part. Such action filters transition error of edges, but it also makes extracted plane segment smaller than true segment size. Due to this reason angular parameters of outer side of candidate will be used for chosen corner to fix this issue. An illustration is shown below.

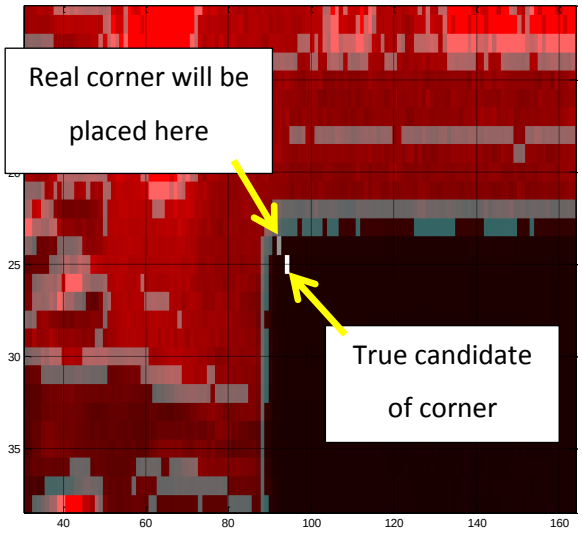


Figure 56 - True candidate of corner is at (25, 94). But the range data will be placed on (24, 92).

By using the correction shown at Figure 56 impact of convolution shown at Figure 48 and Figure 49 is degraded.

So (4-28) chooses candidate points. Let these candidate points are indicated with following coordinates in distance map:

$$Candidate\ Upper\ Right = (UR_{pitch}, UR_{yaw}) \quad \textbf{4-29}$$

$$Candidate\ Bottom\ Right = \left(BR_{pitch}, BR_{yaw}\right) \quad \textbf{4-30}$$

$$Candidate\ Bottom\ Left = \left(BL_{pitch}, BL_{yaw}\right) \quad \textbf{4-31}$$

$$Candidate\ Upper\ Left = (UL_{pitch}, UL_{yaw}) \quad \textbf{4-32}$$

Given the parameters of candidate corner points, the parameters of true corner points will be chosen as following simple shifting operations:

Parameters of upper right corner:

$$Range(UR) = Range(UR_{pitch} + 1, UR_{yaw} - 2) \quad \textbf{4-33}$$

$$Yaw\ Angle\ (UR) = UR_{yaw} + 2 \quad \textbf{4-34}$$

$$Pitch\ Angle\ (UR) = UR_{pitch} - 1 \quad \textbf{4-35}$$

Parameters of bottom right corner:

$$Range(BR) = Range(BR_{pitch} - 1, BR_{yaw} - 2) \quad \textbf{4-36}$$

$$Yaw\ Angle\ (BR) = BR_{yaw} + 2 \quad \textbf{4-37}$$

$$Pitch\ Angle\ (BR) = BR_{pitch} + 1 \quad \textbf{4-38}$$

Parameters of bottom left corner:

$$Range(BL) = Range(BL_{pitch} - 1, BL_{yaw} + 2) \quad \textbf{4-39}$$

$$Yaw\ Angle\ (BL) = BL_{yaw} - 2 \quad \textbf{4-40}$$

$$Pitch\ Angle\ (BL) = BL_{pitch} + 1 \quad \text{4-41}$$

Parameters of upper left corner:

$$Range(UL) = Range(UL_{pitch} + 1, UL_{yaw} + 2) \quad \text{4-42}$$

$$Yaw\ Angle\ (UL) = UL_{yaw} - 2 \quad \text{4-43}$$

$$Pitch\ Angle\ (UL) = UL_{pitch} - 1 \quad \text{4-44}$$

Variance for Corners and CoM of Surfaces:

As stated previously, just as IRSCAN has a mis-measurement rate; the environmental factors, the plane extraction algorithm, the edge detection algorithm and the corner detection algorithm may also lead errors on measurements due to mis-selection of edge lines or corners. Because of that, some of the variance rates will be revised via the results of experiments conducted with IRSCAN. So, standard deviation rates used for each parameter are revised or restated for unification as follows:

Variance of Range of Corner Points:

Experimental results indicate that Figure 38 is a good choice of standard deviation of range of a single point. However mis-selection of corner pixels may create large amount of errors (up to 70mm for some cases). Even though such large error rates occur rarely, they may make EKF SLAM algorithm collapsed. Due to this reason; to create a safety margin, values of Figure 38 will be doubled and used as standard deviation as illustrated at Figure 57. Increased amount of observation distance may seem to decrease performance of EKF SLAM, but as it will be explained in Section 5.5, increased rate of variance of observation distance is tolerable for many cases.

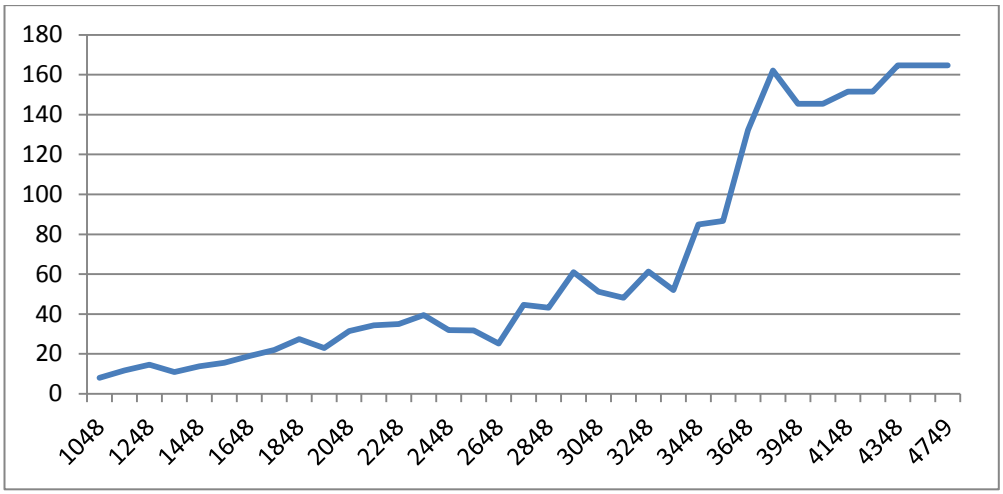


Figure 57 –Standard Deviation (mm) vs. measured distance (mm) - Standard deviation of observation distance for corner points.

Variance of Yaw Angle of Corner Points:

(4-13) is a sensible choice of standard deviation of yaw angle and it will be directly used for standard deviation value.

$$Standard\ Deviation\ of\ Yaw\ Angle\ of\ Corner = \frac{\arcsin\left(\frac{15+0.0124*\ range}{range}\right)}{2} \quad \textbf{4-45}$$

Variance of Pitch Angle of Corner Points:

(4-19) is a sensible choice of standard deviation of pitch angle and it will be directly used for standard deviation value.

$$Standard\ Deviation\ of\ Pitch\ Angle\ of\ Corner = \frac{\arcsin\left(\frac{15+0.0462*range}{range}\right)}{2} \quad \textbf{4-46}$$

Variance of Range of CoM:

CoM is the average of parameters of a number of points, so it is expected a stronger rate of certainty for range of CoM compared to range of corners. However

interference of transition part and range resolution of sensor also creates error on range of CoM. An illustration is shown below:

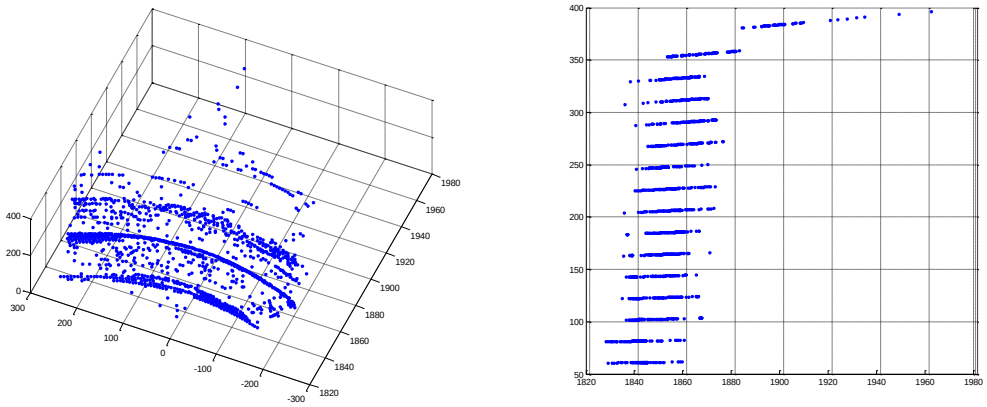


Figure 58 – Two illustrations of a point cloud of a planar object (left image is top view of point cloud; right image is side view of point cloud). As the reader can verify the resolution of the sensor may dominate placement of CoM since the points are grouped around a biased range (left). In addition to this, there is a group of transition pixels, that some of them are farther than 100mm from the true surface of the plane (right). These points lead an additional error on the calculated range of CoM.

Experimental results shows error rates of CoM is about half of error rate of corner points for a single object. So Figure 38 (at page 68) will be used as standard deviation of range.

Variance of Yaw Angle of CoM:

(4-13) is a sensible choice of standard deviation of yaw angle of a single point. The experimental results shows that, error rate of the yaw angle of CoM is about half of the error rate of corners. Actually this is consistent with theory, Angular pose of CoM is expected to be close to center of four corner points for a plane segment, so with an intuitive perception, the standard deviation of angular pose of CoM is expected to be the half of standard deviation of corners.

$$Standard\ Deviation\ of\ Yaw\ Angle\ of\ CoM = \frac{\arcsin\left(\frac{15+0.0124*range}{range}\right)}{4} \quad \text{4-47}$$

Variance of Pitch Angle of CoM:

Similar conditions of variance of the yaw angle also hold for variance of the pitch angle. For this case half of the standard deviation of pitch angle of corners will be used.

$$Standard\ Deviation\ of\ Pitch\ Angle\ of\ CoM = \frac{\arcsin\left(\frac{15+0.0462*range}{range}\right)}{4} \quad \text{4-48}$$

CHAPTER 5

EFFECTS OF PARAMETERS ON SLAM AND SIMULATION RESULTS

As stated previously an algorithm of EKF SLAM was coded in this work via MATLAB, which can be used for both real sensor data or in a simulation environment. In this part; first, the simulation system will shortly be explained, then the results of several monte-carlo runs will be discussed and the effects of parameters of locomotion system and sensor system will be presented.

5.1 Manual Run of Simulation

In this part, the simulation system will be introduced and a manual run will be presented and explained.

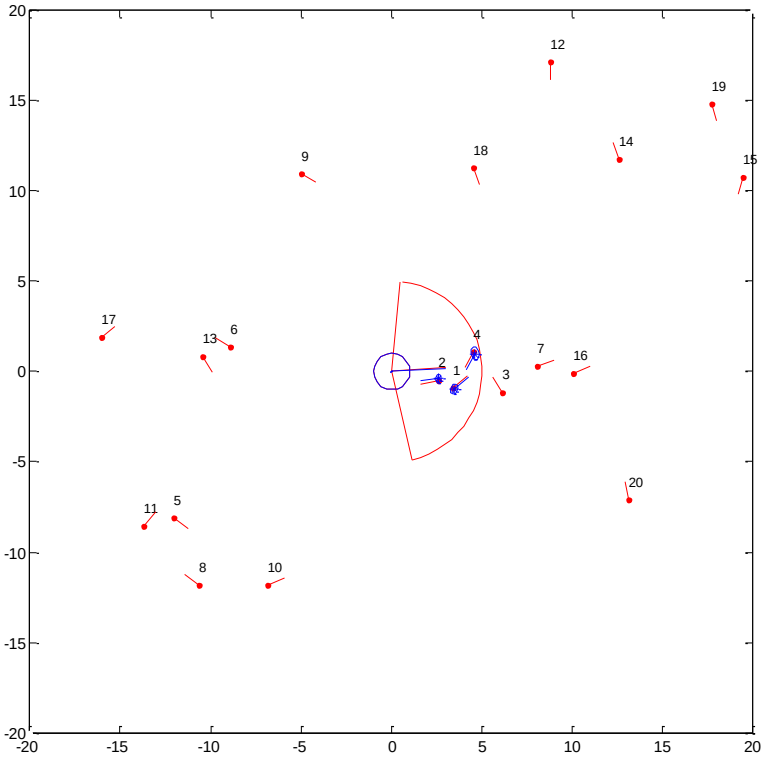


Figure 59 – Simulation#1 Figure#1

The robot and landmarks can be seen in Figure 59. True pose of the robot and the features are indicated by red and the belief pose are indicated by blue (initially, the belief and the real poses of the agent are overlapped). Laser sight of the agent is indicated by semicircle (red). Even though, the features are posed in 3D, height parameter is not illustrated in the figures. The points and stars represent position of features (CoM of plane segment) while the lines represent direction of the features (orientation of plane segment).

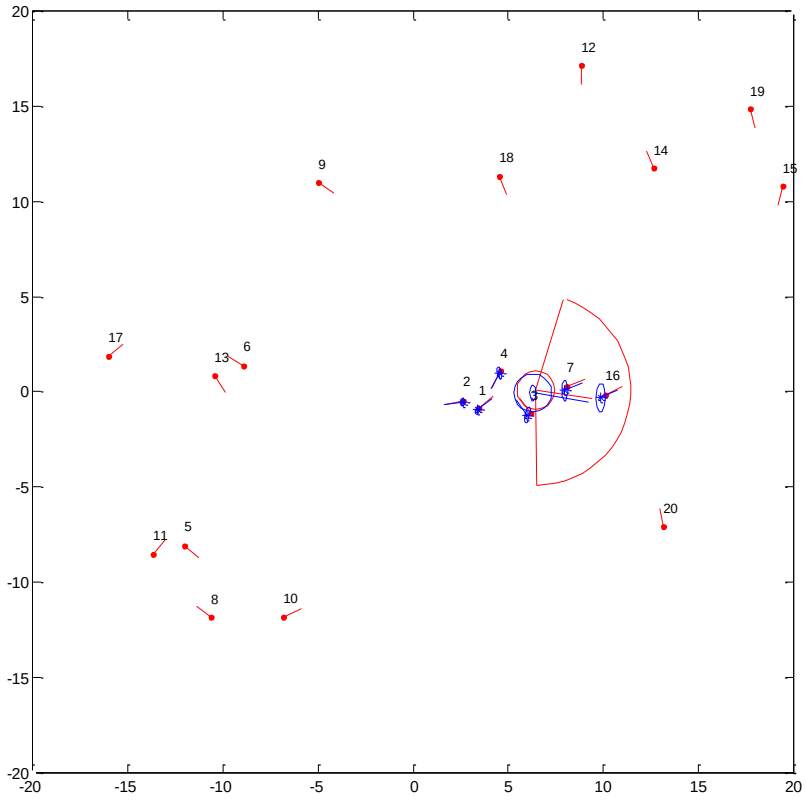


Figure 60 – Simulation#1 Figure#2

In this example the agent moves with a speed of *1m/s*. Figure 60 is taken at 7th second. While the robot moves, it adds new observed features to its correspondence map. Blue ellipses indicates x and y parameters of standard deviation and covariance. Each ellipse is drawn with the size of two standard deviations.

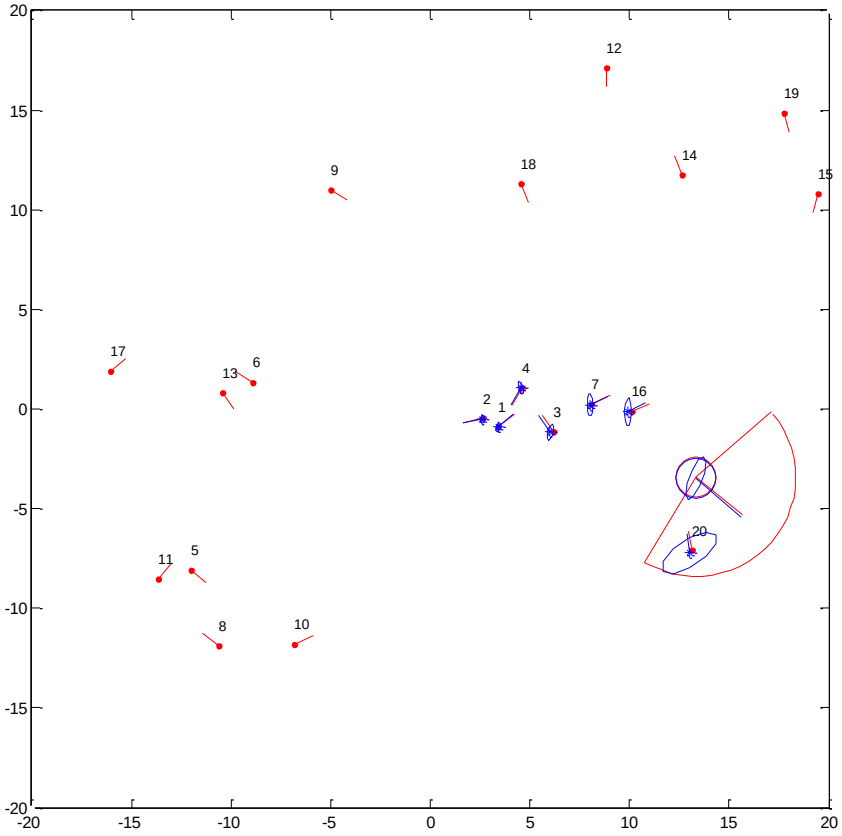


Figure 61 – Simulation#1 Figure#3

Figure 61 shows the map in 15th second. When the agent loses contact with features and moves without observing any objects, the variance of the robot grows faster.

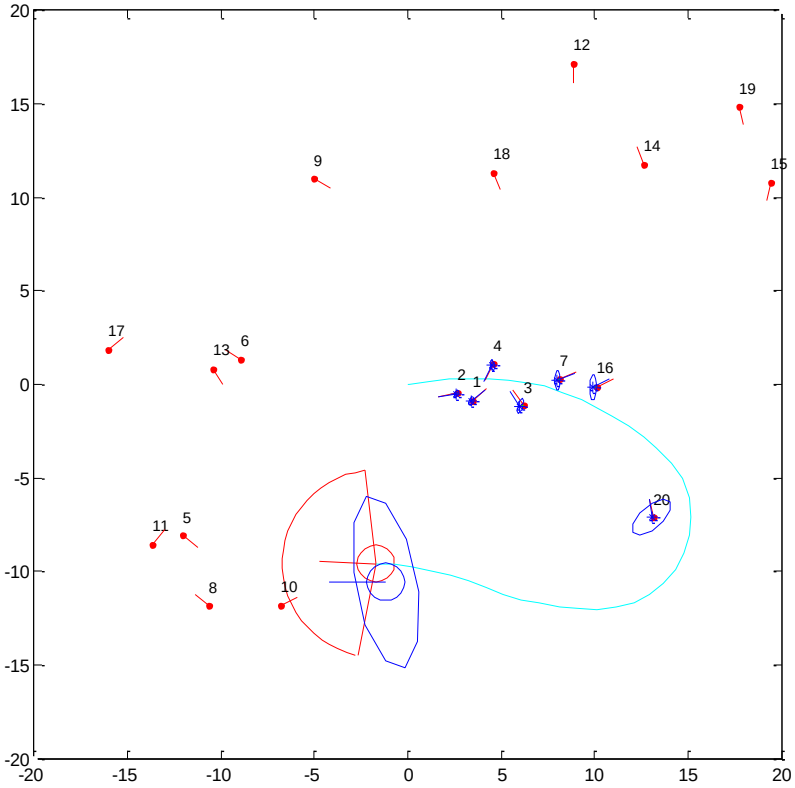


Figure 62 – Simulation#1 Figure#4

In Figure 62, the cyan arc indicates the real path of the agent. As the robot moves with no contact of features, the variance of the agent keeps grooving. Of course the amount of increase in variance depends on the precision of the locomotion system of the agent (In this example variance of agent is set relatively large values. By this way a stronger illustration of feedback of EKF SLAM will be created).

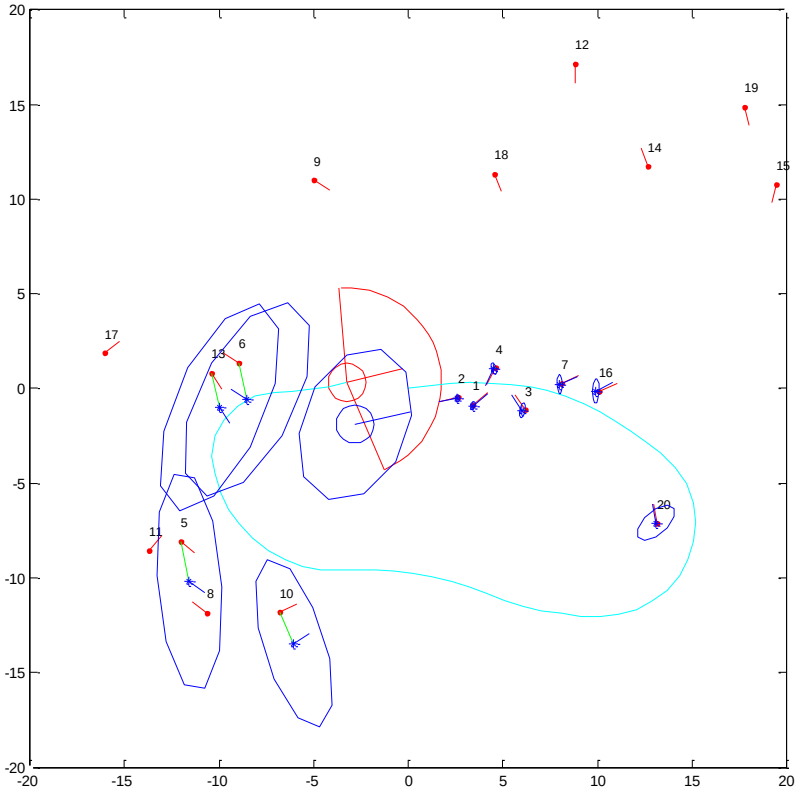


Figure 63 – Simulation#1 Figure#5

60th second is illustrated in Figure 63. Just as expected the effect of the large variance of the agent can also be seen as large variance of landmarks.

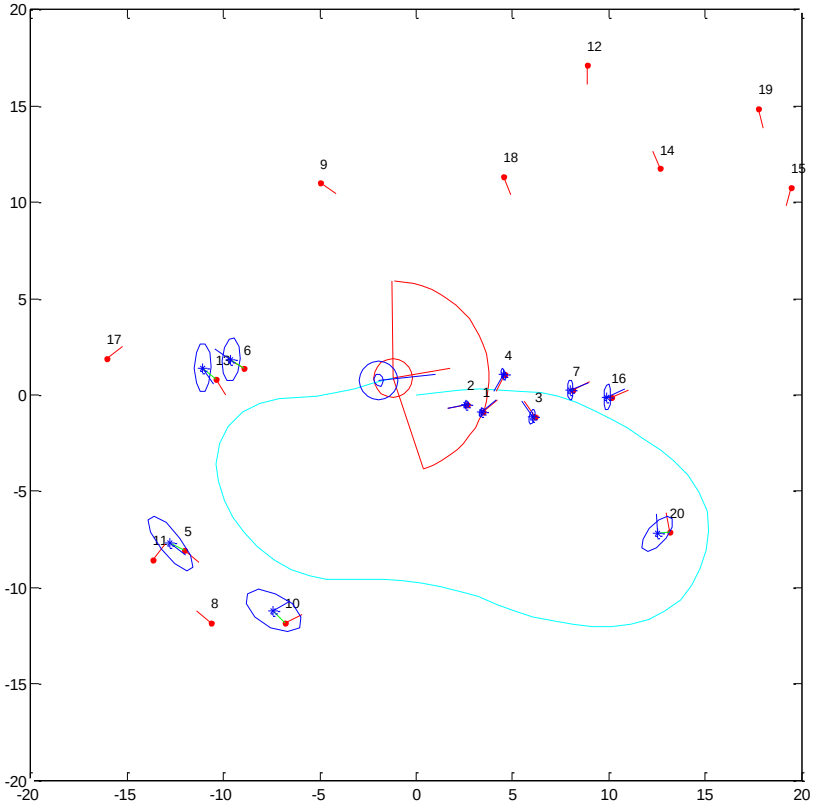


Figure 64 – Simulation#1 Figure#6

Just when the robot observes the initial landmarks and localizes its position in the map, all variance significantly decrease. As Figure 64 states, when precision of the agents pose improves, the precision of the previously observed landmarks are also affected in positive direction.

Another run for simulation will be illustrated bellow. This time correlation between landmarks will be emphasized. No direction vector for landmarks will be used.

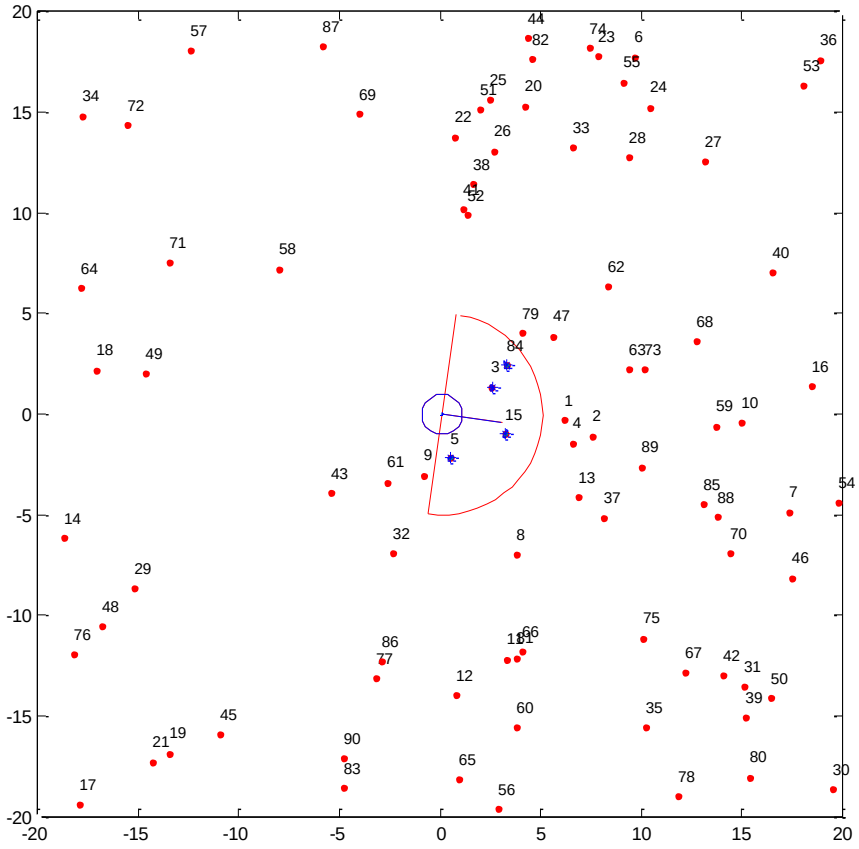


Figure 65 – Simulation#2 Figure#1

A new map is created. There are 90 landmarks in this map. The agent observes lm#3, lm#5, lm#15, and lm #84 at the beginning.

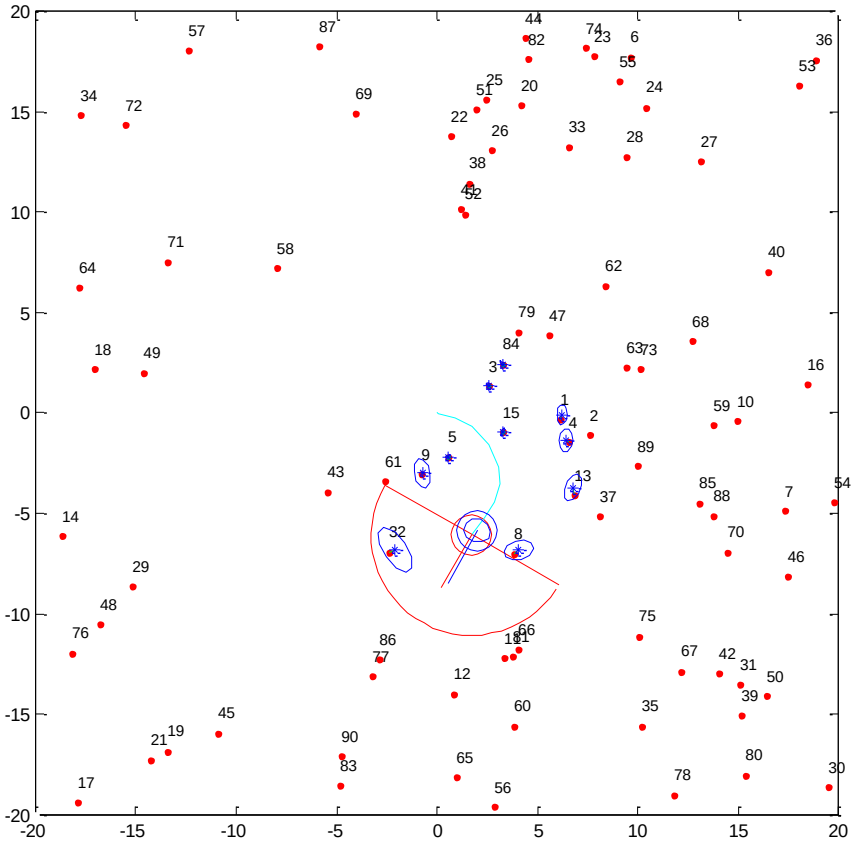


Figure 66 – Simulation#2 Figure#2

While getting farther from the starting point, the error rate and the standard deviation ellipses grow.

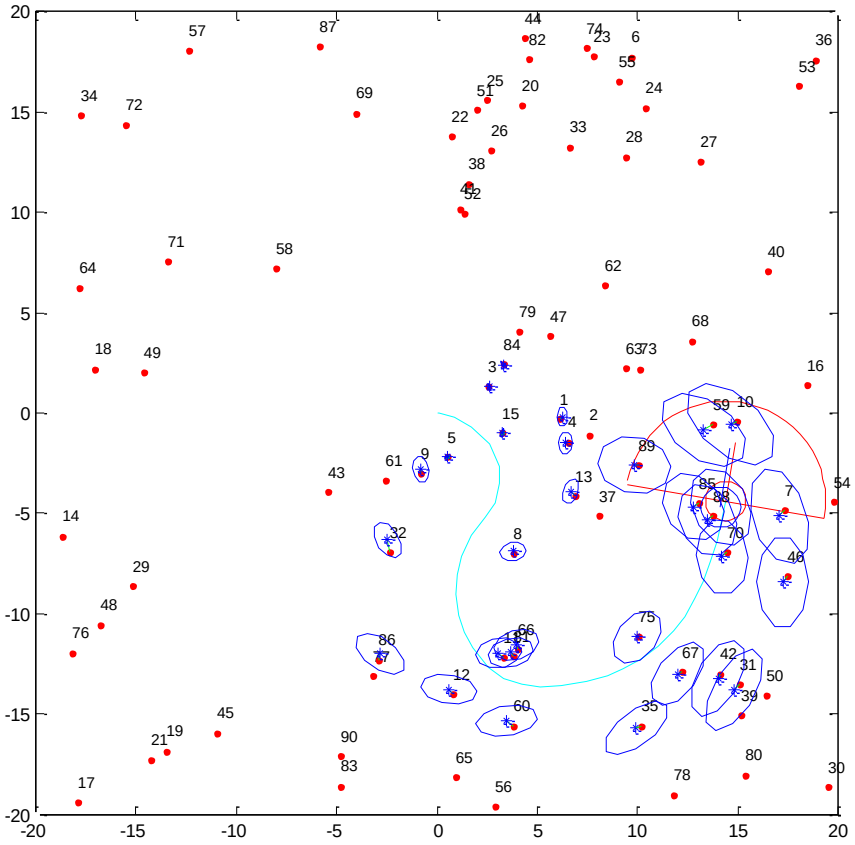


Figure 67 – Simulation#2 Figure#3

The agent passes through a close distance to the starting point but sees no initial landmarks. So the error rates keep increasing through movement.

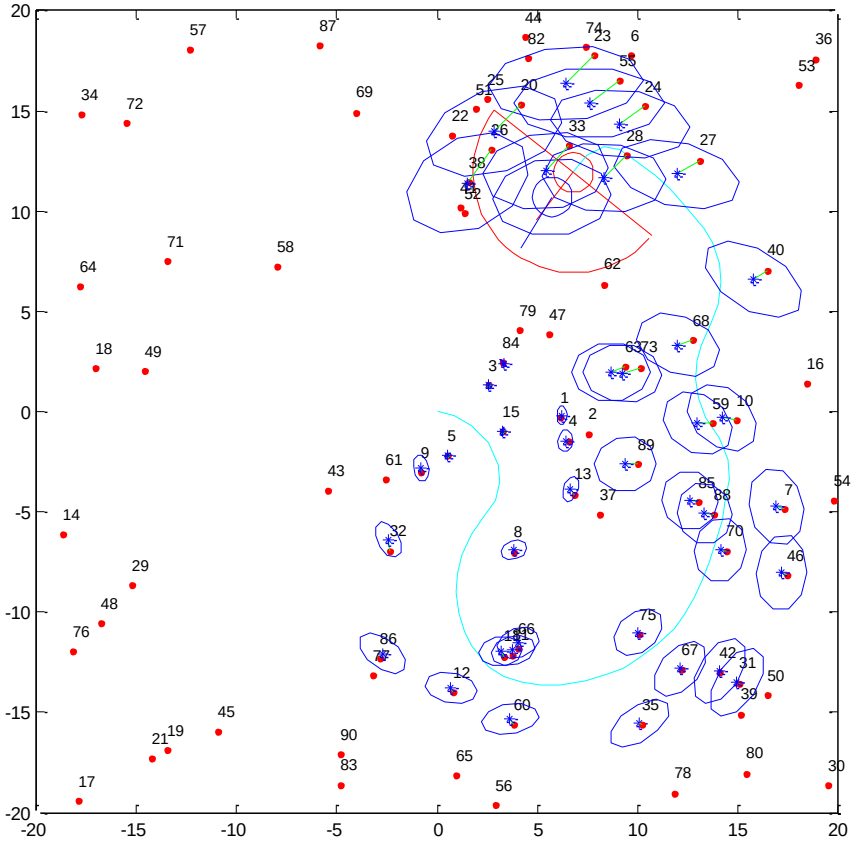


Figure 68 – Simulation#2 Figure#4

Since the robot has never seen a previously observed landmark, the grooving error rate of the robot pose dominates the error rate of newly added landmarks.

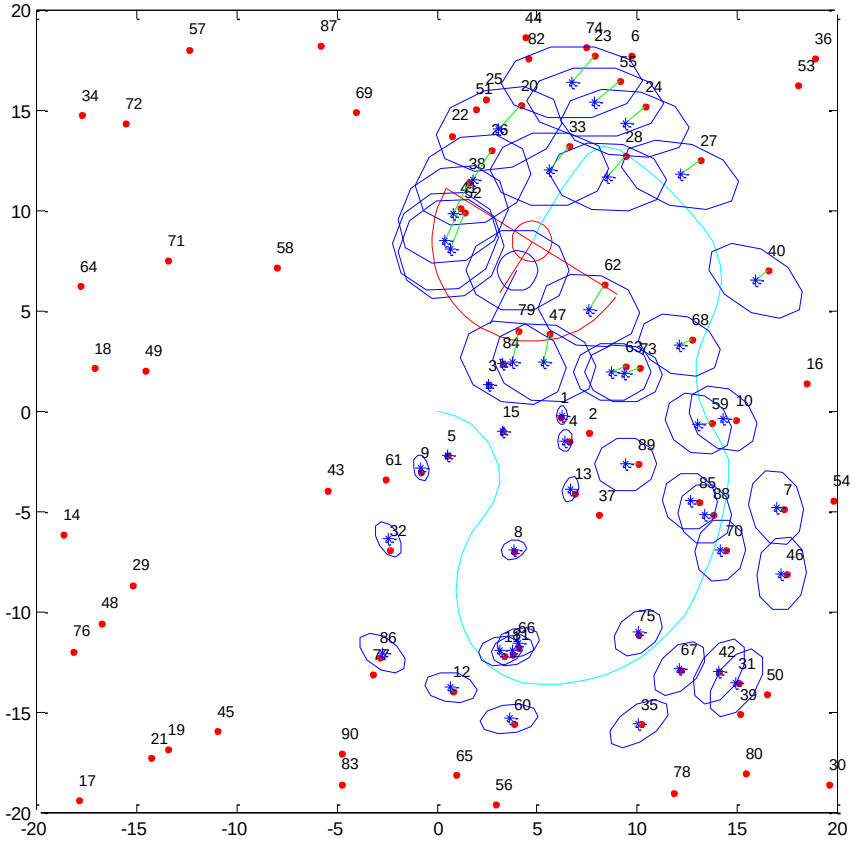


Figure 69 – Simulation#2 Figure#5

Just before observing the initially added landmarks.

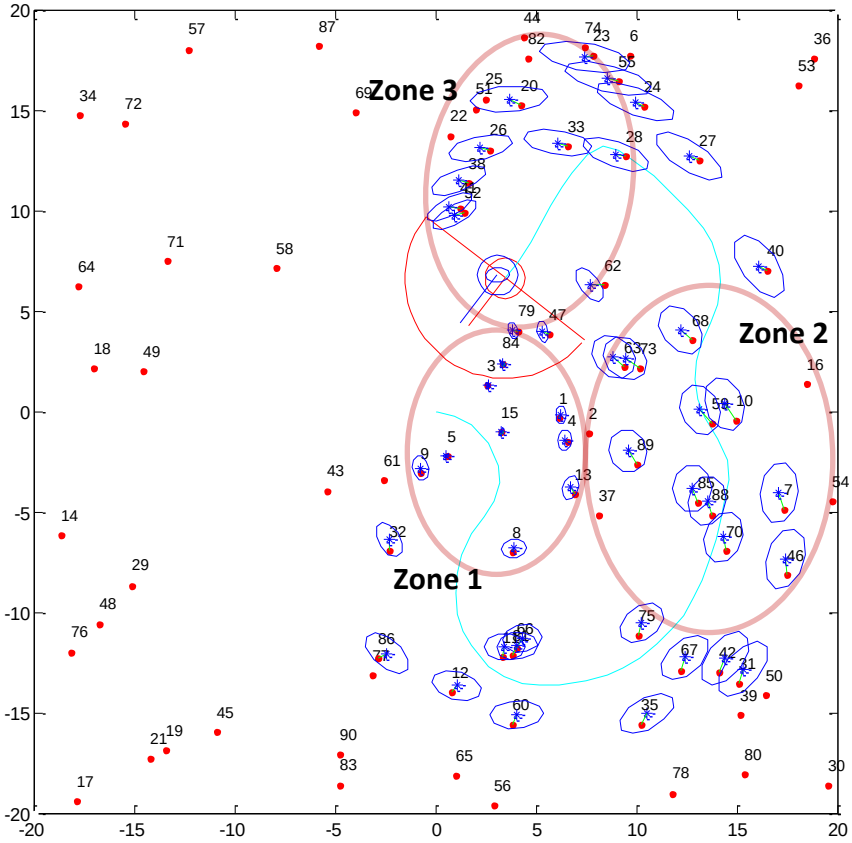


Figure 70 – Simulation#2 Figure#6

When the agent sees lm#84 again, it recognizes its true pose. Due to this reason the error rate of the other landmarks are decreasing too. Three zones are indicated in Figure 70. Correlations between the landmarks are strong in each zone, since the landmarks in the same zone are observed one after the other. Zone 1 shows initially observed landmarks.

Some variance values are given bellow for two landmarks in zone 3.

Table 5 – Variance rates for Figure 69 and Figure 70

#landmark		Figure 69 (Before seeing lm#84 again)	Figure 70 (After seeing lm#84 again)
lm#47	Variance of x	1,700091921	0,021434706
	Variance of y	1,002924156	0,07474686
lm#79	Variance of x	1,72171921	0,018939868
	Variance of y	1,135742843	0,041543165

In zone 3, lm#47 and lm#79 are the closest landmarks to the initial point. Due to this reason; when the initial landmarks are observed, their variance suddenly decreased to relatively low values.

Variance of lm#41, lm#62, and lm#52 are shown below (zone 3).

Table 6 – Variance rates for Figure 69 and Figure 70

#landmark		Figure 69 (Before seeing lm#84 again)	Figure 70 (After seeing lm#84 again)
lm#41	Variance of x	1,758776811	0,321233773
	Variance of y	1,60033739	0,133777038
lm#62	Variance of x	1,609371146	0,121511807
	Variance of y	0,87062264	0,181887278
lm#52	Variance of x	1,750693484	0,306990895
	Variance of y	1,609995055	0,135237631

lm#41, lm#62, and lm#52 are also in Zone 3 and correlation of them with lm#47 and lm#79 are strong. Due to this reason, even though they are not observed in the same scanner frame, certainty of them still improved. From a more general perception, correlation between Zone 1 and Zone 3 becomes stronger. Due to this reason the certainty of all landmarks in zone 3 is improved.

Below table shows the variance rates of two landmarks in zone 2.

Table 7 – Variance rates for Figure 69 and Figure 70

#landmark		Figure 69 (Before seeing lm#84 again)	Figure 70 (After seeing lm#84 again)
lm#46	Variance of x	0,236459526	0,144300758
	Variance of y	0,39427701	0,282492142
lm#7	Variance of x	0,206117676	0,091844527
	Variance of y	0,38707026	0,281333251

lm#46 and lm#7 (Zone 2) are farther from the starting point. They are also far from Zone 3. So, the improvement of their variance is weaker compared to lm#41, lm#62, and lm#52.

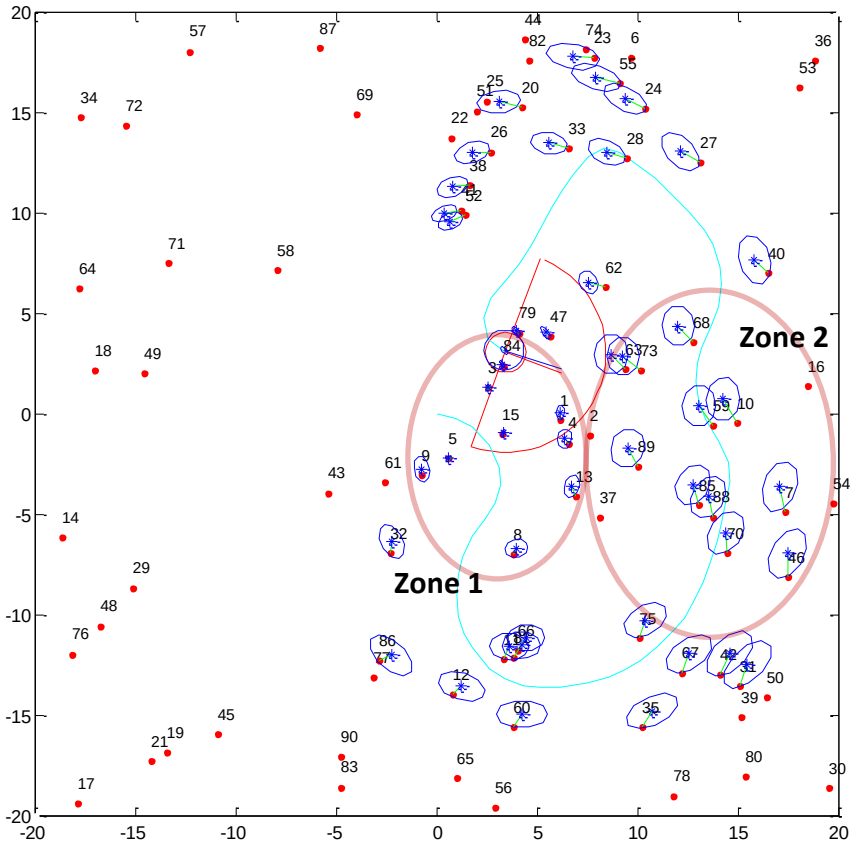


Figure 71 – Simulation#2 Figure#7

After a few observations certainty of robot's pose is improved (standard deviation circle of agent is shrinking in Figure 71 when compared with Figure 70). By this way, variance values of Zone 3 are improved too.

Table 8 – Variance of Figure 70 and Figure 71

#landmark		Figure 70 (After seeing lm#84 again)	Figure 71 (After taking 5 more observations)
lm#41	Variance of x	0,321233773	0,100970883
	Variance of y	0,133777038	0,048805747

Table 8 (continued)

lm#62	Variance of x	0,121511807	0,060371916
	Variance of y	0,181887278	0,082014497
lm#52	Variance of x	0,306990895	0,10001562
	Variance of y	0,135237631	0,048456047

The agent is turned to Zone 2. In a few seconds (Figure 72) it will observe some landmarks once more in Zone 2.

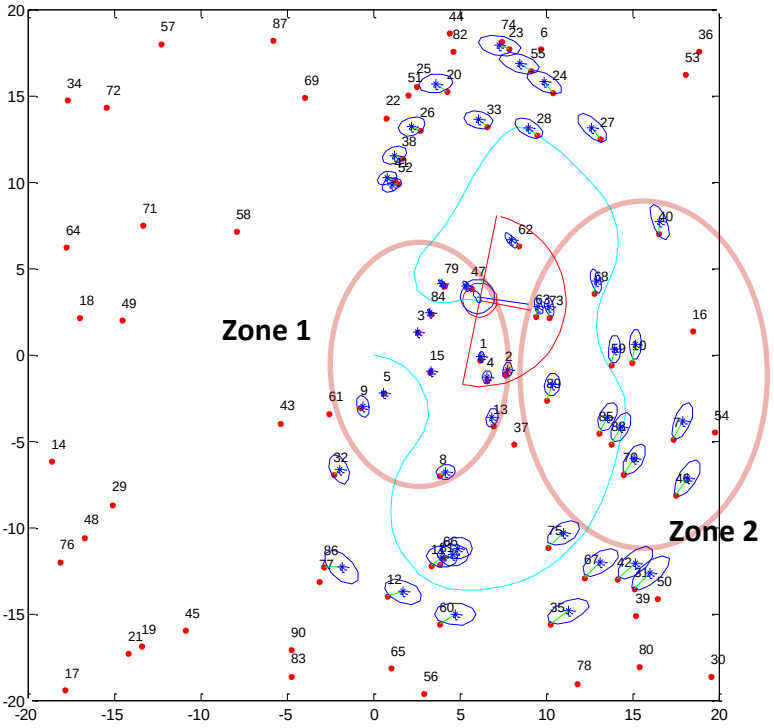


Figure 72 – Simulation#2 Figure#8

By the movement illustrated at Figure 72 the correlation between Zone 1 and Zone 2 is improved. Thus precision of belief pose of all landmarks in Zone 2 are enhanced. Variance values of lm#7 and lm#46 (in Zone 2) are shown below.

Table 9 – Variance of Figure 71 and Figure 72

#landmark		Figure 71 (before seeing lm#63, lm#73 again)	Figure 72 (after seeing lm#63, lm#73 again)
lm#46	Variance of x	0,236459526	0,144300758
	Variance of y	0,39427701	0,282492142
lm#7	Variance of x	0,206117676	0,091844527
	Variance of y	0,38707026	0,281333251

5.2 Erroneous Rotation of Map Frame Relative to Real Frame

In some of the figures above, the real positions of some features are not in the standard deviation ellipse (e.g. Figure 72). In other words, the difference between real pose and belief pose for some features are larger than two standard deviations. Since there is less number of features at the first steps, the impact of the locomotion error creates a gap between the real coordinate frame and the map coordinate frame. For example, if the robot has a *2°* error at the first few steps, the final map coordinate frame will possibly shift approximately with same amount. An exaggerated illustration of this shift can be seen at next simulation.

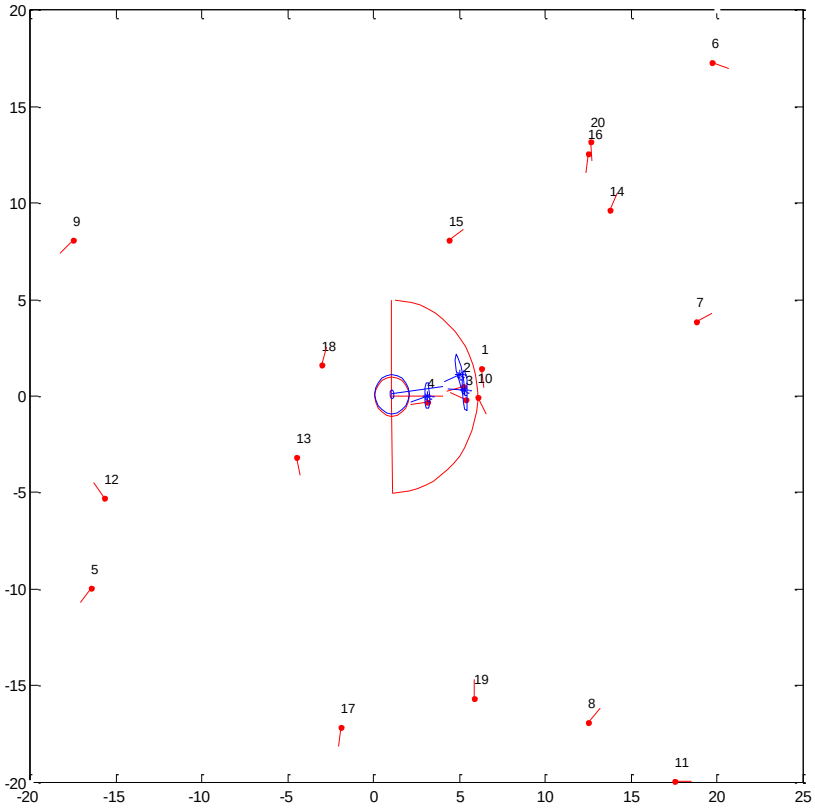


Figure 73 – Simulation#3 Figure#1

Figure 73 shows a state that the agent has a relatively large angular pose error at first step.

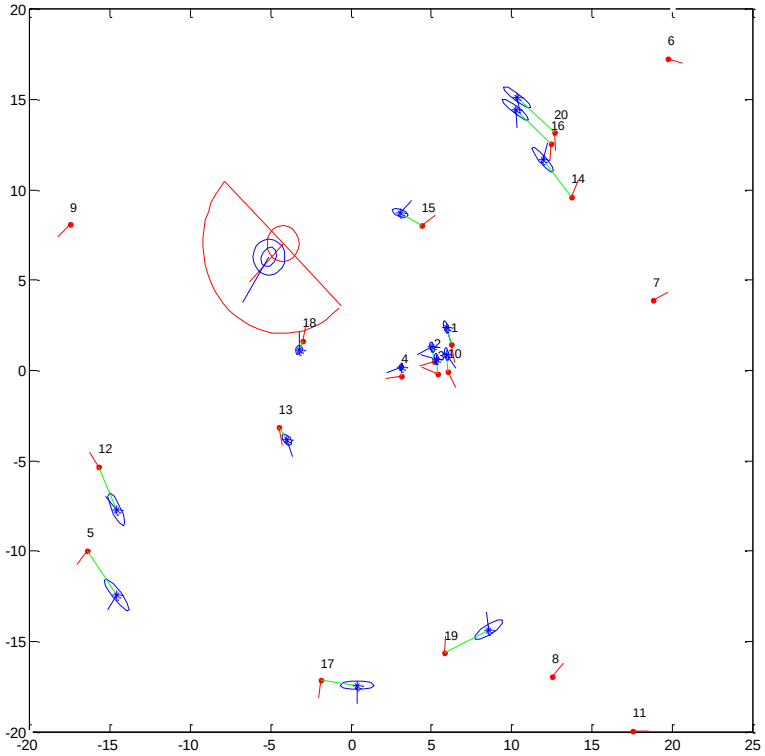


Figure 74 – Simulation#3 Figure#3

Agent explores the environment and after certain time the variance ellipses of the features gets smaller while the map frame still turned in CCW direction relative to the real world frame. In Figure 74 average error for all features is *1.74 meters*. However when the coordinate frame of the map is rotated *9.2°* in CCW direction (rotated through line: *x=0, y=0*), average error decreases to *0.12 meter*.

Similar impact can also be seen at Bailey's work.[41] For example, the map given below is the output of Bailey's MATLAB EKF SLAM algorithm.

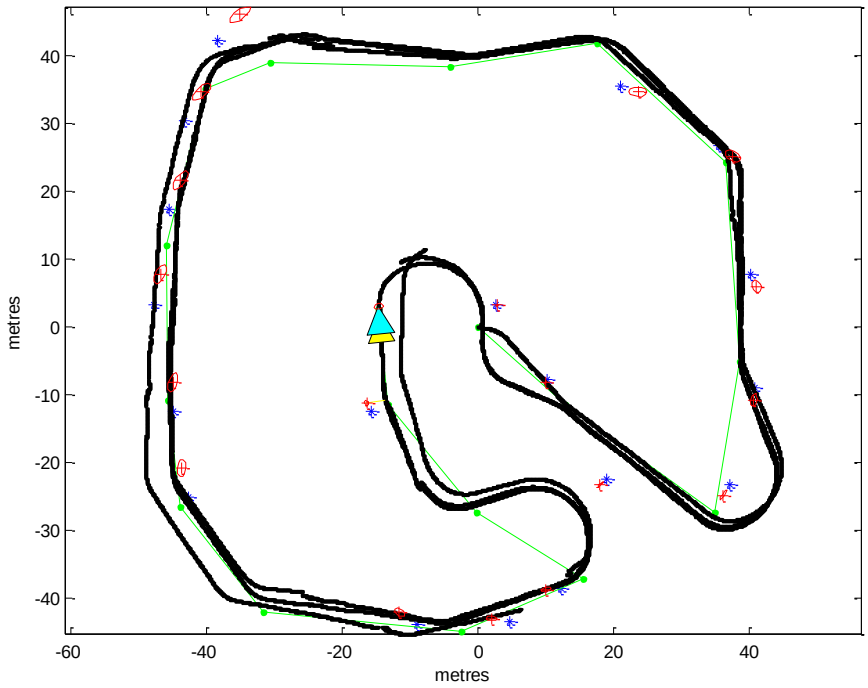


Figure 75 – Map created by simulation of Bailey.

In Figure 75 (Bailey's simulation) the true pose of the landmarks are indicated with blue stars (*) and the belief of them are indicated with red ellipses. Just as the code written in this work, landmarks are out of two standard deviation ellipses and the map is turned around starting point (turned CW direction in this example).

Due to the reasons explained above, average error for corrected map reference frame, will also be used while evaluating the performance of algorithm at following sections.

From Figure 76 to Figure 84 normal lines indicate average error rate of features in map, while dashed lines indicate average error rates of angularly corrected maps.

5.3 Effect of Uncertainty of Speed on EKF SLAM

In this part, the effect of uncertainty of speed on performance of EKF SLAM will be discussed. Keeping all other parameters fixed, the standard deviation of the speed of the agent was altered and 10000 monte-carlo simulation runs were executed for each standard deviation parameter set. Average output of error rate vs. time was illustrated bellow. When Figure 76 is examined, one can see that the error rate of the map increases with the standard deviation of translational speed. However increasing standard deviation of translational speed does not seem to create a large gap at error rate. Whereas, the gap between the resulting outputs may vary with other parameters of the agent.

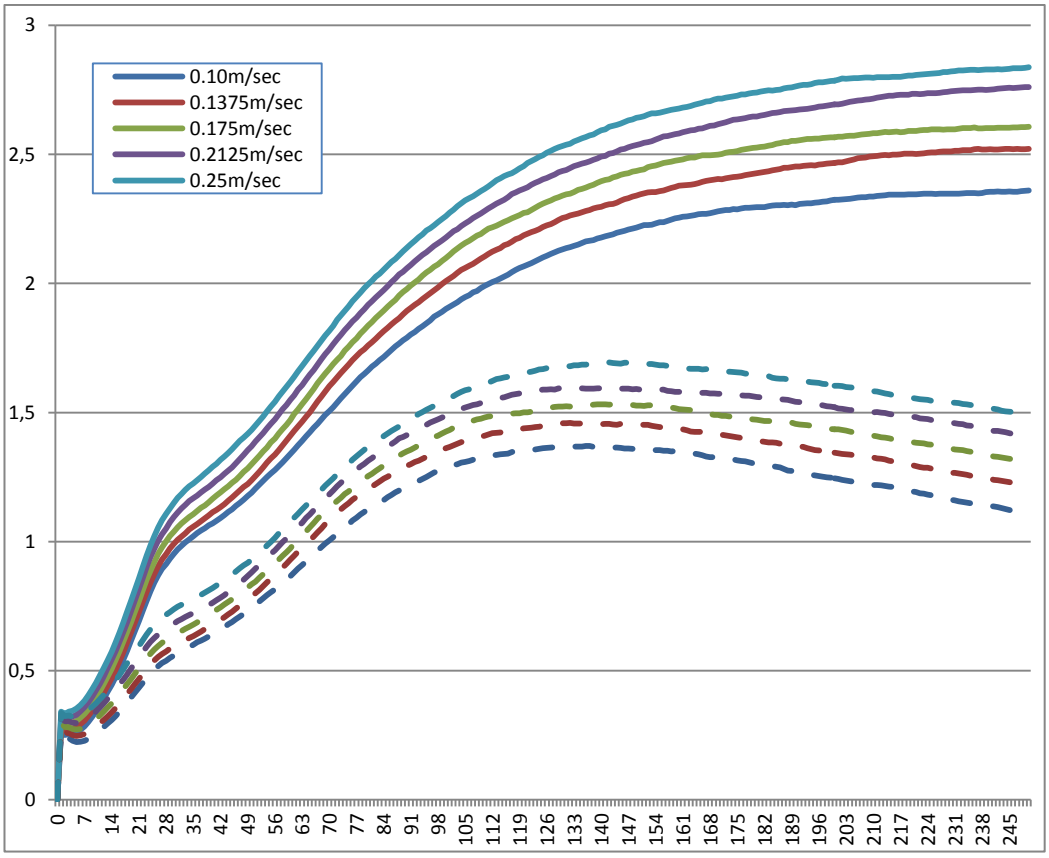


Figure 76 - Average Error Rate of 10000 Runs –x-axis: time (seconds) – y-axis: average error (meters)

The standard deviation of variance of speed is altered through 0.1m/sec to 0.25m/sec

Mean of linear speed: 1m/sec, standard deviation of angular speed: 3°/sec, standard deviation of observation yaw-pitch angles:3°, standard deviation of observation distance: 0.2m

5.4 Effect of Uncertainty of Angular Velocity on EKF SLAM

Similar effect holds for the uncertainty of angular speed of the agent. The graph of the average error rate for different values of the standard deviation of angular velocity can be seen below. Just as expected, increasing standard deviation of angular velocity leads increasing average error rate.

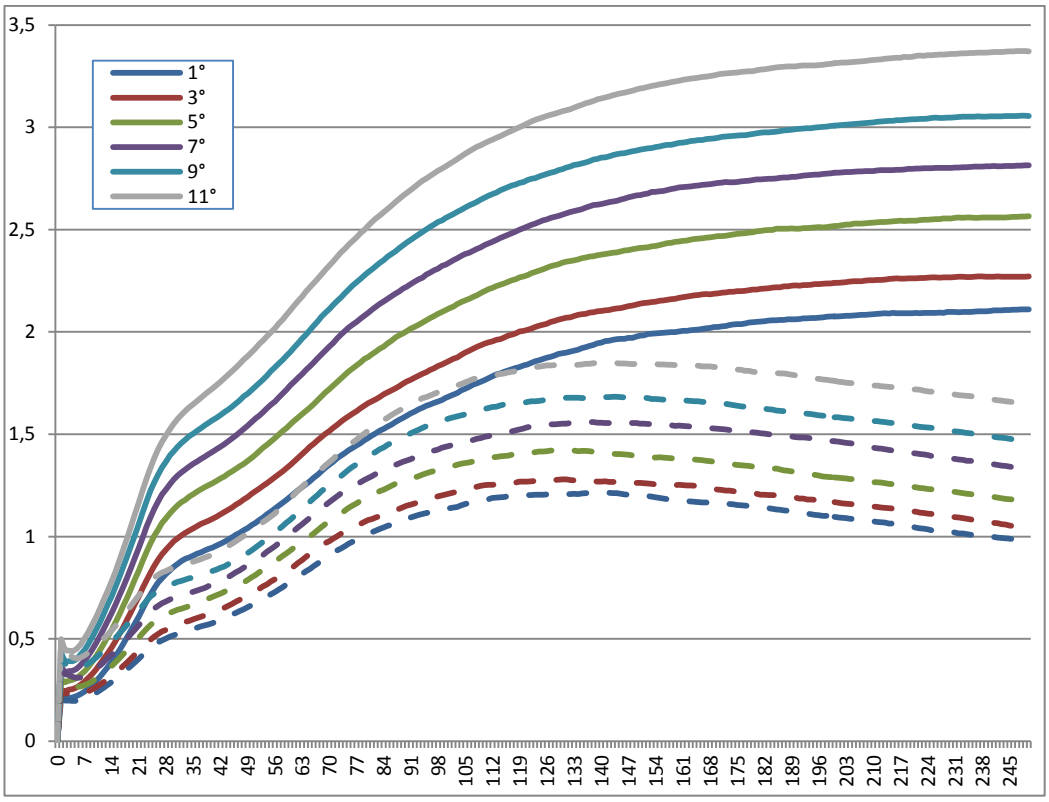


Figure 77 -Average Error Rate of 10000 Runs – x-axis: time (seconds) – y axis: average error (meters)

The standard deviation of variance of angular velocity speed is altered through 1°/sec to 11°/sec

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec, standard deviation of observation yaw angle: 3°, standard deviation of observation pitch angle: 2°, standard deviation of observation distance: 0.15m

5.5 Effect of Uncertainty of Observation Distance on EKF SLAM

A similar effect for change of the standard deviation of observation distance is expected like previous ones. Increasing variance of observation distance will possibly increase error rate but for given case (and possible for many cases) increased variance of observation distance does not create a large error rate since all graphs of different error rates are close to each other.

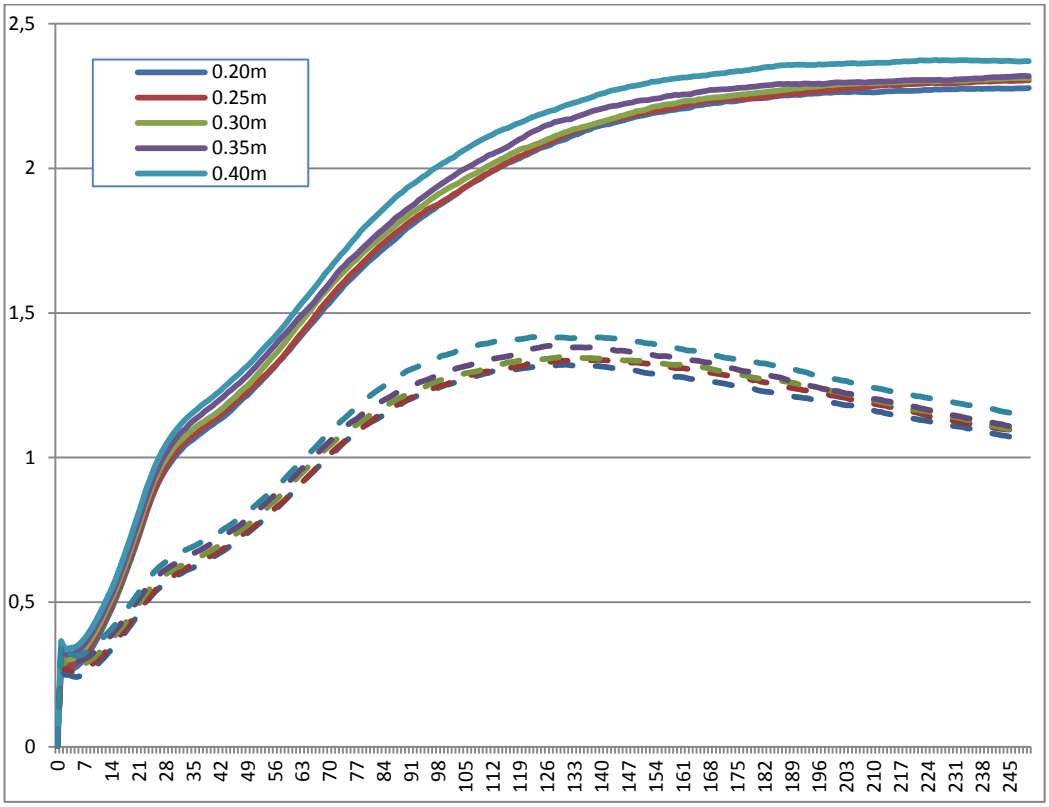


Figure 78 -Average Error Rate of 10000 Runs –x axis: time (seconds) – y axis: average error (meters)

The standard deviation of variance of observation distance is altered through 0.2m to 0.4m

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec, standard deviation of angular speed: 3°/sec, standard deviation of observation yaw angle: 3°, standard deviation of observation pitch angle: 2°, standard deviation of observation distance: 0.15m

5.6 Effect of Uncertainty of Observation Yaw Angle (Azimuth) on EKF SLAM

Standard deviation of the yaw (azimuth) angle of scanner is altered and 5000 monte-carlo runs were executed for each set. The average errors of monte-carlo runs are illustrated at Figure 79.

Increasing error rate with increasing standard deviation of observation is an expected result for this trial.

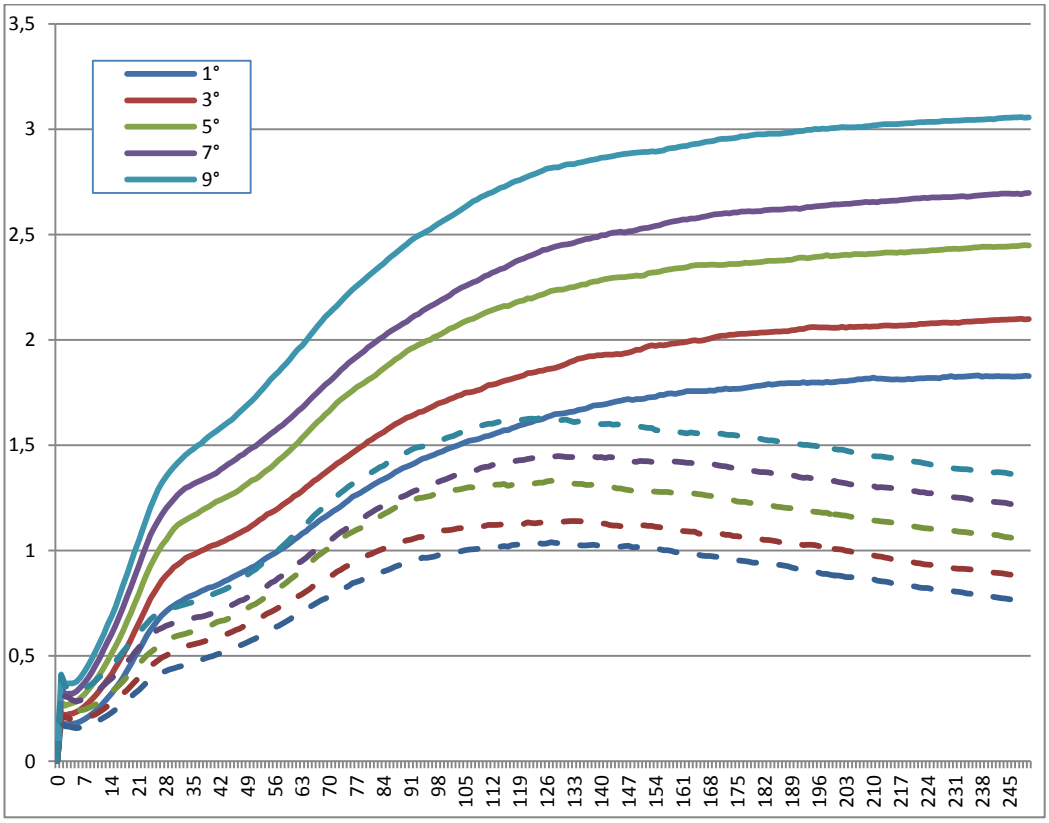


Figure 79-Average Error Rate of 5000 Runs –x axis: time (seconds) – y axis: average error (meters)

The standard deviation of variance of observation yaw angle is altered through 1° to 9°

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.1m/sec, standard deviation of angular speed: 3°/sec, standard deviation of observation pitch angle: 2°, standard deviation of observation distance: 0.15m

5.7 Effect of Uncertainty of Observation Pitch Angle (Elevation) on EKF SLAM

While using a laser scanner system in 3D EKF SLAM, a laser scanner with two degrees of freedom is needed. In this part, the standard deviation of the pitch angle of the agent will be altered and 20000 monte-carlo runs will be executed.

Figure 80 shows the result of the monte-carlo runs. Increasing error rate with increasing standard deviation is an expected result. But as it can be seen from the graph this standard deviation is not a dominant factor on the average error rates (for stated parameters).

As stated previously, the agent is assumed to be in a close door environment with flat ground. So the agent's pitch and roll angle is constant and covariance between z-parameter (height) and other parameters of landmarks is relatively weak. Due to this reason elevation angle of the landmark does not provide valuable information relative to other parameters.

However one must recognize that -since real world is a 3D environment- adding a z-parameter to EKF SLAM creates a more realistic model of the world and provides significant benefit to overcome correspondence problem.

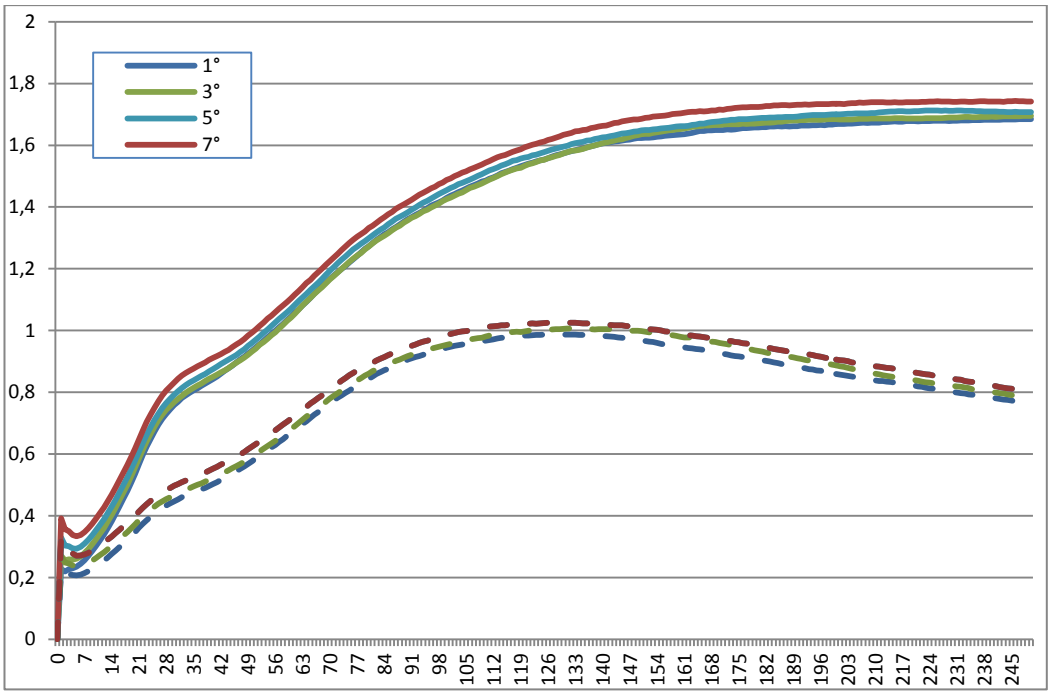


Figure 80-Average Error Rate of 20000 Runs –x time (seconds) vs. y average error (meters)

The standard deviation of variance of observation pitch angle is altered through 1° to 7°

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.1m/sec, standard deviation of angular speed: 2°/sec, standard deviation of observation yaw angle:2°, standard deviation of observation distance:0.2m

5.8 Effect of Uncertainty of Feature Direction on EKF SLAM

In EKF SLAM 3D landmarks are widely used. A 3D landmark consists of three parameters which show the location of the landmark in 3D map. In addition to these three parameters; there are also other variables, which indicate the variance of landmarks and correlation of each landmark with other landmarks.

In this work, a parameter that shows the direction of landmark is used, just like some versions of EKF SLAM in the literature. To represent a feature with more than three parameters, the sensor data must be processed with several kinds of data processing algorithms to extract the parameters of higher dimension features.

This part shows the effect of direction parameter on performance of EKF SLAM. Figure 81 shows the performance of algorithm while changing variance of direction of feature. Each line is average error rate of 12000 monte-carlo runs.

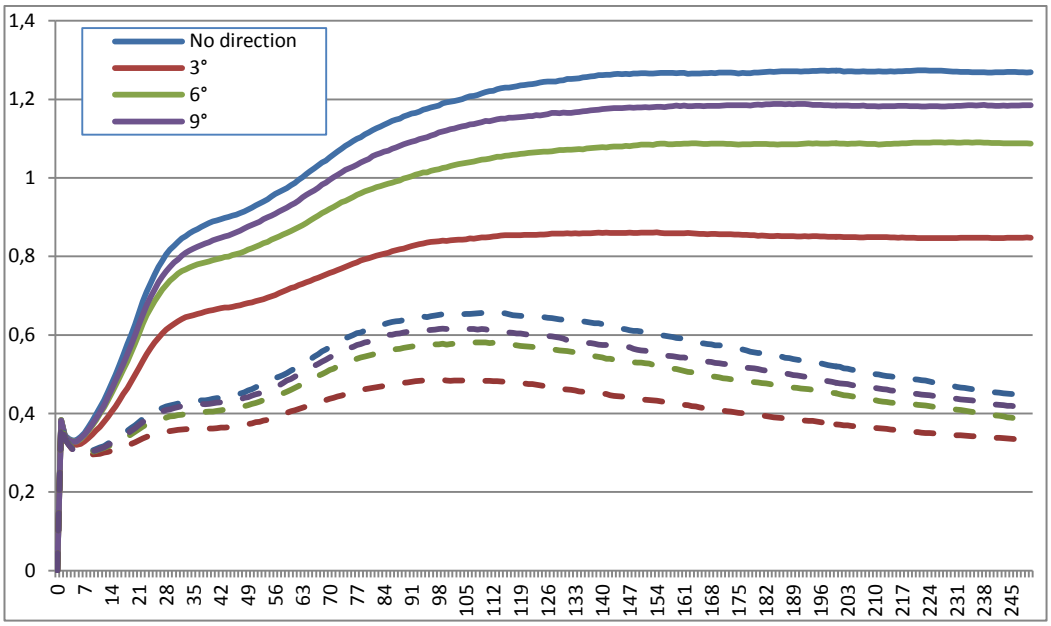


Figure 81 – Performance of EKF SLAM while changing standard deviation of direction of feature.

Average Error Rate of 12000 Runs –x axis: time (seconds) – y axis: average error (meters)

The standard deviation of variance of direction vector of features is altered through 3° to 9°. A line that represents no direction vector error rate is also added.

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec, standard deviation of angular speed: 2°/sec, standard deviation of observation yaw angle: 4°, standard deviation of observation pitch angle: 5°, standard deviation of observation distance: 0.15m

Decreasing EKF SLAM performance with increasing standard deviation is an expected result. But; unlike the previous results, increasing this factor to very high values does not make the algorithm unstable.

As stated previously; adding a direction vector to the features provides additional feedback to direction of the robot. However the algorithm can also be executed without this parameter. So direction vector makes benefit for relatively high levels

of precision. While increasing the uncertainty of this newly added parameter, performance of the algorithm gets closer to the one with no direction vector.

The direction information of the features provides very direct feedback to the direction of the agent. All other Jacobian's of feature direction are "*0*" for the state matrix of the agent and landmark poses. The Jacobian equation of the direction vector (in "*H*") is as follows:

$$H_t^{lm\ i/r} = \sigma\hat{\alpha}_t^i / \sigma\alpha_t^r = -1 \quad \textbf{5-1}$$

Even though, every single parameter in EKF SLAM provides feedback to each other, the direction of the agent is dominated by two parameters stated bellow.

For the robot locomotion system, change in the robot direction at time t, can be computed as follows:

$$\theta_{robot,t} - \theta_{robot,t-1} = \Delta\theta_{robot,t} = \omega_t\,\Delta t \quad \textbf{5-2}$$

To find the Jacobian of change of position, derivative of (5-2) must be found:

$$\sigma(\Delta\theta_{robot,t}) = \sigma\omega_t\,\Delta t \quad \textbf{5-3}$$

Δt represents the change in time and it is the multiplier of all control input. For simplification $\theta_{robot,t-1} = 0$ and $\Delta t = 1$. By this way:

$$\sigma\theta_{robot,t} = \sigma\omega_t \quad \textbf{5-4}$$

So the variance of the angular speed "$\sigma\omega_t$" is one of the main dominant factor of the uncertainty of angular pose of the agent. Actually, in robot control system since:

$$\sigma\theta_{robot,t} / \sigma v_t = 0 \quad \textbf{5-5}$$

$\sigma\omega_t$ is the only factor that affects the uncertainty of the angular pose of the agent in direct manner.

Now the effect of the sensor system on the angular pose of the agent will be examined. Direct contribution of the sensor input to pose of the agent is determined by *H* function, as stated in (3-55):

$$H = \frac{\sigma\ observation\ input\ of\ sensor}{\sigma\ belief\ pose\ of\ robot\ and\ landmarks} \quad \textbf{5-6}$$

So for angular pose of the agent:

$$\sigma\hat{r}_t^i/\sigma\alpha_t^r = 0 \quad \textbf{5-7}$$

$$\sigma\hat{p}_t^i/\sigma\alpha_t^r = 0 \quad \textbf{5-8}$$

and

$$\sigma\hat{\theta}_t^i/\sigma\alpha_t^r = -1 \quad \textbf{5-9}$$

In short, there are two main direct inputs of EKF SLAM algorithm: the locomotion input and the sensor readings of the agent. All other variables and parameters are composed of these two main input systems (and initial pose of the agent).

Locomotion is composed of v_t and ω_t.

$$\sigma\theta_{robot,t}/\sigma v_t = 0 \quad \textbf{5-10}$$

and

$$\sigma\theta_{robot,t} = \sigma\omega_t\,\Delta t \quad \textbf{5-11}$$

Sensor readings are composed of $\hat{r}_t^i$, $\hat{\theta}_t^i$, $\hat{\rho}_t^i$, $\hat{\alpha}_t^i$.

$$\sigma\hat{r}_t^i/\sigma\alpha_t^r = 0 \quad \textbf{5-12}$$

$$\sigma\hat{p}_t^i/\sigma\alpha_t^r = 0 \quad \textbf{5-13}$$

$$\sigma\hat{\theta}_t^i/\sigma\alpha_t^r = -1 \quad \textbf{5-14}$$

$$\sigma\hat{\alpha}_t^i/\sigma\alpha_t^r = -1 \quad \textbf{5-15}$$

In other words, even though every input of the sensor and the locomotion control input provides feedback to all other parameters; ω_t, $\hat{\theta}_t^i$, $\hat{\alpha}_t^i$ provide *very direct feedback* to the angular pose of the agent. So, weakness of precision of ω_t and $\hat{\theta}_t^i$ can be corrected by $\hat{\alpha}_t^i$ for some cases.

At this point, the writer of this work does not claim that the only or the strongest method to fix weakness of angular pose is adding a direction vector to the features. However, since it provides very direct feedback to the angular pose, using a direction vector for features shall be a beneficial method to correct the angular pose of the agent.

Performance of EKF SLAM algorithm is illustrated bellow for different variance values assigned to the direction feature.

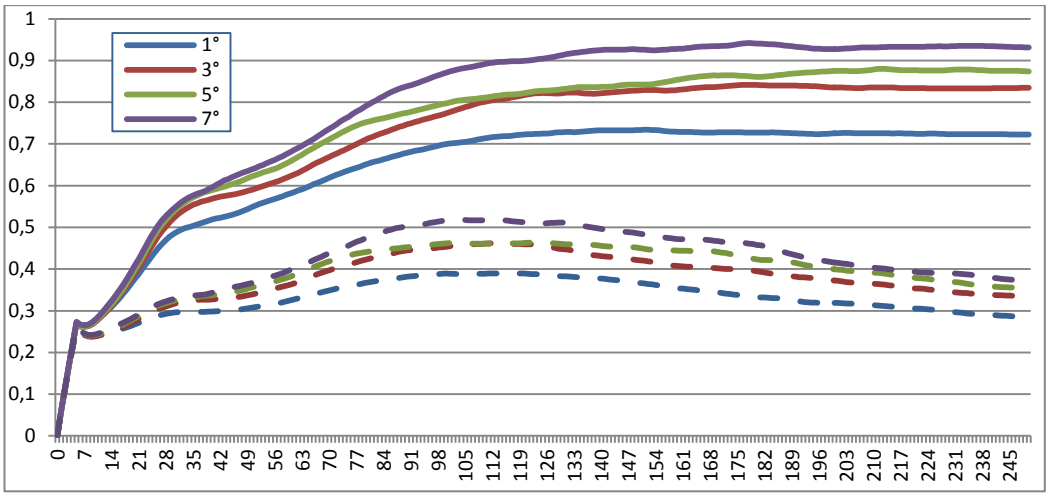


Figure 82 - Performance of EKF SLAM while changing standard deviation of direction of feature

Average Error Rate of 1500 Runs –x axis: time (seconds) – y axis: average error (meters)

The standard deviation of variance of direction vector of features is altered through 1° to 7°, standard deviation of angular speed: 1°/sec, standard deviation of observation yaw angle:1°

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec,, standard deviation of observation pitch angle:5°, standard deviation of observation distance: 0.15m

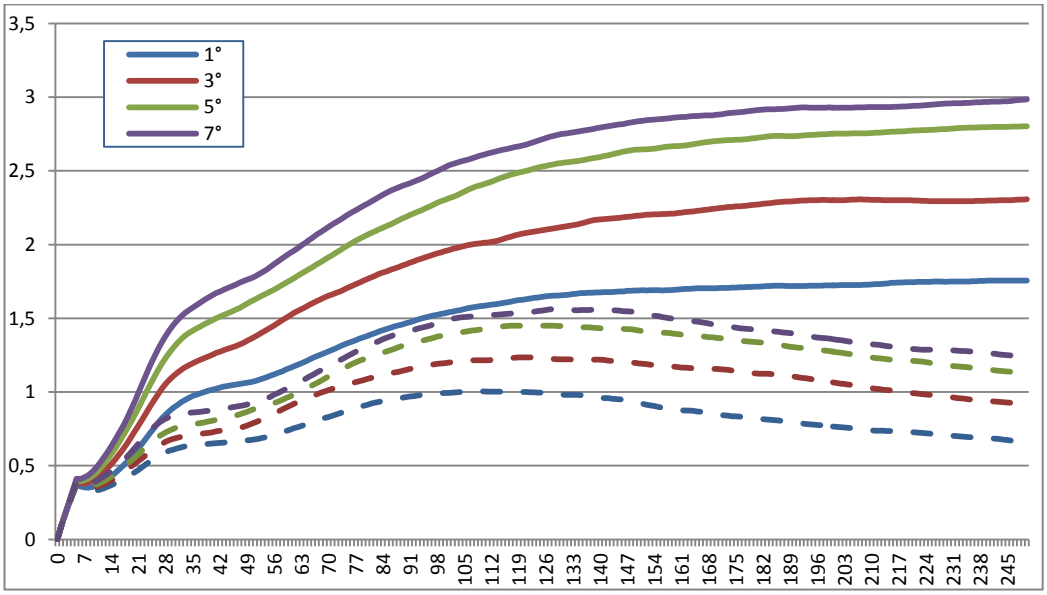


Figure 83 - Performance of EKF SLAM while changing standard deviation of direction of feature

Average Error Rate of 1500 Runs –x axis: time (seconds) – y axis: average error (meters)

standard deviation of variance of direction vector of features is altered through 1° to 7°, standard deviation of angular speed: 7°/sec, standard deviation of observation yaw angle: 7°

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec,, standard deviation of observation pitch angle: 5°, standard deviation of observation distance: 0.1m

Once, Figure 82 - Performance of EKF SLAM while changing standard deviation of direction of featureand Figure 83 - Performance of EKF SLAM while changing standard deviation of direction of feature are examined, one can realize that the contribution of the direction vector precision is stronger in Figure 83 - Performance of EKF SLAM while changing standard deviation of direction of feature. The reason of this issue is that the precision of the angular velocity and the landmark observation angle is stronger at the first analysis. Since, uncertainty of the angular pose of the robot is smaller in Figure 82 - Performance of EKF SLAM while changing standard deviation of direction of feature, contribution of feedback loses its significance.

When the standard deviation of angular velocity (ω_t) and the observation angle ($\hat{\theta}_t^i$) are 1°, increasing standard deviation of direction (decreasing precision) from 1° to 7° increases the average error rate ≈1.3 times.

On the other hand, when standard deviation of angular velocity (ω_t) and observation angle ($\hat{\theta}_t^i$) are 7°, increasing standard deviation of direction (decreasing precision) from 1° to 7° increases the average error rate ≈1.7 times.

So, direction of landmark provides stronger feedback at higher uncertainty of angular velocity (ω_t) and observation angle ($\hat{\theta}_t^i$).

In short, using planar features may be a beneficial method to improve the performance of EKF SLAM for many cases. Once such a vector is added, number of parameters that represent each feature will increase and due to this reason the new parameters shall be used to distinguish landmarks in the map. Thus increased number of parameters is expected to provide valuable information to avoid mis-correspondence problem. As stated in Section 2.4.1 one of the weaknesses of feature based EKF SLAM is the correspondence problem and increasing the number of parameters is a beneficial way to provide a tool for distinguishing the features from each other. Additionally representing the environment with higher level of features leads a deeper level of understanding of the maps, which is a significant property of feature based SLAM as it can be seen on Figure 81. In addition to

providing a stronger representation of the environment, increasing the number of parameters for each feature supplies stronger feedback to the pose of the agent, thus increase the performance of EKF SLAM.

However one must remember that adding a direction vector requires an appropriate sensor system and several feature extraction and comparison mechanisms. In addition to this, adjusting precision of direction vector may be a challenging issue and certainty of direction vector may be restricted by performance of sensor system. So, with improved precision of sensor system, an improvement at other parameters of features is expected too. Due to this reason, utilizing a stronger sensor system and stronger feature extraction algorithms is expected to increase precision of direction vector. But such operation may not provide expected amount of benefit on performance for each case, since certainty of other parameters are also expected to increase with stronger sensor system and feature extraction algorithms.

5.9 Using Incorrect Standard Deviation Values

In simulation part of the thesis, the algorithms run from the path that is illuminated by theoretic knowledge, but in real experiments there may be unpredicted interference from environment or undesired sensor failures may occur.

As stated previously, while presenting a sensor noise model for IRSCAN, the variance rates used for the algorithms are chosen higher from the measured variance rates. At this section, the effect of choosing wrong variance rates will be examined. The following table and graph shows a set of cases that, the standard deviations of the sensors were chosen wrong.

Table 10 – Error rates of Figure 84 at the end of 500 seconds.

Manipulation ratio	True Values of Standard Deviation of Scanner Yaw – Pitch Angle – Landmark Direction Angle (comes from real sensor model)	EKF SLAM Algorithm Values of Standard Deviation of Scanner Yaw – Pitch Angle – Landmark Direction Angle (manipulated sensor model)	Average Error Rate at the end of 500 Seconds
0.25	3° - 3° - 3°	0.75° - 0.75° - 0.75°	4,441884m
0.5		1.5° - 1.5° - 1.5°	3,121644m
0.75		2.25° - 2.25° - 2.25°	2,688035m
1		3° - 3° - 3° (no manipulation)	2,612749m
1.25		3.75° - 3.75° - 3.75°	2,605406m
1.5		4.5° - 4.5° - 4.5°	2,619884m
1.75		5.25° - 5.25° - 5.25°	2,694553m
2		6° - 6° - 6°	2,708768m

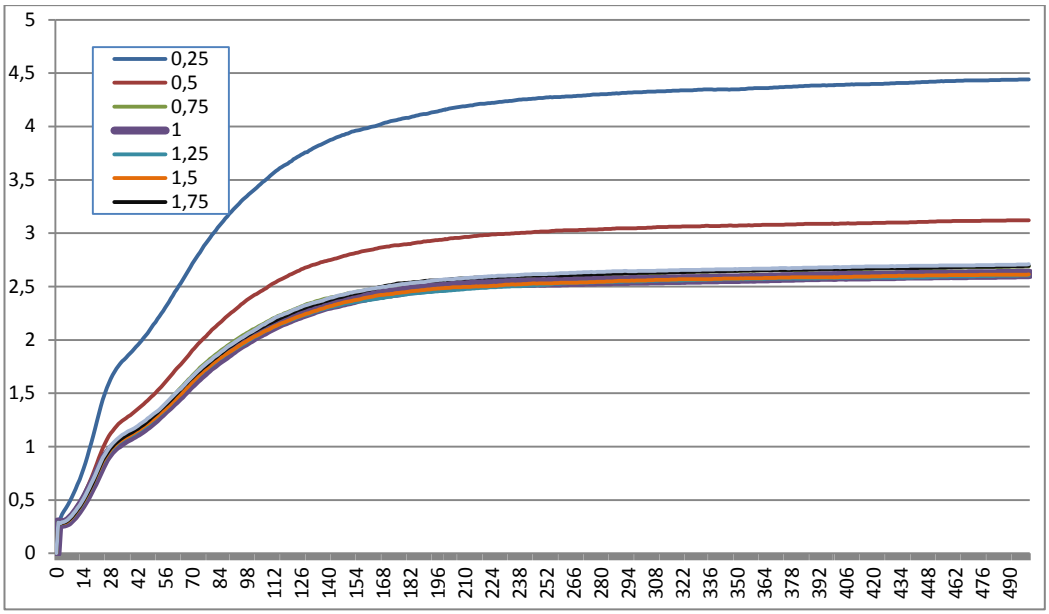


Figure 84 – Shows performance of EKF SLAM when standard deviation values are chosen wrong. Bold purple line (1) indicates true standard deviation. Blue line (0.25) indicates performance of EKF SLAM where belief of standard deviation used by EKF SLAM is four times smaller than true noise model. Gray line (2) indicates performance of EKF SLAM where belief of standard deviation used by EKF SLAM is two times greater than true noise model. True and belief standard deviation rates can be seen in Table 10.

Average Error Rate of 18500 Runs –x time (seconds) vs. y average error (meters)

Mean of linear speed: 1m/sec, standard deviation of linear speed: 0.15m/sec, standard deviation of angular speed: 3°/sec, standard deviation of observation distance: 0.2m

As Figure 84 and Table 10 indicate, using a higher deviation rate is not as risky as using a lower one. Actually the performance of EKF SLAM does not degrade when the belief of standard deviation is slightly increased. In contrast, decreased deviation rate may deteriorate performance of EKF SLAM dramatically. The algorithm may even collapse at some cases with decreased variance rate. Due to this reason; while constructing a sensor model in the true experiments, it is sensible to set the variance of the sensors a little bit higher values from the measured variance rates. By this way, a stronger safety margin shall be created to prevent the algorithm against the impact of undesired interference from the environment or the sensor failures. That is why standard deviations of sensor system were set to values higher than measured values in Section 4.4.2.

CHAPTER 6

EXPERIMENTS WITH REAL SENSOR

In this part the experimental EKF SLAM runs will be demonstrated. The sensor system constructed for this work was used to collect data from real environment. The experiments are conducted at indoor environment and artificial objects (rectangular shapes covered with white paper) were used to improve consistency of IRSCAN.

Feature extraction is done by edge based segmentation methods, using *edge* function with several parameters of MATLAB 7.12.0 (explained at Section 4.4 and

APPENDIX A). Features were never extracted via complete supervisor support. But in some cases non-marked pixels of edges was marked manually to obtain closed objects (manually marked edges will be emphasized at explanations of figures and will be shown with brighter pixels on edge maps in Appendix F). Using edge based segmentation for the feature extraction algorithm may suffer from finding the edge lines with a few pixels fault. Such faulty impacts were not corrected by the supervisor. For example the plane may be extracted a few pixels larger or smaller for some cases; due to this reason the transition between the object and the background may be included into the feature itself. By this way, the range and the angular pose of the feature will be found slightly incorrect. Since real world EKF SLAM may also suffer from such erroneous effects, none of the challenging problems stated above was fixed by the supervisor. By this way, it is believed that performance of the algorithm shall be tested with more realistic parameters. On the

other hand, to avoid divergence of EKF SLAM, variance of input data was increased as stated in Section 4.4.

Finding true correspondence between the observations is out of scope of this work and the correspondence is conducted with complete supervisor support (the features are numbered via supervisor).

To extract the direction of the planes, a plane fitting algorithm was used.[38] Since the agent has no locomotion system, it was moved manually and the variance of locomotion system is fixed (*standard deviation of velocity (σv) = 0.01m/sec, standard deviation of angular velocity (σw) = 2°*).

At the experiments:

Edge detection method defined in Section 4.4.2 and

APPENDIX A was used to find edges in distance maps

Segmentation of object was done as explained in Section 4.4.2;

Corner points were found as explained in Section 4.4.3;

Position and direction of the CoM were found as explained in Section 4.4.2;

Variance of the corner points and variance of the position of CoM were calculated as explained in Section 4.4.1 and Section 4.4.2;

Finally, variance of the plane direction was calculated as explained at 4.4.2

Once the control inputs, the variance of the control inputs, the pose of corners and CoM, the variance of pose of corners and CoM are obtained; EKF SLAM algorithm can be executed.

Through the experiments the planar segments are placed in front of IRSCAN and after collecting the data set IRSCAN is moved one step forward to plane segment. An image of experiment with two planar segments is shown below:

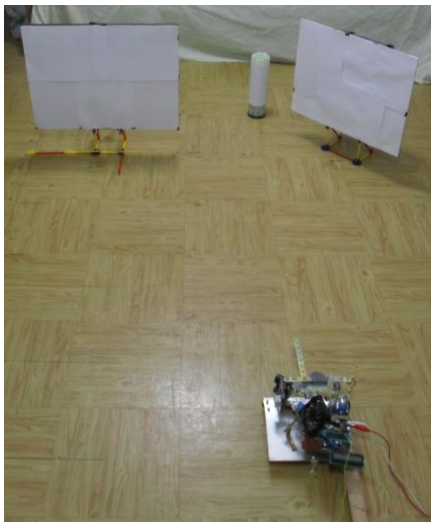


Figure 85 – Two objects are placed in front of IRSCAN. After collecting the data, the sensor will be moved to the next point and another set of data will be collected. Such moving and data collecting operation will be repeated for each set of data of each experiment.

Graphics of error rates of four experiments are presented below:

Figure 86 – Error rate of Experiment#1 – x axis: step number – y axis: average error rate – IRSCAN collected 4 set of data from different locations – Details of experiment can be seen on Appendix F

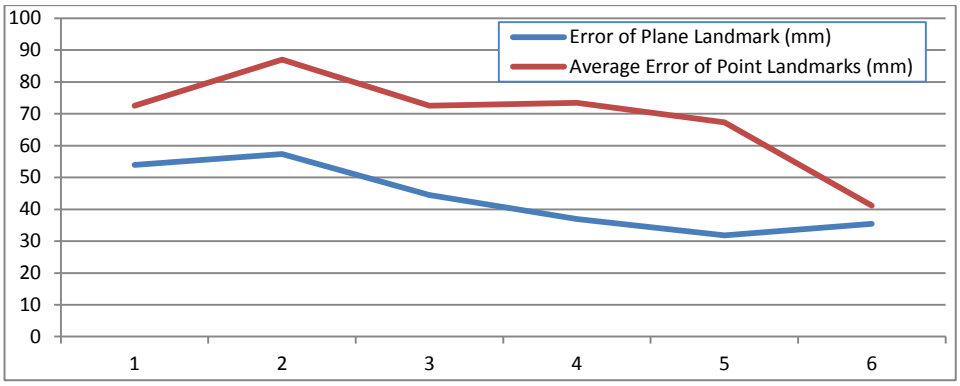


Figure 87 – Error rate of Experiment#2 – x axis: step number – y axis: average error rate – IRSCAN collected 6 set of data from different locations – Details of experiment can be seen on Appendix F

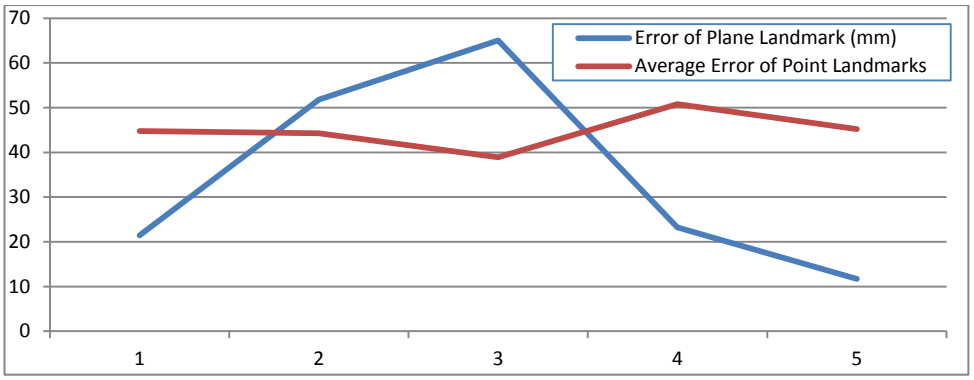


Figure 88 – Error rate of Experiment#3 – x axis: step number – y axis: average error rate – IRSCAN collected 5 set of data from different locations – Details of experiment can be seen on Appendix F

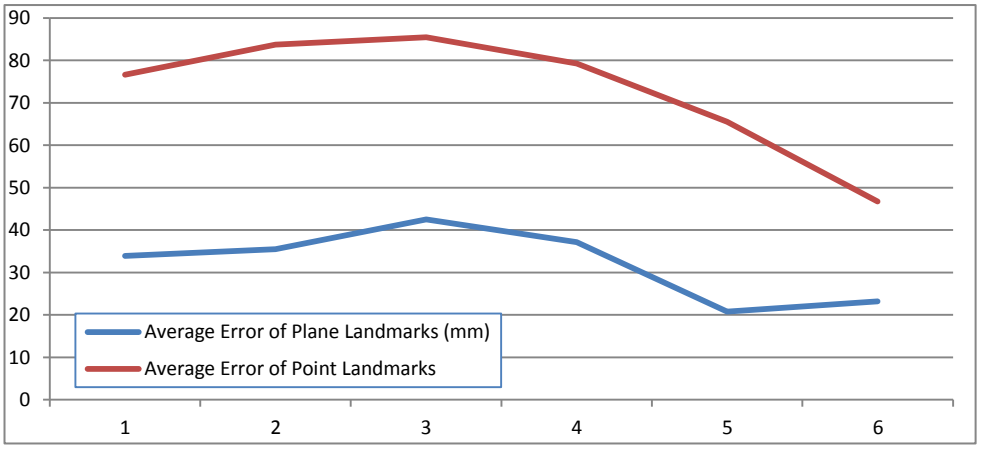


Figure 89 – Error rate of Experiment#4 – x axis: step number – y axis: average error rate – IRSCAN collected 6 set of data from different locations – Details of experiment can be seen on Appendix F

As it can be seen on the above graphs, error rate of the point landmarks is fluctuating between 25mm to 90mm while error rate of the plane landmarks is fluctuating between 10mm to 60mm. Such result supports simulation outputs. For the give case, implementing feature based EKF SLAM with planar landmarks is a stronger method then using four corner point features for IRSCAN and provided sensor model.

However due to the probabilistic nature of EKF SLAM, the performance of the point features may outrun the performance of the plane segment features at several cases. For example; the read can realize that, the point features provides less amount of error rate in experiment#3 - step#2-3. Such fluctuations are expected in EKF SLAM, because of its probabilistic nature. The writer of this work emphasizes that, the smooth graphs illustrated in Section 0 are the average error rate of thousands of monte-carlo trials. Whereas; when the number of the monte-carlo trials is decreased to five or ten executions, such undesired fluctuations also become visible in monte-carlo graphs.

EKF SLAM is expected to converge for given parameters through experiment time. Such effect can also be seen on the graphs. Average error rate of the plane landmarks decreases from 34.8mm to 23.5mm at four experiments. Similarly average error rate of the point landmarks decreases from 55.8mm to 38.5mm, through the experiments. These results supports converging behavior of the feature based EKF SLAM for the presented algorithms and presented parameters. At this point the writer of the work emphasize that, this decreasing error rate indicates the main trend of the algorithm for given parameters; but in some cases error rate may also increase due to probabilistic structure of EKF SLAM algorithm. An example of increasing error rate can be observed in experiment#3 step#2-3. Such increasing average error rates can also be visualized in simulation environment.

A significant contribution of using 3D landmarks with orientation information is about the correspondence issue. However since correspondence in experiments are

conducted via supervisor support, it is not possible to observe such contribution via these experiments.

In experiment#1-2-3 the size of the covariance matrix is 7x7 for the planar features, while the size of covariance matrix is 15x15 for the point features. Similarly in experiment#4 the size of covariance matrix is 11x11 for the planar features while the size is 27x27 for the point features. Thus, as stated in Section 2.4.1 processing power required for EKF SLAM with planar features is less than processing power needed for EKF SLAM with point features. However, feature extraction algorithms also use processing power to extract features and the amount of processing power needed for feature extraction algorithm may dominate the time required for execution cycle.

In short an error rate with increased amount of variance values is presented at Section 4.4. The presented feature extraction algorithms with indicated variance rates are combined to execute feature based EKF SLAM. The output results of experiments are consistent with converging behavior of EKF SLAM for given parameters and this supports the idea of utilizing 3D planar features instead of 3D points features to improve the performance of feature based EKF SLAM algorithms.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this chapter of the thesis it will be summarized what was done throughout the work, conclusions gained from the experiments will be reviewed and possible future extensions of this work will be presented. Starting from the review and the contributions;

- An accurate algorithm of SLAM is implemented by utilizing Extended Kalman Filter. The algorithm use CoM information for features and a direction vector is added to CoM information. Adding a direction vector to CoM provides valuable information for SLAM but it requires full observability for the extracted feature while popular SP-Model plane representation does not[20].
    - The previous methods used for SLAM are shortly explained and a friendly mathematical background of Extended Kalman Filter is presented.
    - Center of mass and the orientation of the features are used as landmarks for the algorithm.
    - Open form of the control update equations, Jacobians of the control update equations and Jacobians of the observation equations for EKF are presented.
    - All mathematical equations used for EKF SLAM are presented with the same order that they are used in the algorithm. Every element of

Jacobians, update equations, error matrixes and Kalman Gain equations are presented in open form with true execution order.

  - A second EKF SLAM algorithm that uses 3D point features instead of plane segments is also presented (the second algorithm is created by omitting some parts of first algorithm's equations).

- An open source simulation system with a useful interface, that makes it available to test performance of –both plane segment based and point based- EKF SLAM algorithm is coded and presented. Basic usage of the simulation needs no coding operation (or code manipulation) but works via buttons on MATLAB interface.
  - The simulation system creates a virtual environment composed of several landmarks and a scanner integrated agent which can sense the virtual landmarks. The exact control and measurement data is hidden form EKF SLAM algorithm, but Gaussian noise added versions of these data are inserted as inputs.
  - True and belief pose of the agent true pose of the landmarks, the standard deviation ellipses of belief pose of the agent and the landmarks can be visualized with simulation interface. The robot in the simulation environment can actively be controlled via MATLAB interface. By this way, one can experience behavior of EKF SLAM by controlling the simulation agent.
  - Every single standard deviation parameter of EKF SLAM and several skills of agent can be adjusted via MATLAB simulation interface. By this way, the behavior of EKF SLAM algorithm can be tested for different parameters.
  - A monte-carlo analysis option is added to the simulation code. By this way the effect of change of the noise parameters, several properties of the agent and some environmental conditions can be altered and average error rates for each set of parameters can be

graphed through time. By this way main trends of effect of parameters can be observed.

- A custom sensor IR distance measurement based scanner system is designed and constructed, named IRSCAN.
    - All mechanical structure, electronics circuit, control and interface software programming of IRSCAN are designed and implemented by the writer of this work.
    - A method for calibration of IRSCAN is presented and implemented.
    - A noise model for range measurement, yaw and pitch angles is proposed for IRSCAN.
    - Open source MATLAB codes of well-known edge detection methods are implemented to extract features from data collected by IRSCAN. By this way, a practical method for edge based segmentation is presented for the sensor system.
    - A practical open source plane extraction algorithm is modified and implemented to segmented point clouds of IRSCAN. By this way the planes used for feature based EKF SLAM can automatically be generated. A practical noise model is presented for the generated planes.
    - A practical corner detection method was presented for the distance image created by IRSCAN. By this way, the points used for point-feature EKF SLAM can automatically be generated. A practical noise model is presented for the generated points.
- The simulation algorithm written for EKF SLAM code is executed and the results of several monte-carlo runs are illustrated.
    - For each monte-carlo run, one of the parameter is swept through a range and the resulting average error rates are graphed. By this way, several clear illustrations of effect of parameters on EKF SLAM are presented.

- The effect of precision of the direction vector is emphasized through simulation runs. The feedback effect of the direction vector to uncertainty of the agent's angular velocity and the sensor's angular observation is discussed.
- The effect of using numerically wrong variance rates, is discussed and graphed. Through this discussion, a method to enhance the performance of IRSCAN, against possible failures is presented (and used in sensor noise model).

- EKF SLAM algorithms are tested with real data which is collected IRSCAN.
  - Since in all theoretical knowledge, sensor models, EKF SLAM equations and codes are practiced in the experiments, it is believed that, the output results of the experiments provides data to compare the accuracy of proposed methods with different extensions of SLAM methods.
  - Both plane based features and point based features are used for real experiments. Output results of both methods are illustrated and compared in same the tables and graphs. By this way, a set of sensor tools and sensor data is presented to compare the average error trends of both methods.

Finally, suggesting several subjects to extend this work for future;

- The resulting simulation and experiment performance of CoM and direction based EKF SLAM presented in this work shall be compared with performance of SP-Model based EKF SLAM. Benefits and drawbacks of these algorithms shall be compared for error performance and correspondence issue. In addition to these ones, performance of these two methods shall be compared for utilization of different kinds of non-planar feature types.
- In this work, planar features are represented with CoM and a normal vector. However, such algorithm is applicable to different kinds of 3D features once they can be represented by a point and a direction vector. So, the presented algorithm can be applied for different kinds of features such as [45], [46],

[47] and performance of the algorithm shall be tested with these kinds of non-planar features.

- To avoid unintended manipulative impact of supervisor on the data collected by sensor system, widely used methods of edge/corner/feature detection and extraction algorithms are executed. Stronger algorithms to extract valuable information from the point clouds can be searched and optimized for sensor system. By this way it may be possible to increase precision of processed data collected by sensor system.
- As stated through the work, the data association operation was done by the supervisor. Automatic association techniques can be used and the performance of such methods for this type of sensor shall be presented.
- The sensor constructed for this work is able to create the point cloud of different kinds of objects and environments. Such sensor system may be used to extract different kinds of features and the resulting features can be used in several kinds of SLAM techniques. Comparison of several techniques of feature extraction methods and several extensions of SLAM algorithms for this type of sensor may be a beneficial contribution for literature.

REFERENCES

[1] Thrun, S., Beetz, M., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Haehnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. Int. J. Robo. Res. 19, 11 (2000), 972–999.

[2] S. Thrun, W. Burgard, D. Fox "Probabilistic Robotics". Book, The MIT Press, September 2005.

[3] Weingarten, J. Feature-based 3D SLAM. PhD Thesis, Swiss Federal Institute of Technology Lausanne, EPFL, no 3601, Dir.: Roland Siegwart,(2006).

[4] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. Artificial Intelligence, 114(1-2):3–55, 1999.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1999.

[6] Birk, A., & Carpin, S. (2006). Merging occupancy grids from multiple robots. Proceedings of the IEEE, 94(7), 1384–1397.

[7]T. Bailey. Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, Univ. of Sydney, 2002.

[8] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments" Robotics and Autonomous Systems, vol. 45, pp. 181–198, 2003.

[9] Nüchter, A., Surmann, H., Lingemann, K., Hertzberg, J., & Thrun, S. 2004. 6D SLAM with an Application in autonomous mine mapping. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 1998–2003, New Orleans, USA.

[10] K. O. Arras, J. A. Castellanos, M. Schilt, and R. Siegwart. Feature based multi-hypothesis localization and tracking using geometric constraints. Robotics and Autonomous Systems, 1056, pages 1–13, 2003.

[11] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “Fast-SLAM: A factored solution to the simultaneous localization and mapping problem,” in Proc. AAAI Nat. Conf. Artif. Intell., 2002, pp. 593–598.

[12] J.E. Guivant and E.M. Nebot, “Optimization of the simultaneous localization and map-building algorithm for real-time implementation,” IEEE Trans. Robot. Automat., vol. 17, no. 3, pp. 242–257, 2001.

[13] Lemaire, T., Lacroix, S.: Monocular-vision based SLAM using line segments. In: IEEE Int. Conf. on Robotics & Automation. (2007).

[14] Ahn, S., Choi, M., Choi, J., & Chung, W. K. (2006). Data association using visual object recognition for EKF-SLAM in home environment. In Proc. of IEEE/RSJ international conference on intelligent robots and systems (pp. 2588–2594).

[15] P. de la Puente, D. Rodriguez-Losada, R. Lopez, and F. Matia, “Extraction of geometrical features in 3d environments for service robotic applications,” in HAIS, 2008.

[16] Berger, C., Lacroix, S.: Using planar facets for stereovision SLAM. In: IROS, IEEE (2008) 1606–1611

[17] P.M. Newman, J.J. Leonard, J. Neira, and J. Tardos, “Explore and return: Experimental validation of real time concurrent mapping and localization,” in Proc. IEEE Int. Conf. Robot. Automat., 2002, pp. 1802–1809.

[18] A.P. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas. Discovering higher level structure in visual slam. IEEE Trans. Robotics, 24(5):980–990, Oct. 2008.

[19] J. Kwon and K. Lee. Monocular SLAM with Locally Planar Landmarks via Geometric Rao-Blackwellized Particle Filtering on Lie Groups. In CVPR, 2010.

[20] J. Folkesson, P. Jensfelt, and H. I. Christensen. The m-space feature representation for SLAM. IEEE Trans. Robotics, 23(5):1024–1035, October 2007.

[21] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): part II,” Robotics & Automation Magazine, IEEE, vol. 13, no. 3, pp. 108–117, 2006.

[22] An Introduction to the Kalman Filter. UNCChapel - Welch, Bishop – 2006.

[23]H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part i,” Robotics & Automation Magazine, IEEE, vol. 13, no. 2, pp. 99–110, June 2006.

[24] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. Robotics and Autonomous Systems, 25(3-4): 195-207,1998.

[25] Figure taken from, http://www.cs.utexas.edu/~kuipers/handouts/S07/L5%20Markov%20localization.pdf, last visited on August 2012.

[26] S.B. Williams, “Efficient solutions to autonomous mapping and navigationproblems,” Ph.D. dissertation, Univ. Sydney, Australian Ctr. Field Robotics, 2001.

[27] Weingarten, J. and Siegwart, R. 3D SLAM using Planar Segments. In Proceedings of IROS, Beijing, October 9-15, (2006).

[28] A. Kalay. An Implementation of Mono And Stereo Slam System Utilizing Efficient Map Management Strategy. MS thesis, METU, 2008.

[29] K. Arras, N. Tomatis, and R. Siegwart, “Multisensor on-the-fly localization using laser and vision,” in Proc. of IROS’00, vol. 1, 2000, pp. 462–476.

[30] Biber, P., Andreasson, H., Duckett, T., & Schilling, A. 2004. 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS ’04, Sendai, Japan.

[31] W. Burgard, A. Derr, D. Fox, and A. B. Cremers, “Integrating global position estimation and position tracking for mobile robots: the Dynamic Markov Localization approach,” in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1998.

[32] J. S. Yedidia, W. T. Freeman, Y. Weiss, in Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer ed., Morgan Kaufmann, 2003.

[33] J. L. Crowley, P. Stelmaszyk, T. Skordas and P. Puget, "Measurement and Integration of 3-D Structures By Tracking Edge Lines", International Journal of Computer Vision, Vol 8, No. 2, July 1992.

[34] L. M. Paz, J. D. Tardo´s, and J. Neira, “Divide and conquer: EKF SLAM in O(n),” IEEE Trans. Robot., to be published.

[35] Faugeras, O., Three-dimensional Computer Vision: A Geometric Viewpoint, The MIT Press, Cambridge, Massachusetts, 1993.

[36] Sensor data sheet, http://www.roboticsconnection.com/multimedia/docs/GP2Y0A700K0F_SS.pdf, last visited on 3 August 2012.

[37] Type data sheet, http://www.roboticsconnection.com/multimedia/docs/GP2Y0A700K0F_SS.pdf, last visited on 3 August 2012.

[38] Plane fitting code adapted from, http://www.mathworks.com/products/statistics/examples.html;jsessionid=0d7aec83dbff9d61b0d02bcd4bd8?file=/products/demos/shipping/stats/orthoregdemo.html , last visited on 3 August 2012.

[39] J. W. Weingarten, G. Gruener, and R. Siegwart. Probabilistic plane fitting in 3d and an application to robotic mapping. In Proc. IEEE Int. Conf. Robotics and Automation, volume 1, pages 927–932, 2004.

[40] T. Bailey, J. Nieto, J. Guivant, M. Stevens and E. Nebot, “Consistency of the EKF-SLAM algorithm”, In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 9 - 15, 2006, Beijing, China, pp. 3562-3568.

[41] MatLab code of Bailey: http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm, last visited on 3 August 2012.

[42] Akagündüz, E., and Ulusoy, I.: “Scale and orientation invariant 3D interest point extraction using HK curvatures”. Proc. IEEE 13th Int. Conf. Computer Vision, Workshop on 3D Representation for Recognition 3DRR, ICCV 2009, Kyoto, Japan, 2009.

[43] Akagündüz, E., Ulusoy, I.: 3D object representation using transform and scale invariant 3D features. In: Proc. of the IEEE 11th International Conference on Computer Vision, pp. 1–8 (2007).

[44] Akagündüz, E. and Ulusoy, I.: “Extraction of 3D Transform and Scale Invariant Patches from Range Scans,” CVPR, 2007.

[45]http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm?iframe=true&width=80%&height=80%, last visited on 3 August 2012.

[46] N. Senthilkumaran, R. Rajesh, "Edge Detection Techniques for Image Segmentation and A Survey of Soft Computing Approaches", International Journal of Recent Trends in Engineering, Vol. 1, No. 2, PP.250-254, May 2009.

[47] http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm, last visited on 3 August 2012.

[48] http://homepages.inf.ed.ac.uk/rbf/HIPR2/zeros.htm, last visited on 3 August 2012.

[49] http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm, last visited on 3 August 2012.

APPENDIX A

EDGE DETECTORS

This appendix aims to explain the edge detection methods used for segmentation of the point cloud. For each method shown here the MATLAB 7.12.0 functions are implemented to Figure 14.

Sobel Edge Detector

This is a simple edge detection method with low computational cost. The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high transition frequency. It is used to find the approximate absolute gradient magnitude at each point of an input gray scale image. The absolute output of edge detector is expected to correspond to edges.

In Sobel edge detection method, two matrixes are used to be convolved with image. The set of matrixes offered for Sobel are shown below.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \text{A-1}$$

These two matrixes are convolved with image and two images that show vertical and horizontal transition strengths are obtained. Once these images are combined with one of the following equations;

$$G = \sqrt{G_x^2 + G_y^2} \qquad \text{A-2}$$

or

$$G = |G_x| + |G_y| \qquad \text{A-3}$$

the matrix which represents power of transitions is obtained. This G shall be filtered with a threshold value and the edges will be obtained by this way.[45]

Output of Sobel edge detector is illustrated below.

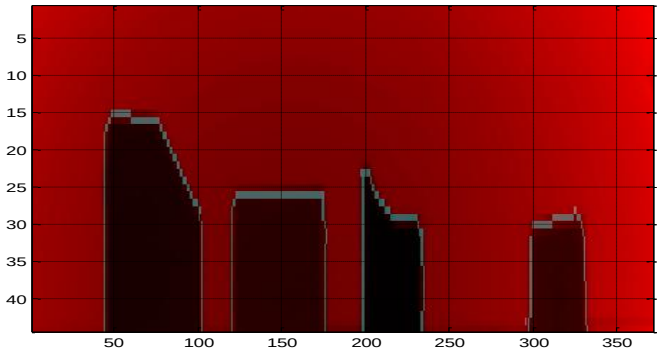


Figure 90 – Resulting edge map of Sobel edge detector (in both directions)

Robert's Cross Edge Detector

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. It thus highlights regions of high spatial frequency which often correspond to edges. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point. Similar to Sobel edge detection method, there are two convolution kernels in Robert's Cross method.

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \qquad \text{A-4}$$

These two matrixes can be convolved with image and resulting gray scale images shall be integrated via (A-2) or (A-3).

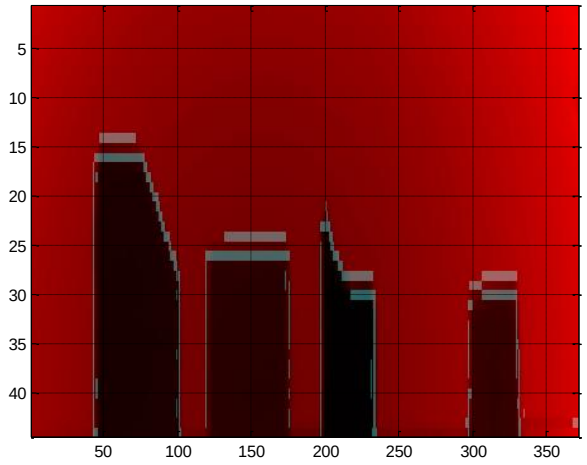


Figure 91 – Resulting edge map of Robert's Cross method.

Prewitt Edge Detector

The Prewitt edge detector is an appropriate way to estimate the magnitude and orientation of an edge. Although differential gradient edge detection needs a rather time consuming calculation to estimate the orientation from the magnitudes in the x and y-directions, the compass edge detection obtains the orientation directly from the kernel with the maximum response. The Prewitt operator is limited to 8 possible orientations. To obtain maximum edge transition direction, true kernel must be implemented, or after implementing all possible kernels, the one with maximum output could be selected.[46]

$$\text{Prewitt convolution mask for } 45° = \begin{bmatrix} +1 & +1 & +1 \\ -1 & -2 & +1 \\ -1 & -1 & +1 \end{bmatrix} \qquad \text{A-5}$$

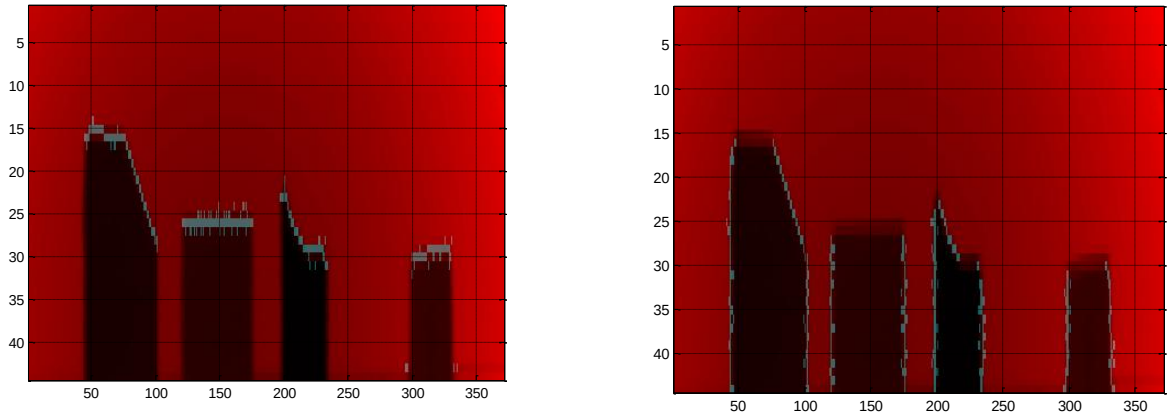


Figure 92 – Resulting edge map of Prewitt edge detector with two different masks

Laplacian of Gaussian Edge Detector

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and this property makes it a robust tool for edge detection. However since Laplacian detects all intensity changes this method is sensitive against noise. Laplacian of an image is indicated with below equation.

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad \text{A-6}$$

For a digital image this can be iterated with one of the kernels shown below.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \text{A-7}$$

However (A-7) is sensitive against noise. Due to this reason Laplacian shall be used with convolution of Gaussian. With this additional convolution operation (A-6) becomes;

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \text{A-8}$$

with graph;

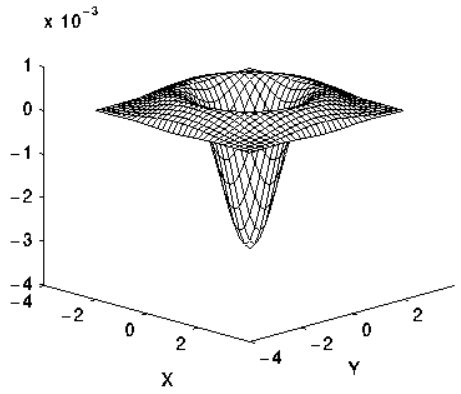


Figure 93 – Graph of Laplacian of Gaussian

When (A-8) is approximated to a discrete kernel one can find the convolution matrix shown below.[47]

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 94 - Discrete graph of Laplacian of Gaussian with σ=1.4

When Laplacian of Gaussian is implemented to Figure 14, following edge map is created.

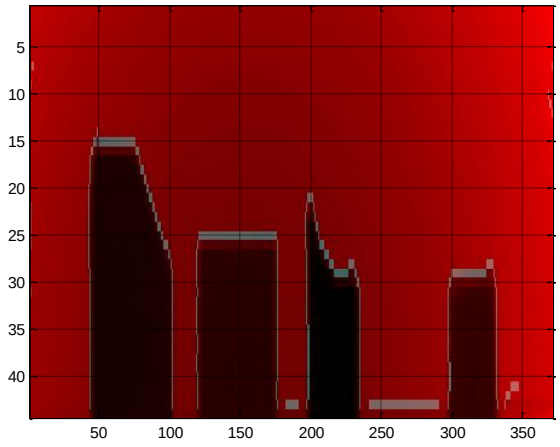


Figure 95 – Resulting edge map of LoG Edge Detector

Zero Cross Edge Detector

The zero crossing detector looks for places in the Laplacian of an image where the value of the Laplacian passes through zero (or in discrete image where Laplacian changes its sign). Such places often represent edges of objects. Zero crossing detector shall be thought as some sort of feature detector rather than an edge detector. Zero crossings lie on closed contours. Due to this reason the output of the zero crossing detector is usually an image with single pixel thickness lines showing the positions of the zero crossing points.[48] This property makes zero crossing method a suitable choice of feature extractor for distance images.

The core of the zero crossing detector is the LoG filter. As stated previously in LoG edge detector, `edges' in images give rise to zero crossings in the LoG output. For instance, below figure shows the response of a 1-D LoG filter to a step edge in the image.

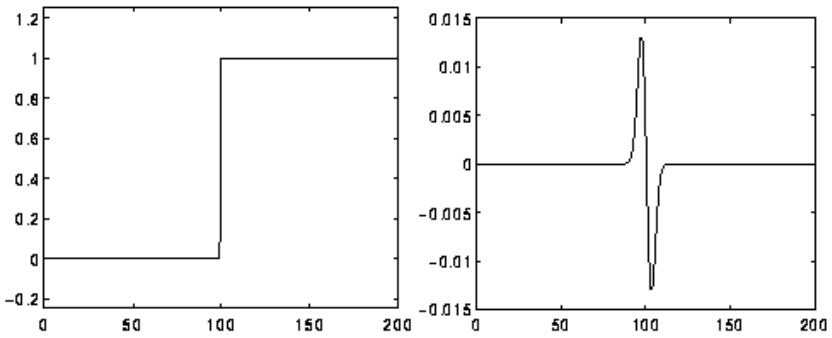


Figure 96– Graph at left shows 1D image. Graph at the right shows LoG convolved 1D image with standard deviation = 3 pixels [48]

Once Laplacian of Gaussian image is obtained, it is needed to find zero crossing points on this image. There are various methods to find these zero crossing points, the reader shall see literature for detailed information.[48]

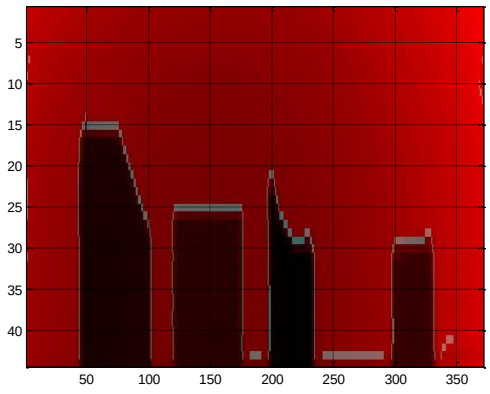


Figure 97 – Resulting edge map of Zero Cross Edge Detector. It is a very similar edge map with Figure 95

Canny Edge Detector

The Canny operator works in a multi-level process. At initial step the image is smoothed by Gaussian convolution. After that part, a simple 2-D first derivative

kernel such as Roberts Cross is applied to image. By this way regions of the image with high first spatial derivatives can be obtained. Edges give rise to ridges in the gradient image. The algorithm then tracks along the top of these ridges and erase all pixels that are not actually on the ridge top. So the algorithm creates a thin line in the output. The tracking process exhibits hysteresis controlled by two thresholds: T1 and T2, with T1 > T2. Tracking can only begin at a point on a ridge higher than T1. Tracking then continues in both directions out from that point until the height of the ridge falls below T2. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.[49]

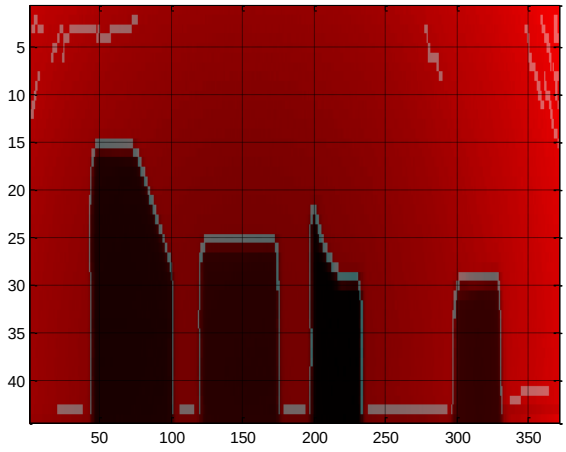


Figure 98 – Resulting edge map of Canny Edge Detector

APPENDIX B

KALMAN FILTER JACOBIAN EQUATIONS

Open form of elements of equation 3-55 is as follows:

$$\sigma\hat{r}_t^i/\sigma x_t^r = -\ \delta_t^x/\hat{r}_t^i \quad \text{B-1}$$

$$\sigma\hat{r}_t^i/\sigma y_t^r = -\ \delta_t^y/\hat{r}_t^i \quad \text{B-2}$$

$$\sigma\hat{r}_t^i/\sigma x_t^i = \ \delta_t^x/\hat{r}_t^i \quad \text{B-3}$$

$$\sigma\hat{r}_t^i/\sigma y_t^i = \ \delta_t^y/\hat{r}_t^i \quad \text{B-4}$$

$$\sigma\hat{r}_t^i/\sigma z_t^i = \ \delta_t^z/\hat{r}_t^i \quad \text{B-5}$$

$$\sigma\hat{\theta}_t^i/\sigma x_t^r = \ \delta_t^y/\left(\delta_t^{x^2} + \ \delta_t^{y^2}\right) \quad \text{B-6}$$

$$\sigma\hat{\theta}_t^i/\sigma y_t^r = \ -\delta_t^x/\left(\delta_t^{x^2} + \ \delta_t^{y^2}\right) \quad \text{B-7}$$

$$\sigma\hat{\theta}_t^i/\sigma\alpha_t^r = \ -1 \quad \text{B-8}$$

$$\sigma\hat{\theta}_t^i/\sigma x_t^i = \ -\ \delta_t^y/\left(\delta_t^{x^2} + \ \delta_t^{y^2}\right) \quad \text{B-9}$$

$$\sigma\hat{\theta}_t^i/\sigma y_t^i = \delta_t^x/\left(\delta_t^{x^2} + \delta_t^{y^2}\right) \quad \text{B-10}$$

$$\sigma\hat{\rho}_t^i/\sigma x_t^r = \delta_t^x\delta_t^z/\left(\hat{r}_t^{i^2}\sqrt{\delta_t^{x^2} + \delta_t^{y^2}}\right) \quad \text{B-11}$$

$$\sigma\hat{\rho}_t^i/\sigma y_t^r = \delta_t^y\delta_t^z/\left(\hat{r}_t^{i^2}\sqrt{\delta_t^{x^2} + \delta_t^{y^2}}\right) \quad \text{B-12}$$

$$\sigma\hat{\rho}_t^i/\sigma x_t^i = -\delta_t^x\delta_t^z/\left(\hat{r}_t^{i^2}\sqrt{\delta_t^{x^2} + \delta_t^{y^2}}\right) \quad \text{B-13}$$

$$\sigma\hat{\rho}_t^i/\sigma y_t^i = -\delta_t^y\delta_t^z/\left(\hat{r}_t^{i^2}\sqrt{\delta_t^{x^2} + \delta_t^{y^2}}\right) \quad \text{B-14}$$

$$\sigma\hat{\rho}_t^i/\sigma z_t^i = \sqrt{\delta_t^{x^2} + \delta_t^{y^2}}/\hat{r}_t^{i^2} \quad \text{B-15}$$

$$\sigma\hat{\alpha}_t^i/\sigma\alpha_t^r = -1 \quad \text{B-16}$$

$$\sigma\hat{\alpha}_t^i/\sigma\alpha_t^i = 1 \quad \text{B-17}$$

APPENDIX C

KALMAN FILTER JACOBIAN INITIALIZATION EQUATIONS

Open form of elements of matrix 3-68 are as follows:

$$\sigma x_t^j/\sigma r_t^j = \cos(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \quad \text{C-1}$$

$$\sigma x_t^j/\sigma \theta_t^j = -r_t^j \sin(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \quad \text{C-2}$$

$$\sigma x_t^j/\sigma \rho_t^j = -r_t^j \cos(\alpha_t^r + \theta_t^j)\sin(\rho_t^j) \quad \text{C-3}$$

$$\sigma y_t^j/\sigma r_t^j = \sin(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \quad \text{C-4}$$

$$\sigma y_t^j/\sigma r_t^j = \sin(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \quad \text{C-5}$$

$$\sigma y_t^j/\sigma \theta_t^j = r_t^j \cos(\alpha_t^r + \theta_t^j)\cos(\rho_t^j) \quad \text{C-6}$$

$$\sigma y_t^j/\sigma \rho_t^j = -r_t^j \sin(\alpha_t^r + \theta_t^j)\sin(\rho_t^j) \quad \text{C-7}$$

$$\sigma z_t^j/\sigma r_t^j = \sin(\rho_t^j) \quad \text{C-8}$$

$$\sigma z_t^j/\sigma \rho_t^j = r_t^j \cos(\rho_t^j) \quad \text{C-9}$$

APPENDIX D

IRSCAN OUTPUT EXAMPLE FOR INDDOR ENVIRONMENT

Point clouds of real indoor environment collected by sensor system can be seen below. There are three point clouds below and each set of points is the average of different amount of sensor readings. Figure 100 to Figure 104 are the point clouds of average of 25 sensor readings. Figure 105 is the average of 60 readings and Figure 106 is the average of only 2 readings. At the rest of this work average of 25 readings will be used by default. Below figures are illustrated to make the reader have a stronger idea about IRSCAN.

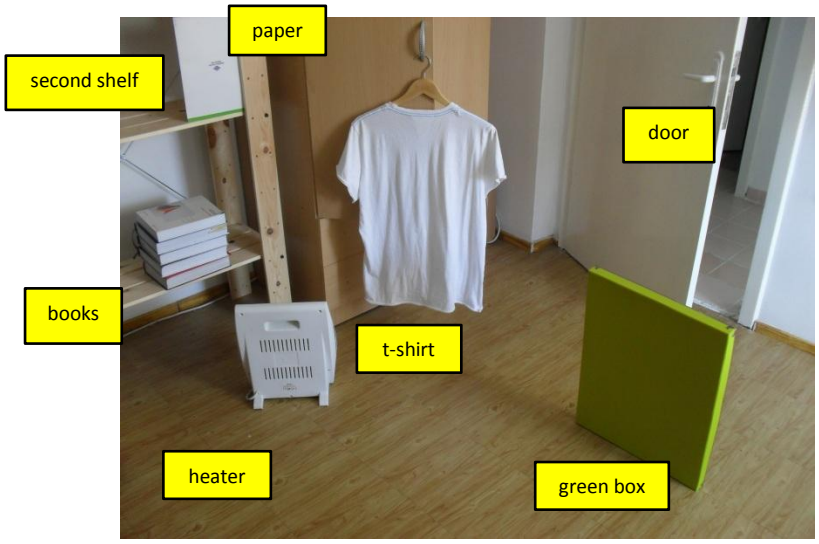


Figure 99 – Photograf of room

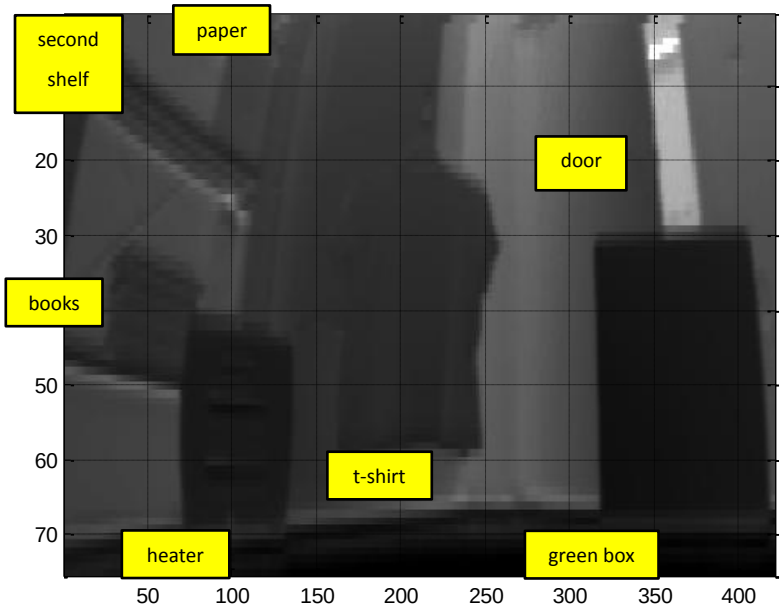


Figure 100 – Distance image of room collected by sensor system (25 readings for each pixel)

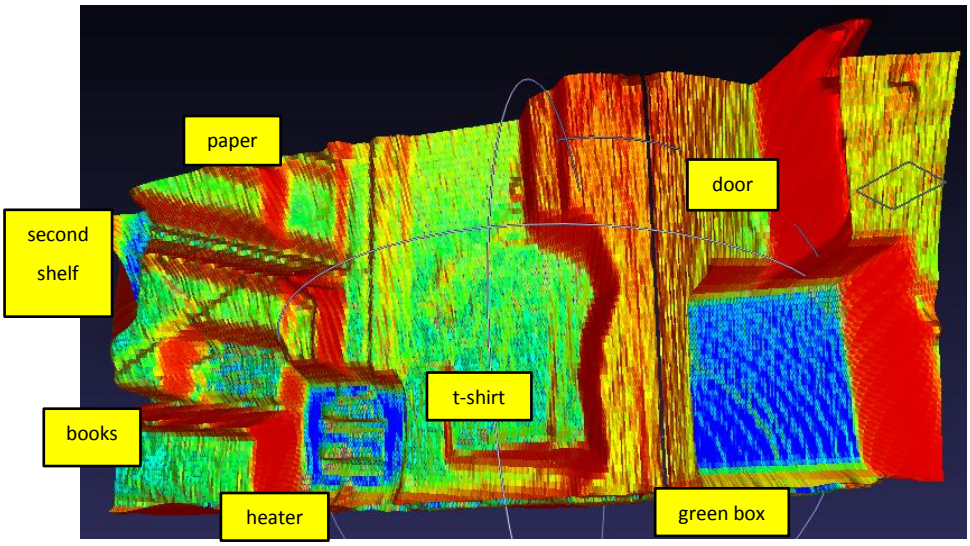


Figure 101 – constructed version of point cloud (25 readings for each point). Red zones indicate higher distance between neighboring points. Blue zones indicate lower distance between neighboring points and green zones indicate middle range of distance. Closer objects to sensor are constructed by closer points, because neighboring pixels are close to each other by trigonometric perception. Due to this reason nearest objects to sensor are colored with blue. With increasing distance of objects, distance between points is also increased, so these objects are indicated with green. Finally at transition of edges, distance between points gets their maximum value and these zones are indicated with red.

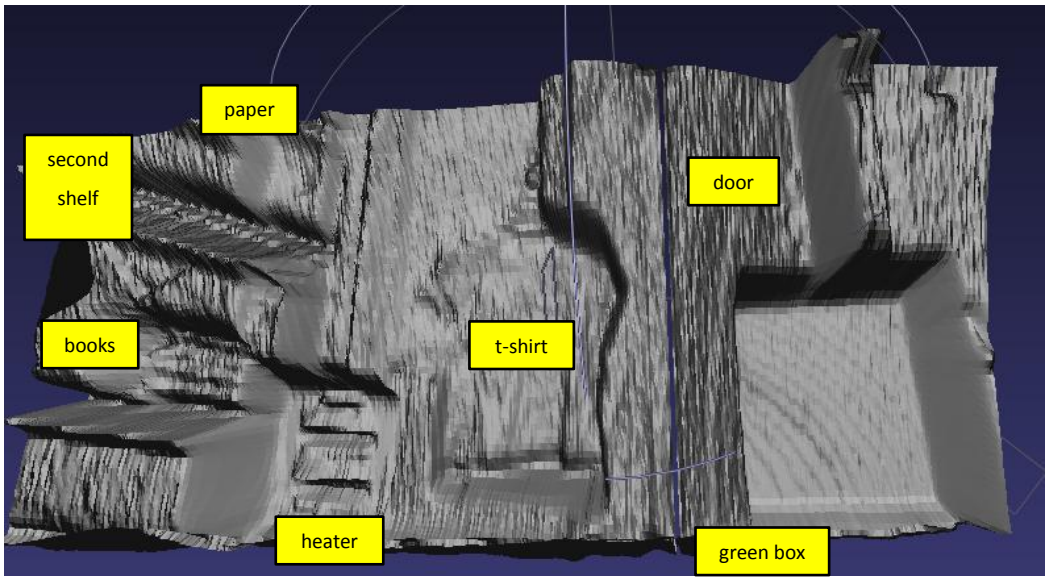


Figure 102 - 25 readings for each point. Uncolored version of Figure 101.

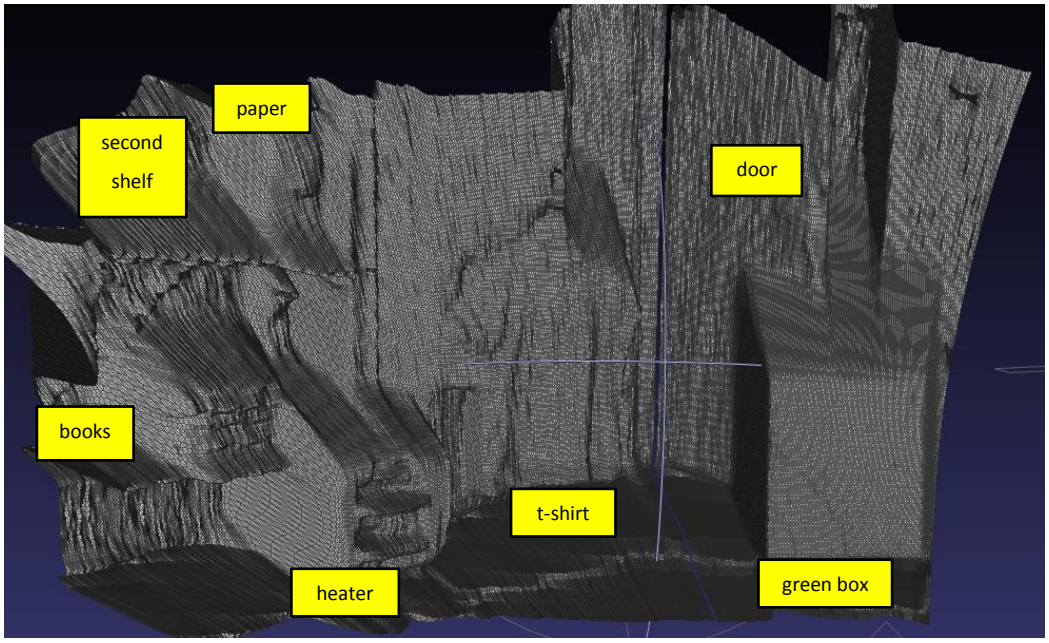


Figure 103 - 25 readings for each point

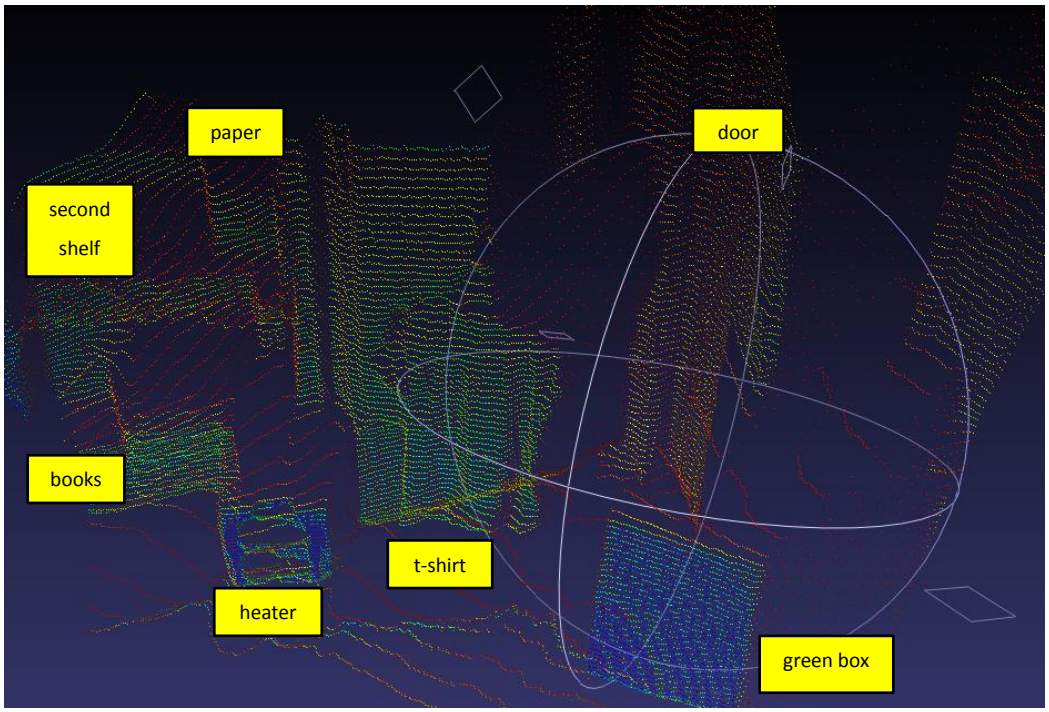


Figure 104 – Non-constructed point cloud (25 readings for each point)

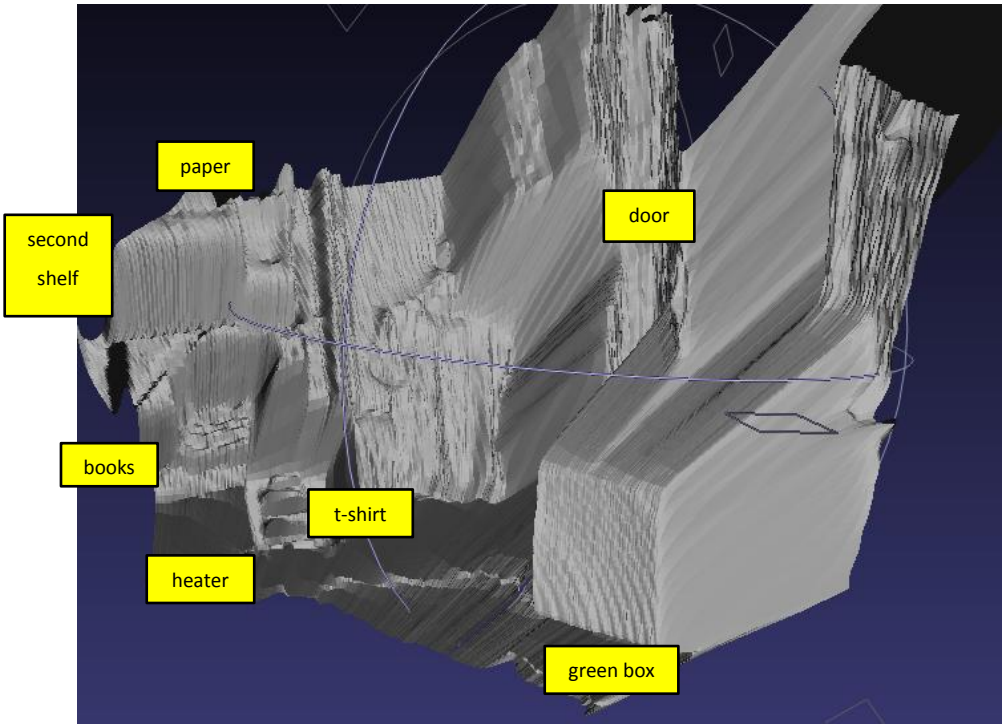


Figure 105 – Constructed point cloud of second observation (60 readings for each point)

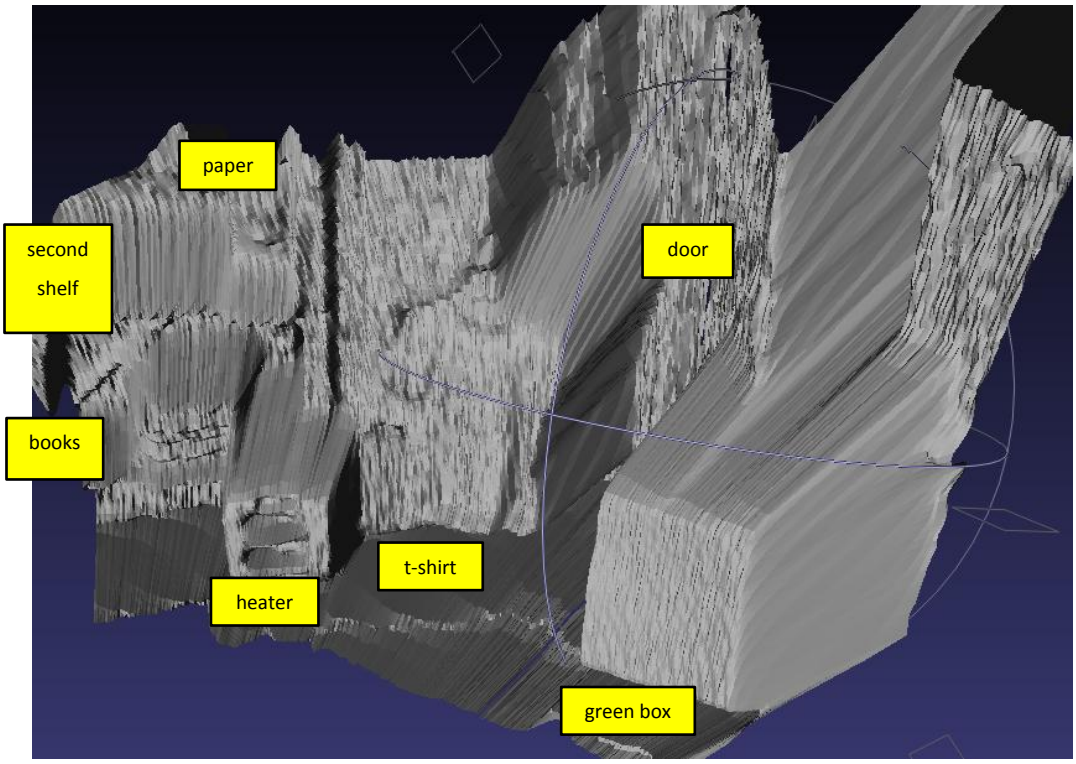


Figure 106 - Constructed point cloud of third observation (2 readings for each point)

APPENDIX E

IRSCAN NOISE MODEL

The histograms presented below shows error model of IRSCAN from different distances. Vertical axis (y-axis) of the histograms represents the number of points while horizontal axis (x-axis) represents the division of true distance of point to measured distance of point. As the reader can verify variance rate of IRSCAN fluctuates with distance instead of monotonically increasing with distance.

Table 11 – Histograms of error distributions of sensor system

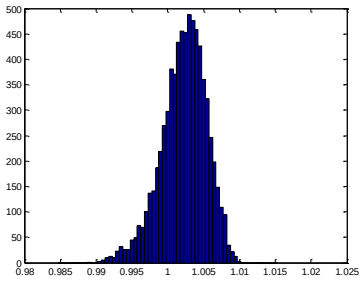
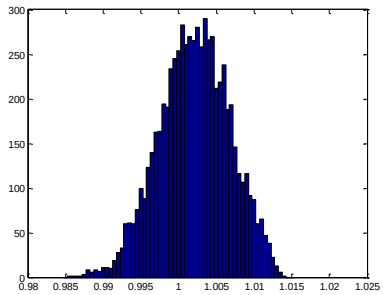
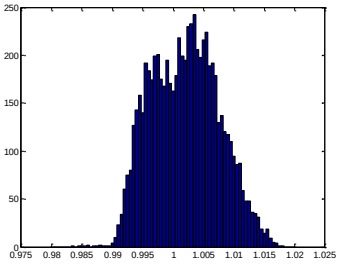
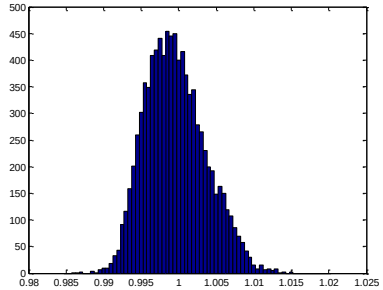
 996-1098mm Number of Points Used = 7221 Average Error = 3.35mm Average of Squares of Error = 16.0	 1098-1198mm Number of Points Used = 6710 Average Error = 4.73mm Average of Squares of Error = 34.2
 1198-1298mm Number of Points Used = 6800 Average Error = 6.06mm Average of Squares of Error = 53.8	 1298-1398mm Number of Points Used = 9064 Average Error = 4.42mm Average of Squares of Error = 29.8

Table 11 (continued)

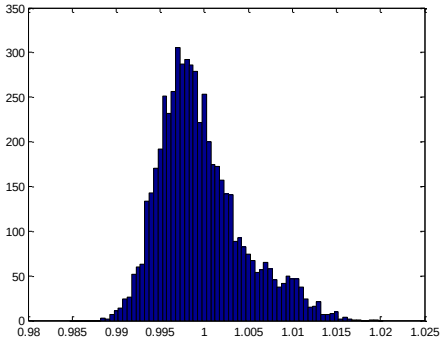
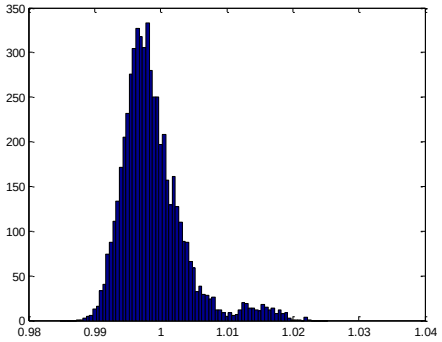
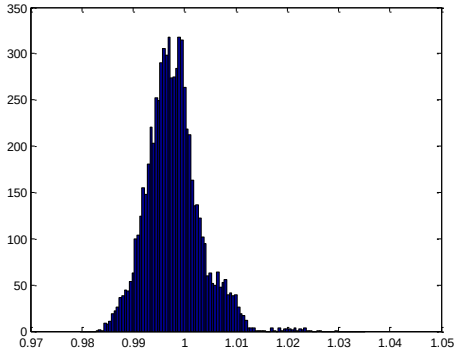
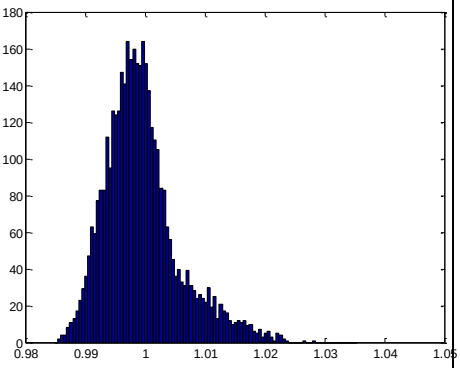
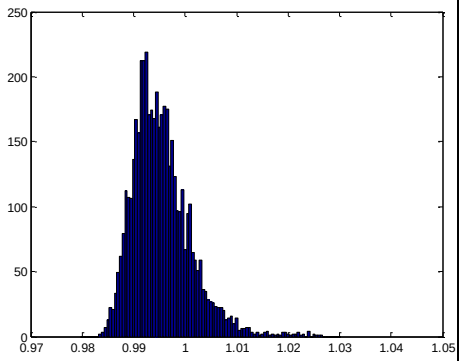
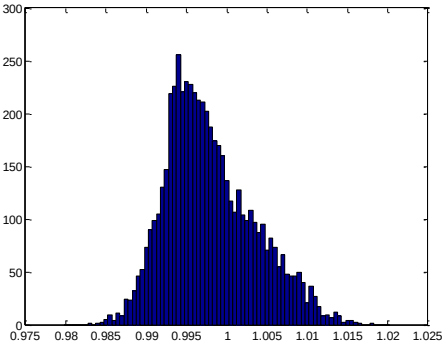
 1398-1498mm Number of Points Used = 5614 Average Error = 5.45mm Average of Squares of Error = 47.4	 1498-1598mm Number of Points Used = 5613 Average Error = 6.03mm Average of Squares of Error = 61.2
 1598-1698mm Number of Points Used = 6976 Average Error = 7.45mm Average of Squares of Error = 89.0	 1698-1798mm Number of Points Used = 3939 Average Error = 8.43mm Average of Squares of Error = 120.7
 1798-1898mm Number of Points Used = 4703 Average Error = 11.70mm Average of Squares of Error = 189.0	 1898-1998mm Number of Points Used = 5593 Average Error = 9.70mm Average of Squares of Error = 132.3

Table 11 (continued)

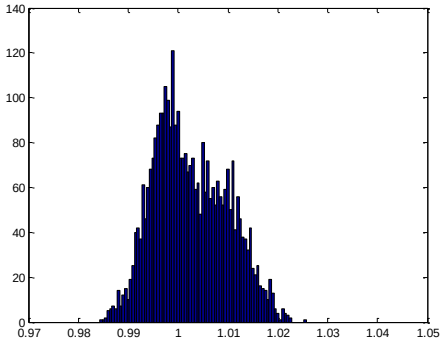
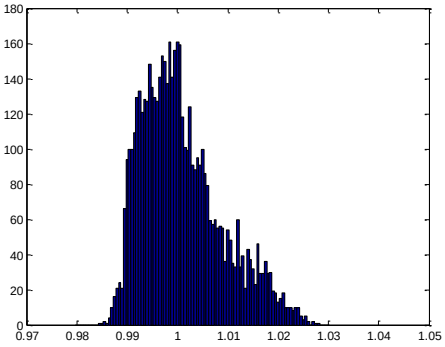
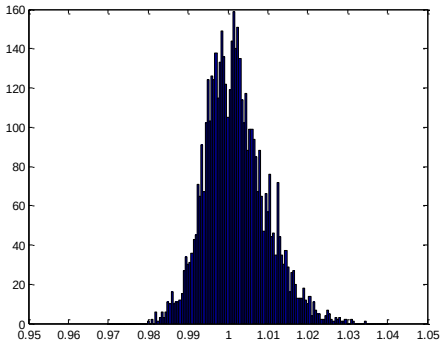
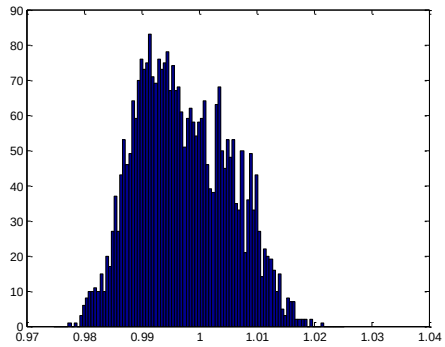
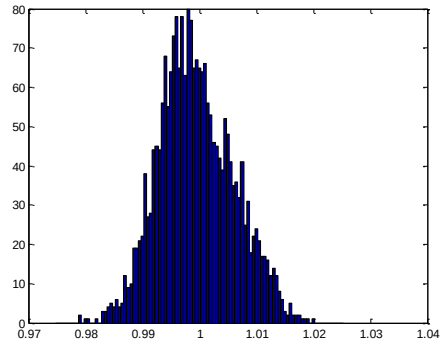
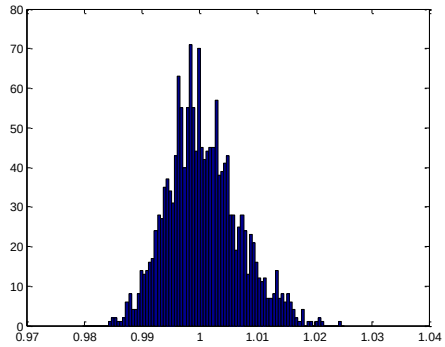
 1998-2098mm Number of Points Used = 3380 Average Error = 12.63mm Average of Squares of Error = 248.4	 2098-2198mm Number of Points Used = 5373 Average Error = 13.53mm Average of Squares of Error = 296.0
 2198-2298mm Number of Points Used = 4878 Average Error = 13.52mm Average of Squares of Error = 305.6	 2298-2398mm Number of Points Used = 3135 Average Error = 16.89mm Average of Squares of Error = 389.3
 2398-2498mm Number of Points Used = 2290 Average Error = 12.97mm Average of Squares of Error = 255.2	 2498-2598mm Number of Points Used = 1613 Average Error = 12.51mm Average of Squares of Error = 253.0

Table 11 (continued)

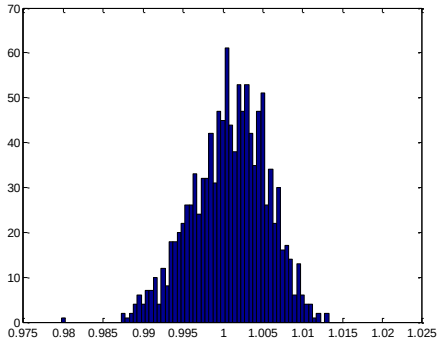
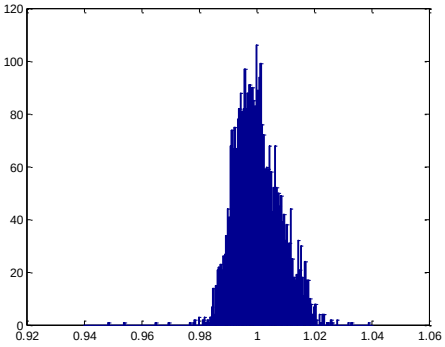
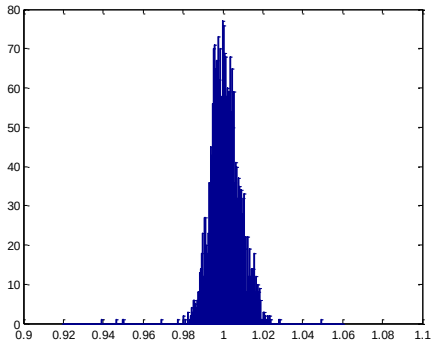
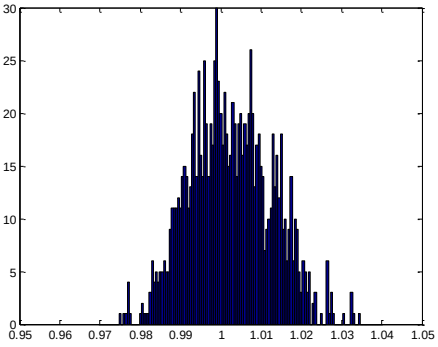
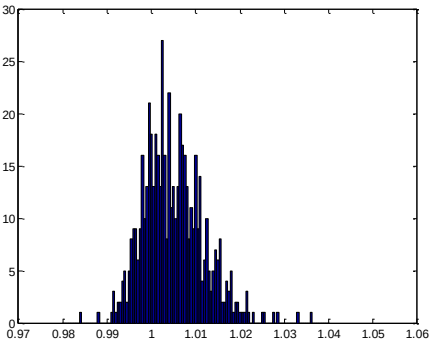
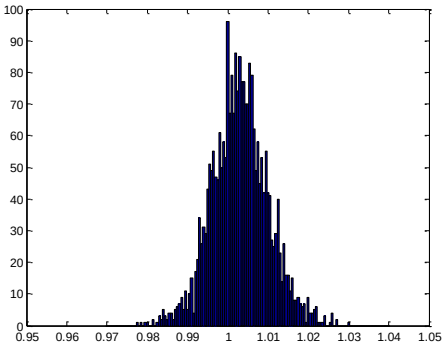
 2598-2698mm Number of Points Used = 1152 Average Error = 10.25mm Average of Squares of Error = 159.7	 2698-2798mm Number of Points Used = 3541 Average Error = 17.64mm Average of Squares of Error = 497.5
 2798-2898mm Number of Points Used = 2376 Average Error = 16.32mm Average of Squares of Error = 465.5	 2898-2998mm Number of Points Used = 1107 Average Error = 24.7mm Average of Squares of Error = 932.5
 2998-3098mm Number of Points Used = 548 Average Error = 19.53mm Average of Squares of Error = 656.3	 3098-3198mm Number of Points Used = 2567 Average Error = 18.85mm Average of Squares of Error = 579.0

Table 11 (continued)

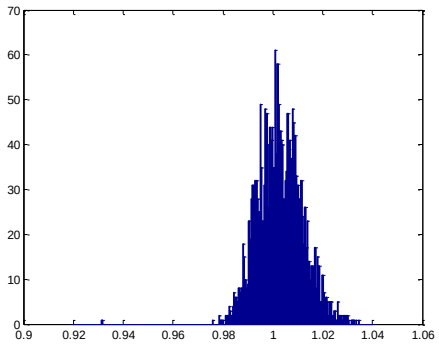
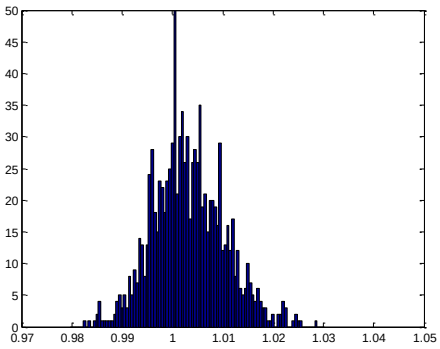
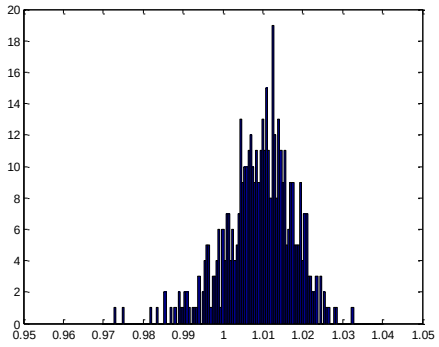
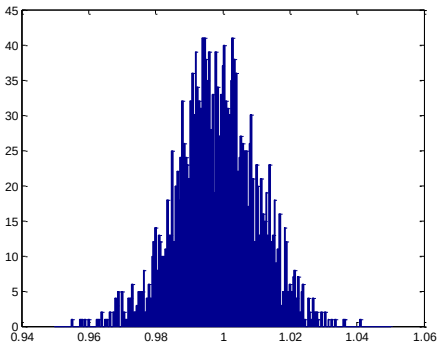
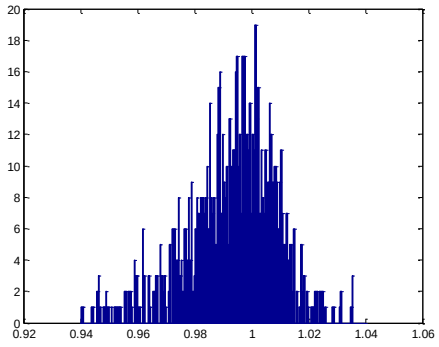
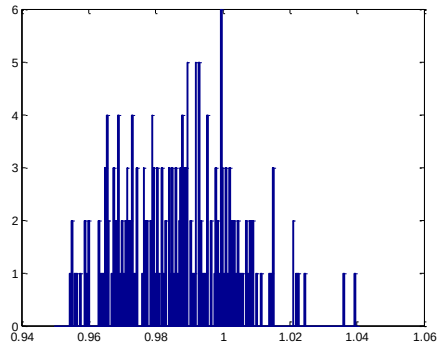
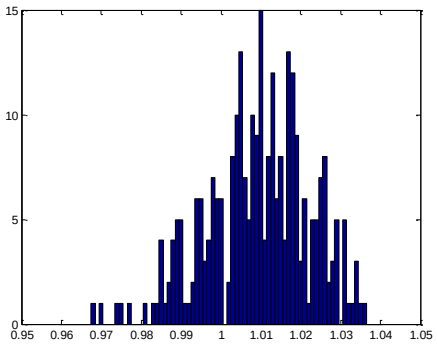
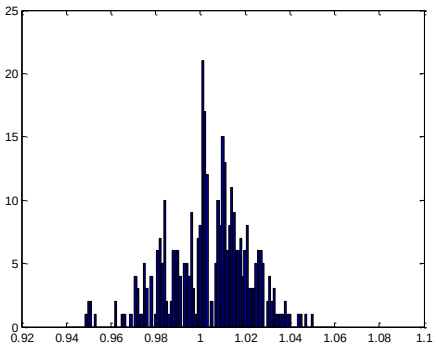
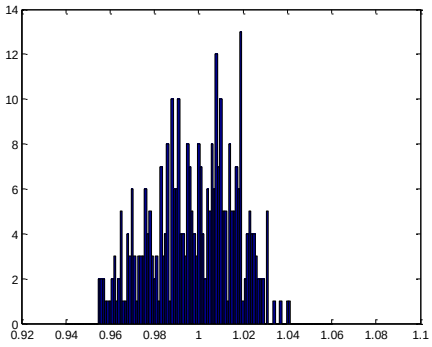
 3198-3298mm Number of Points Used = 2013 Average Error = 24.05mm Average of Squares of Error = 941.2	 3298-3398mm Number of Points Used = 990 Average Error = 20.21mm Average of Squares of Error = 674.1
 3398-3498mm Number of Points Used = 457 Average Error = 36.62mm Average of Squares of Error = 1804.7	 3498-3598mm Number of Points Used = 2061 Average Error = 34.24mm Average of Squares of Error = 1877.4
 3598-3698mm Number of Points Used = 873 Average Error = 48.77mm Average of Squares of Error = 4371.6	 3698-3898mm Number of Points Used = 201 Average Error = 65.83mm Average of Squares of Error = 6566.0

Table 11 (continued)

 3898-4098mm Number of Points Used = 238 Average Error = 53.56mm Average of Squares of Error = 5284.0	 4098-4298mm Number of Points Used = 358 Average Error = 60.58mm Average of Squares of Error = 5741.6
 4298-4598mm Number of Points Used = 335 Average Error = 67.64mm Average of Squares of Error = 6774.1	

APPENDIX F

EXPERIMENTAL RESULTS

Experiment#1

Figure 107 – object of experiment#1

In this experiment the object shown in Figure 107 is placed at x=2100mm, y=0mm, α=0°. IRSCAN, moves through x axis and collected data from following poses:

step#1: x=-27mm, y=0mm, α=-0.75°; step#2: x=273mm, y=0mm, α=-0.75°;

step#3: x=573mm, y=0mm, α=-0.75°; step#4: x=873mm, y=0mm, α=-0.75°.

Table 12 –visual data collected for experiment#1move#1

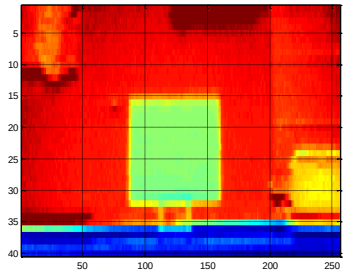
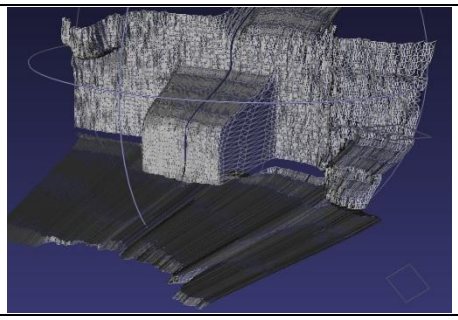
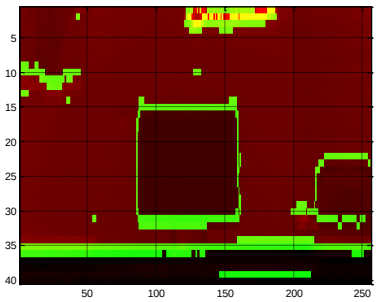
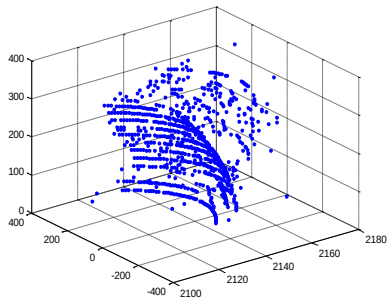
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 13 – Background truth and measured parameters for experiment#1move#1

	x	y	z	r	α	β	θ	h	w
Robot Real Pose	-27	0	115	-	-	-	-0.8	-	-
Lm#1 real pose relative to sensor	2127	28	192	2136	0.8	5.2	0.8	385	500
Lm#1 measured pose relative to sensor	2114	0	190	2123	0	5.1°	-0.9	355	480
Pnt#1 real pose relative to sensor	2130	-222	385	2176	-6.0	10.2	-	-	-
Pnt#1 measured pose relative to sensor	2135	-203	360	2175	-5.4	9.5	-	-	-
Pnt#2 real pose relative to sensor	2130	-222	0	2142	-6.0	0	-	-	-
Pnt#2 measured pose relative to sensor	2143	-204	24	2153	-5,4	0,6	-	-	-
Pnt#3 real pose relative to sensor	2124	278	0	2142	7.5	0	-	-	-
Pnt#3 measured pose relative to sensor	2130	280	24	2149	7,5	0,6	-	-	-
Pnt#4 real pose relative to sensor	2124	278	385	2176	7.5	10.2	-	-	-
Pnt#4 measured pose relative to sensor	2138	274	362	2185	7,3	9,5	-	-	-

Table 14 –visual data collected for experiment#1move#2

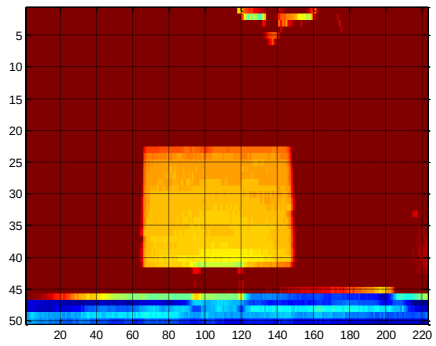
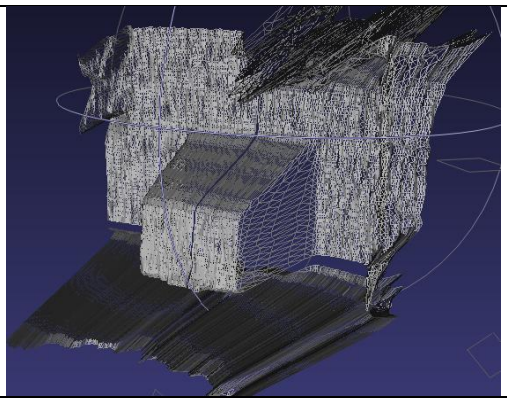
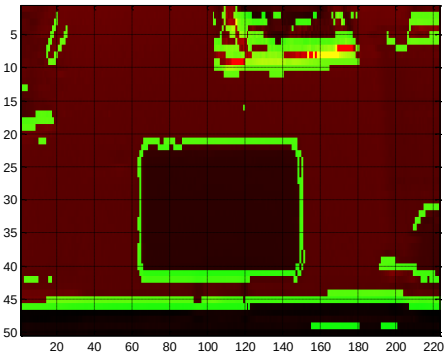
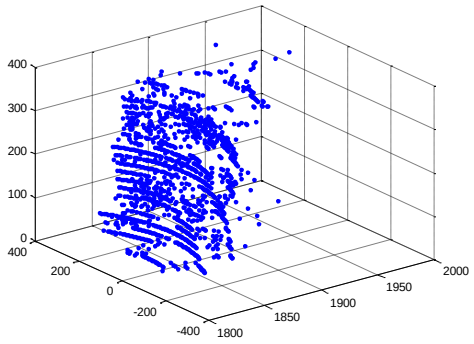
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 15 – background truth and measured parameters for experiment#1move#2

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	273	0	115	-	-	-	-0.8	-	-
Lm#1 real pose relative to sensor	1827	24	192	1837	0.8	6.0	0.8	385	500
Lm#1 measured pose relative to sensor	1855	27	204	1866	0.8	6.3	0.5	375	500
Pnt#1 real pose relative to sensor	1830	-226	385	1884	-7	11,8	-	-	-
Pnt#1 measured pose relative to sensor	1859	-202	400	1913	-6,2	12,1	-	-	-
Pnt#2 real pose relative to sensor	1830	-226	0	1844	-7	0	-	-	-

Table 15 (continued)

Pnt#2 measured pose relative to sensor	1857	-214	21	1870	-6,6	0,6	-	-	-
Pnt#3 real pose relative to sensor	1824	274	0	1844	8,5	0	-	-	-
Pnt#3 measured pose relative to sensor	1845	274	21	1866	8,4	0,6	-	-	-
Pnt#4 real pose relative to sensor	1824	274	385	1884	8,5	11,8	-	-	-
Pnt#4 measured pose relative to sensor	1861	257	402	1922	7,9	12,1	-	-	-

Table 16 –visual data collected for experiment#1move#3

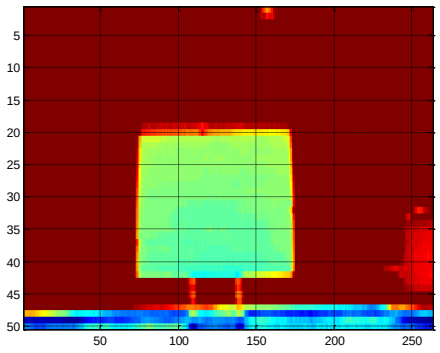
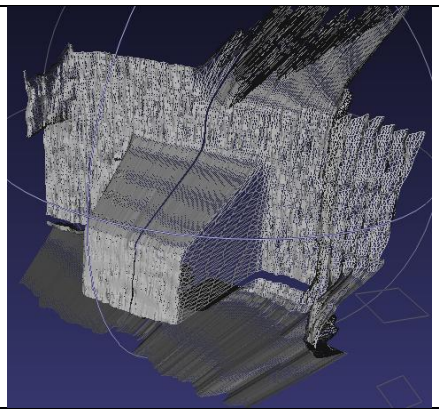
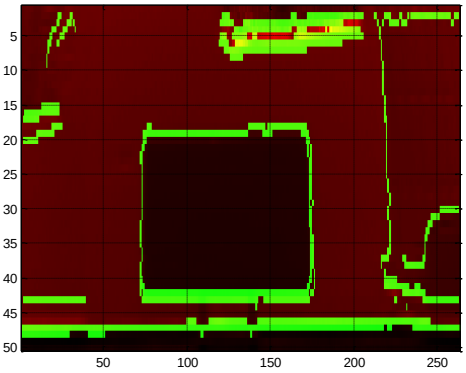
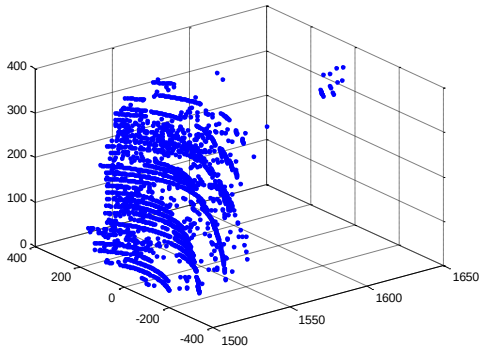
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 17 – background truth and measured parameters for experiment#1move#3

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	573	0	115	-	-	-	-0.8	-	-
Lm#1 real pose relative to sensor	1527	20	192	1539	0.8	7.2	0.8	385	500
Lm#1 measured pose relative to sensor	1533	24	181	1544	0.9	6.7	0.9	382	496
Pnt#1 real pose relative to sensor	1530	-230	385	1595	-8,5	14	-	-	-
Pnt#1 measured pose relative to sensor	1536	-197	385	1596	-7,3	14	-	-	-
Pnt#2 real pose relative to sensor	1530	-230	0	1547	-8,5	0	-	-	-
Pnt#2 measured pose relative to sensor	1518	-215	0	1533	-8,1	0	-	-	-
Pnt#3 real pose relative to sensor	1524	270	0	1547	10.0	0	-	-	-
Pnt#3 measured pose relative to sensor	1527	267	0	1550	9,9	0	-	-	-
Pnt#4 real pose relative to sensor	1524	270	385	1595	10.0	14	-	-	-
Pnt#4 measured pose relative to sensor	1530	263	368	1596	9,8	13,3	-	-	-

Table 18 –visual data collected for experiment#1move#4

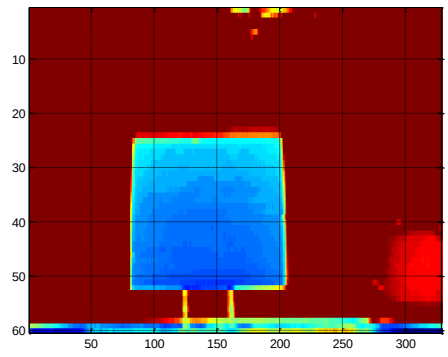
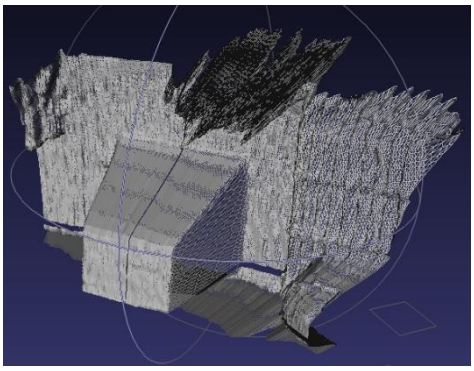
	
Distance Image	Constructed Point Cloud

Table 18 (continued)

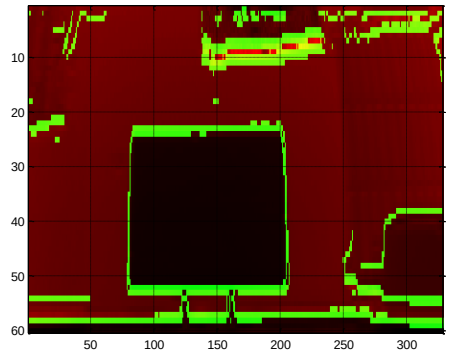
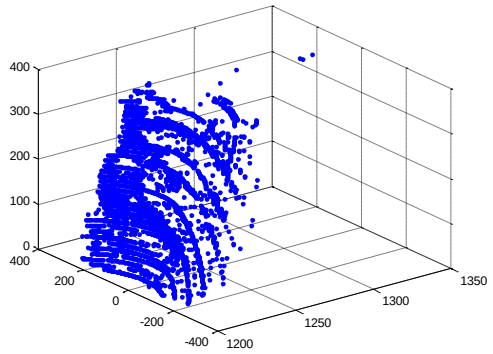
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 19 – background truth and measured parameters for experiment#1move#4

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	873	0	115	-	-	-	-0.6	-	-
Lm#1 real pose relative to sensor	1227	12	192	1242	0.7	8.9	0.6	385	500
Lm#1 measured pose relative to sensor	1224	13	182	1237	0.6	8.4	0.5	379	503
Pnt#1 real pose relative to sensor	1229	-238	385	1310	-11	17,1	-	-	-
Pnt#1 measured pose relative to sensor	1228	-215	385	1305	-9,9	17,1	-	-	-
Pnt#2 real pose relative to sensor	1229	-238	0	1252	-11	0	-	-	-
Pnt#2 measured pose relative to sensor	1212	-225	0	1233	-10,5	0	-	-	-
Pnt#3 real pose relative to sensor	1224	262	0	1252	12,1	0	-	-	-
Pnt#3 measured pose relative to sensor	1209	257	0	1236	12	0	-	-	-
Pnt#4 real pose relative to sensor	1224	262	385	1310	12,1	17,1	-	-	-
Pnt#4 measured pose relative to sensor	1221	251	369	1300	11,6	16,5	-	-	-

Table 20 – map constructed at each step with plane landmarks for experiment#1

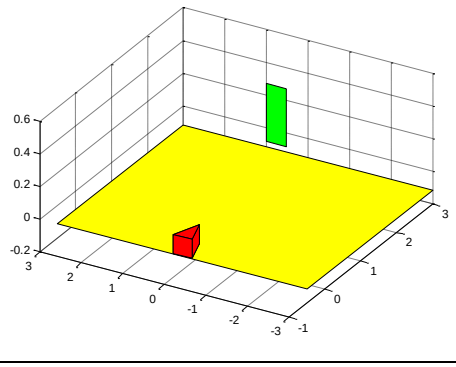
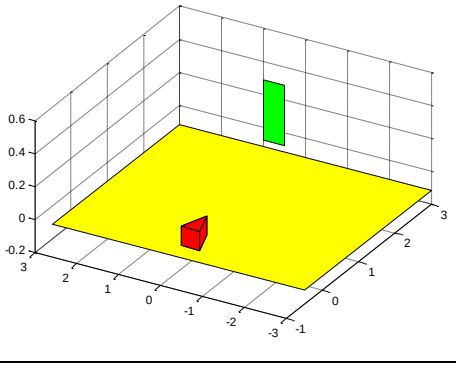
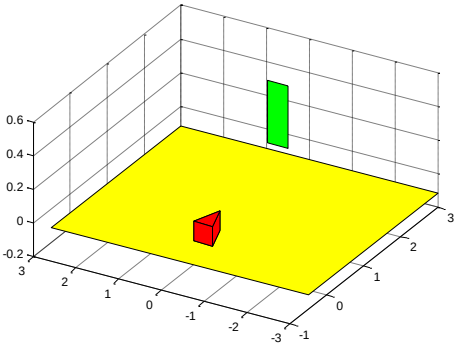
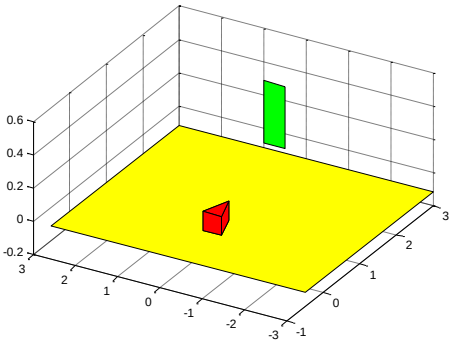
	
time = 1	time = 2
	
time = 3	time = 4

Table 21 – extracted data with background truth for plane landmarksexperiment#1move#1

time = 1	x	y	z	θ	height	width
robot real	-27	0	0(115)	-0.75	-	-
robot belief	-27	0	0	-0.75	-	-
lm#1 real	2100	0	193(308)	0	385	500
lm#1 belief	2087	-27	190	-1.61	355	480

Table 22 – extracted data with background truth for plane landmarks experiment#1move#2

time = 2	x	y	z	θ	height	width
robot real	273	0	0(115)	-0.75	-	-
robot belief	273	-5	0	-1.20	-	-
lm#1 real	2100	0	193(308)	0	385	500
lm#1 belief	2108	-21	198	-1.08	366	491

Table 23 – extracted data with background truth for plane landmarks experiment#1move#3

time = 3	x	y	z	θ	height	width
robot real	573	0	0(115)	-0.75	-	-
robot belief	573	-11	0	-1.26	-	-
lm#1 real	2100	0	193(308)	0	385	500
lm#1 belief	2113	-21	191	-0.78	372	492

Table 24 – extracted data with background truth for plane landmarks experiment#1move#4

time = 4	x	y	z	θ	height	width
robot real	873	0	0(115)	-0.56	-	-
robot belief	873	-17	0	-0.9	-	-
lm#1 real	2100	0	193(308)	0	385	500
lm#1 belief	2110	21	188	-0.75	375	496

Table 25 – map constructed at each step with point landmarks for experiment#1

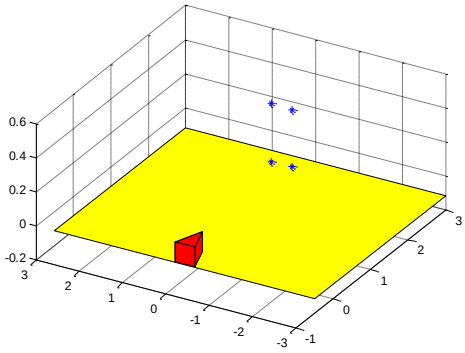
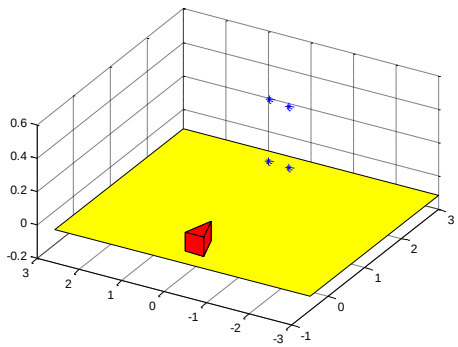
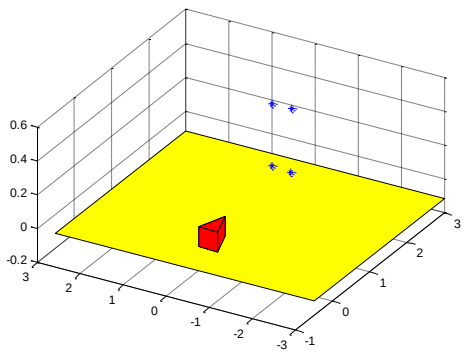
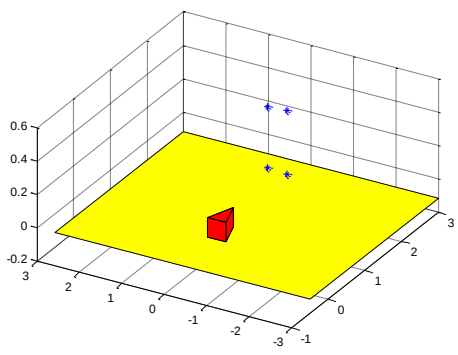
	
time = 1	time = 2
	
time = 3	time = 4

Table 26 – extracted data with background truth for point landmarks experiment#1move#1

time = 1	x	y	z	θ	height	width
robot real	-27	0	0(115)	-0.75	-	-
robot belief	-27	0	0	-0.75	-	-
pnt#1 real	2100	-250	385(500)	-	-	-
pnt#1 belief	2105	-231	360	-	-	-
pnt#2 real	2100	-250	0(115)	-	-	-
pnt#2 belief	2113	-232	24	-	-	-
pnt#3 real	2100	250	0(115)	-	-	-
pnt#3 belief	2107	253	24	-	-	-
pnt#4 real	2100	250	385(500)	-	-	-
pnt#4 belief	2114	246	361	-	-	-

Table 27 – extracted data with background truth for point landmarks experiment#1move#2

time = 2	x	y	z	θ	height	width
robot real	273	0	0(115)	-0.75	-	-
robot belief	273	-4	0	-0.62	-	-
pnt#1 real	2100	-250	385(500)	-	-	-
pnt#1 belief	2126	-229	383	-	-	-
pnt#2 real	2100	-250	0(115)	-	-	-
pnt#2 belief	2117	-235	22	-	-	-
pnt#3 real	2100	250	0(115)	-	-	-
pnt#3 belief	2127	253	22	-	-	-
pnt#4 real	2100	250	385(500)	-	-	-
pnt#4 belief	2154	243	389	-	-	-

Table 28 – extracted data with background truth for point landmarks experiment#1move#3

time = 3	x	y	z	θ	height	width
robot real	573	0	0(115)	-0.75	-	-
robot belief	573	-7	0	-0.61	-	-
pnt#1 real	2100	-250	385(500)	-	-	-
pnt#1 belief	2123	-226	385	-	-	-
pnt#2 real	2100	-250	0(115)	-	-	-
pnt#2 belief	2082	-233	12	-	-	-
pnt#3 real	2100	250	0(115)	-	-	-
pnt#3 belief	2126	251	13	-	-	-
pnt#4 real	2100	250	385(500)	-	-	-
pnt#4 belief	2123	240	378	-	-	-

Table 29 – extracted data with background truth for point landmarks experiment#1move#4

time = 4	x	y	z	θ	height	width
robot real	873	0	0(115)	-0.75	-	-
robot belief	873	-9	0	-0.24	-	-
pnt#1 real	2100	-250	385(500)	-	-	-
pnt#1 belief	2097	-225	381	-	-	-
pnt#2 real	2100	-250	0(115)	-	-	-
pnt#2 belief	2072	-234	7	-	-	-
pnt#3 real	2100	250	0(115)	-	-	-
pnt#3 belief	2102	247	7	-	-	-
pnt#4 real	2100	250	385(500)	-	-	-
pnt#4 belief	2103	237	374	-	-	-

Experiment#2

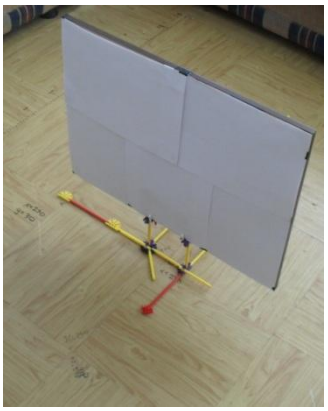


Figure 108 – object of experiment#2

In this experiment the object shown in Figure 108 is placed at x=2700mm, y=0mm, α=2°. IRSCAN, moves through x axis and collected data from following poses:

step#1: x=-27mm, y=0mm, α=-1.3°; step#2: x=273mm, y=0mm, α=-0.75°;

step#3: x=573mm, y=0mm, α=-0.93°; step#4: x=873mm, y=0mm, α=-1.13°;

step#5: x=1173mm, y=0mm, α=-1.13°; step#6: x=1473mm, y=0mm, α=-0.56°.

Table 30 –visual data collected for experiment#2move#1

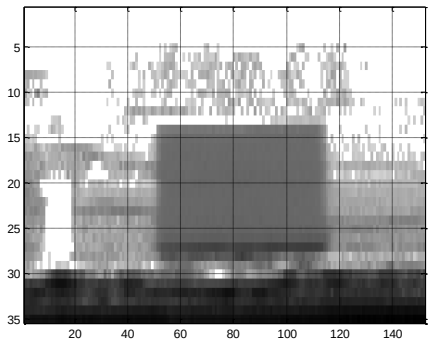
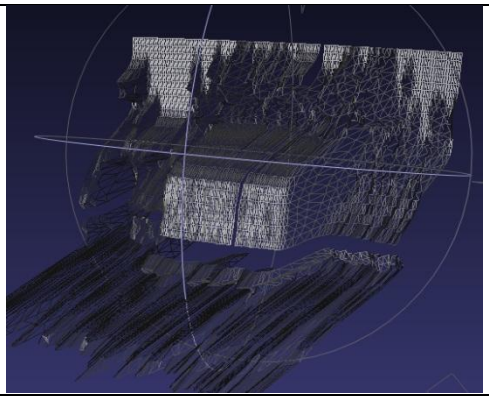
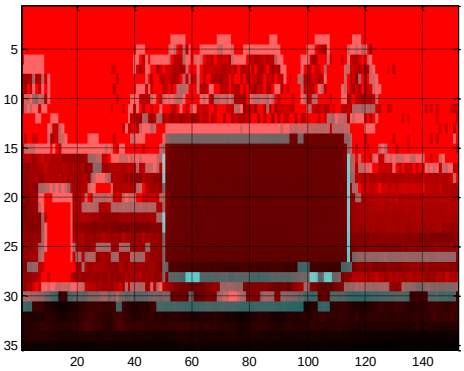
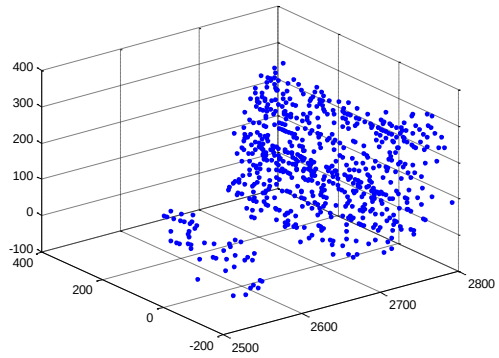
	
Distance Image	Constructed Point Cloud
	
Edge Image – Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature – Bottom part of the point cloud is undesired interference indicated at 0

Table 31 – background truth and measured parameters for experiment#2move#1

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	-27	0	115	-	-	-	-1.3	-	-
Lm#1 real pose relative to sensor	2726	62	210	2735	1,3	4,4	3,3	420	544
Lm#1 measured pose relative to sensor	2720	70	157	2725	1,5	3,3	0,7	390	537
Pnt#1 real pose relative to sensor	2741	-208	420	2781	-4,3	8,7	-	-	-
Pnt#1 measured pose relative to sensor	2773	-200	341	2801	-4,1	7	-	-	-
Pnt#2 real pose relative to sensor	2741	-208	-115	2752	-4,3	-2,4	-	-	-

Table 31 (continued)

Pnt#2 measured pose relative to sensor	2676	-184	-59	2683	-3,9	-1,3	-	-	-
Pnt#3 real pose relative to sensor	2711	333	-115	2734	7	-2,4	-	-	-
Pnt#3 measured pose relative to sensor	2695	337	-60	2716	7,1	-1,3	-	-	-
Pnt#4 real pose relative to sensor	2711	333	420	2764	7	8,7	-	-	-
Pnt#4 measured pose relative to sensor	2728	350	306	2768	7,3	6,4	-	-	-

Table 32 –visual data collected for experiment#2move#2

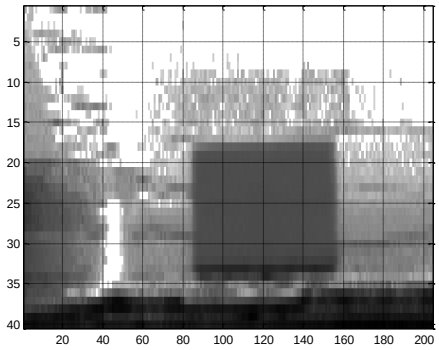
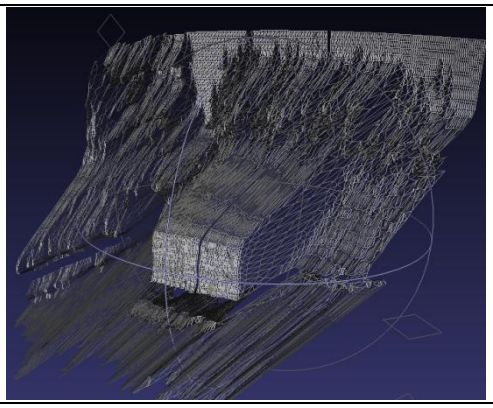
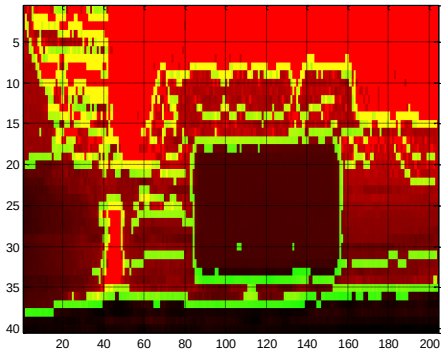
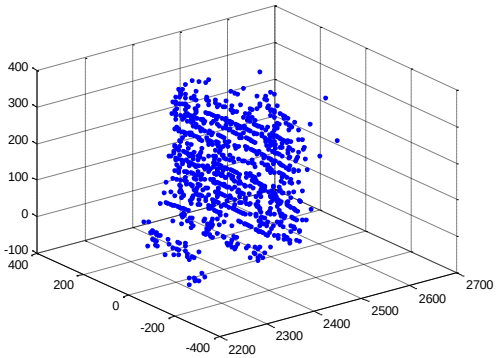
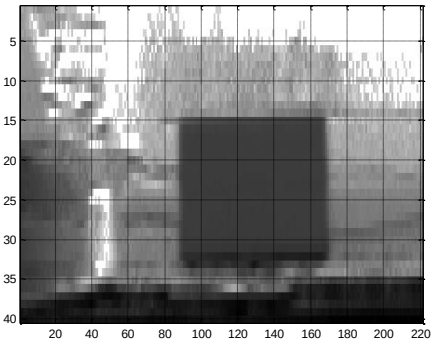
	
Distance Image	Constructed Point Cloud
	
Edge Image – Two mismarked points in the object are deleted by supervisor	Point Cloud Of Depth Segmented Feature – Bottom part of the point cloud is undesired interference indicated at 0

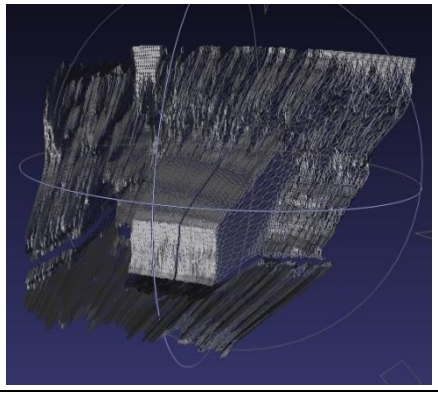
Table 33 – background truth and measured parameters for experiment#2move#2

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	273	0	115	-	-	-	-0.8	-	-
Lm#1 real pose relative to sensor	2427	32	210	2436	0,8	4,9	2,8	420	544
Lm#1 measured pose relative to sensor	2430	29	150	2435	0,7	3,5	-0,2	396	523
Pnt#1 real pose relative to sensor	2439	-239	420	2487	-5,6	9,7	-	-	-
Pnt#1 measured pose relative to sensor	2427	-231	354	2464	-5,4	8,3	-	-	-
Pnt#2 real pose relative to sensor	2439	-239	-115	2454	-5,6	-2,7	-	-	-
Pnt#2 measured pose relative to sensor	2379	-226	-53	2390	-5,4	-1,3	-	-	-
Pnt#3 real pose relative to sensor	2414	303	-115	2436	7,1	-2,7	-	-	-
Pnt#3 measured pose relative to sensor	2411	277	-54	2427	6,6	-1,3	-	-	-
Pnt#4 real pose relative to sensor	2414	303	420	2469	7,1	9,8	-	-	-
Pnt#4 measured pose relative to sensor	2417	310	326	2459	7,3	7,6	-	-	-

Table 34 –visual data collected for experiment#2move#3



Distance Image



Constructed Point Cloud

Table 34 (continued)

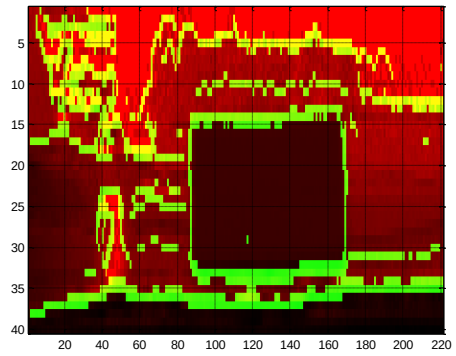
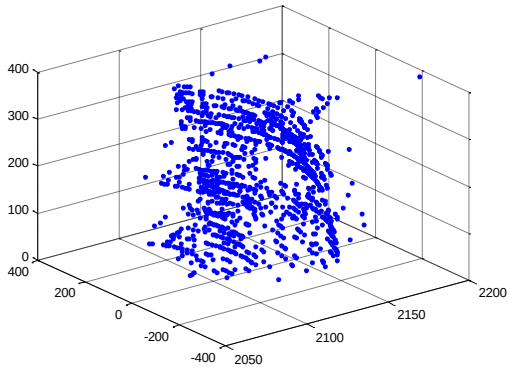
	
Edge Image –Mismarked point in the object is deleted by supervisor	Point Cloud Of Depth Segmented Feature

Table 35 – background truth and measured parameters for experiment#2move#3

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	573	0	115	-	-	-	-0.9	-	-
Lm#1 real pose relative to sensor	2127	35	210	2137	0,9	5,6	2,9	420	544
Lm#1 measured pose relative to sensor	2130	36	186	2138	1	5	1,2	372	534
Pnt#1 real pose relative to sensor	2140	-236	420	2194	-6,3	11	-	-	-
Pnt#1 measured pose relative to sensor	2125	-216	407	2175	-5,8	10,8	-	-	-
Pnt#2 real pose relative to sensor	2140	-236	-115	2156	-6,3	-3,1	-	-	-
Pnt#2 measured pose relative to sensor	2143	-239	24	2157	-6,4	0,6	-	-	-
Pnt#3 real pose relative to sensor	2113	305	-115	2138	8,2	-3,1	-	-	-
Pnt#3 measured pose relative to sensor	2136	295	0	2157	7,9	0	-	-	-
Pnt#4 real pose relative to sensor	2113	305	420	2176	8,2	11,1	-	-	-
Pnt#4 measured pose relative to sensor	2108	313	358	2160	8,4	9,5	-	-	-

Table 36 –visual data collected for experiment#2move#4

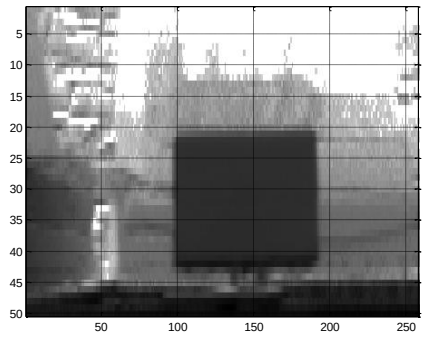
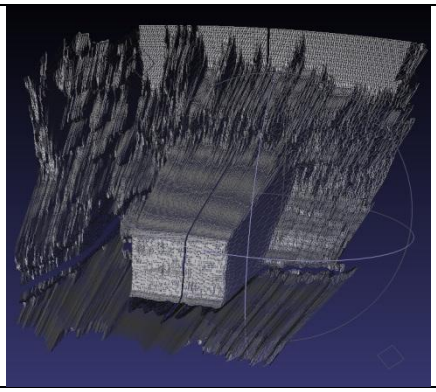
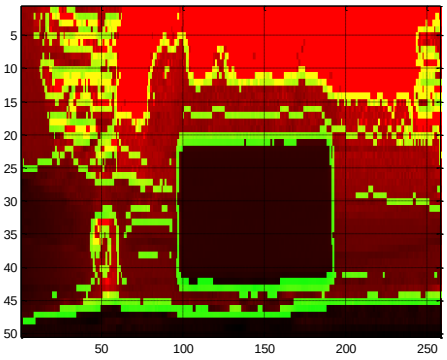
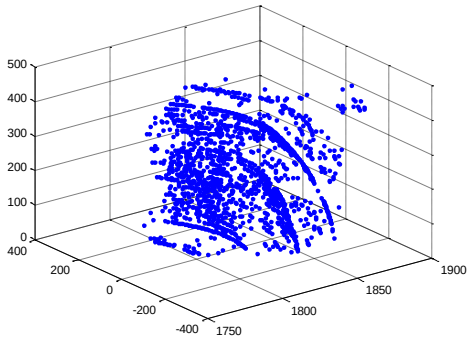
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 37 – background truth and measured parameters for experiment#2move#4

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	873	0	115	-	-	-	-1.1	-	-
Lm#1 real pose relative to sensor	1827	36	210	1839	1,1	6,6	3,1	420	544
Lm#1 measured pose relative to sensor	1834	35	202	1846	1,1	6,3	2,3	374	540
Pnt#1 real pose relative to sensor	1841	-235	420	1903	-7,3	12,8	-	-	-
Pnt#1 measured pose relative to sensor	1830	-223	415	1890	-6,9	12,7	-	-	-
Pnt#2 real pose relative to sensor	1841	-235	-115	1859	-7,3	-3,5	-	-	-
Pnt#2 measured pose relative to sensor	1848	-250	41	1866	-7,7	1,3	-	-	-

Table 37 (continued)

Pnt#3 real pose relative to sensor	1812	307	-115	1842	9,6	-3,6	-	-	-
Pnt#3 measured pose relative to sensor	1824	307	20	1850	9,6	0,6	-	-	-
Pnt#4 real pose relative to sensor	1812	307	420	1885	9,6	12,9	-	-	-
Pnt#4 measured pose relative to sensor	1816	300	393	1882	9,4	12,1	-	-	-

Table 38 –visual data collected for experiment#2move#5

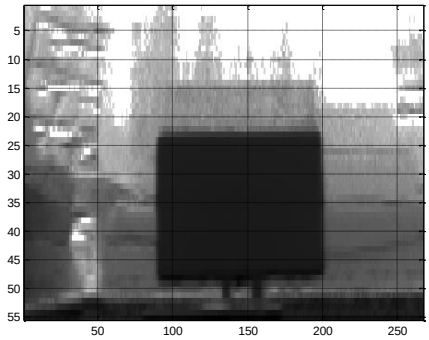
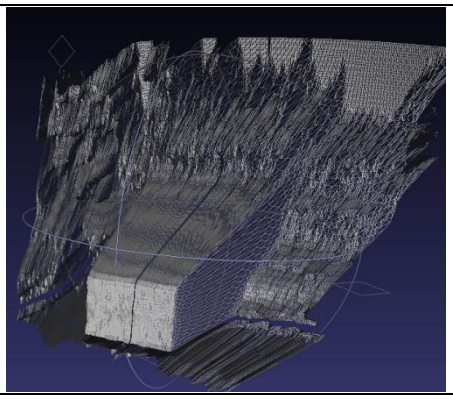
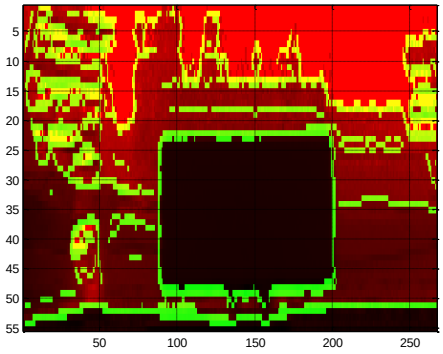
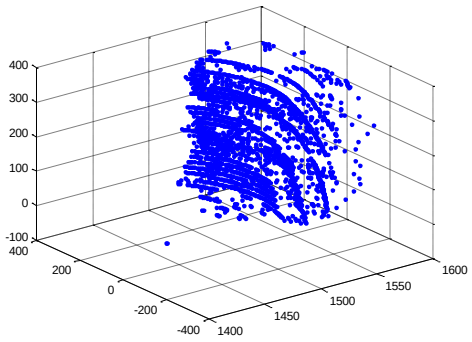
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 39 – background truth and measured parameters for experiment#2move#5

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	1173	0	115	-	-	-	-1.1	-	-
Lm#1 real pose relative to sensor	1527	30	210	1541	1,1	7,8	3,1	420	544
Lm#1 measured pose relative to sensor	1525	27	190	1537	1	7,1	2,3	384	539
Pnt#1 real pose relative to sensor	1541	-241	420	1615	-8,9	15,1	-	-	-
Pnt#1 measured pose relative to sensor	1531	-237	404	1601	-8,8	14,6	-	-	-
Pnt#2 real pose relative to sensor	1541	-241	-115	1564	-8,9	-4,2	-	-	-
Pnt#2 measured pose relative to sensor	1535	-248	17	1555	-9,2	0,6	-	-	-
Pnt#3 real pose relative to sensor	1512	301	-115	1546	11,2	-4,3	-	-	-
Pnt#3 measured pose relative to sensor	1509	295	0	1537	11,1	0	-	-	-
Pnt#4 real pose relative to sensor	1512	301	420	1598	11,2	15,2	-	-	-
Pnt#4 measured pose relative to sensor	1508	295	382	1584	11,1	14	-	-	-

Table 40 – visual data collected for experiment#2move#6

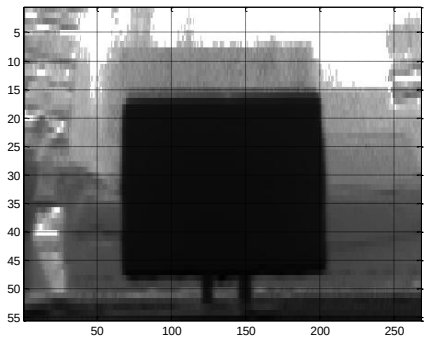
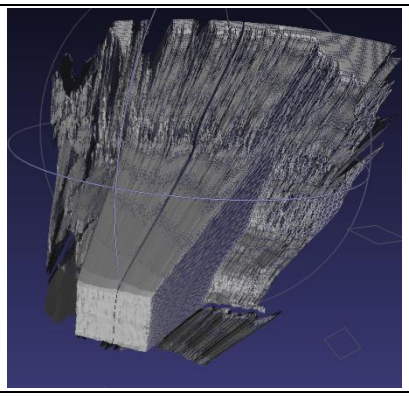
	
Distance Image	Constructed Point Cloud

Table 40 (continued)

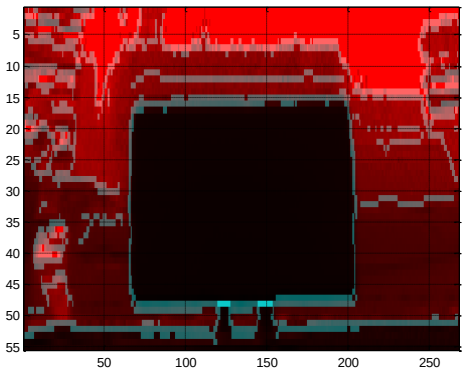
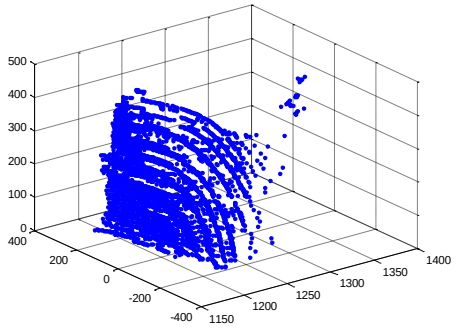
	
Edge Image – to prevent stand legs of object interfere point cloud, brighter pixels are marked by supervisor	Point Cloud Of Depth Segmented Feature

Table 41 – background truth and measured parameters for experiment#2move#6

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	1473	0	115	-	-	-	-0.6	-	-
Lm#1 real pose relative to sensor	1227	12	210	1245	0,6	9,7	2,6	420	544
Lm#1 measured pose relative to sensor	1216	25	202	1233	1,2	9,4	2,8	409	525
Pnt#1 real pose relative to sensor	1239	-259	420	1333	-11,8	18,4	-	-	-
Pnt#1 measured pose relative to sensor	1232	-228	433	1325	-10,5	19,1	-	-	-
Pnt#2 real pose relative to sensor	1239	-259	-115	1271	-11,8	-5,2	-	-	-
Pnt#2 measured pose relative to sensor	1221	-247	14	1245	-11,4	0,6	-	-	-
Pnt#3 real pose relative to sensor	1215	283	-115	1253	13,1	-5,3	-	-	-
Pnt#3 measured pose relative to sensor	1197	283	0	1230	13,3	0	-	-	-
Pnt#4 real pose relative to sensor	1215	283	420	1317	13,1	18,6	-	-	-
Pnt#4 measured pose relative to sensor	1207	290	398	1304	13,5	17,8	-	-	-

Table 42 – map constructed at each step with plane landmarks for experiment#2

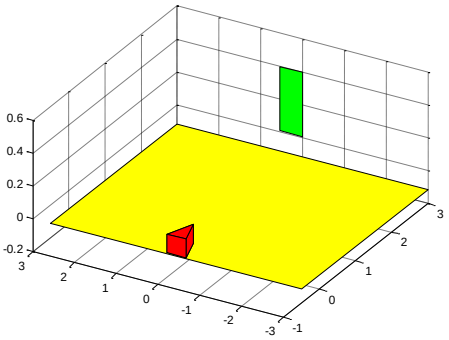
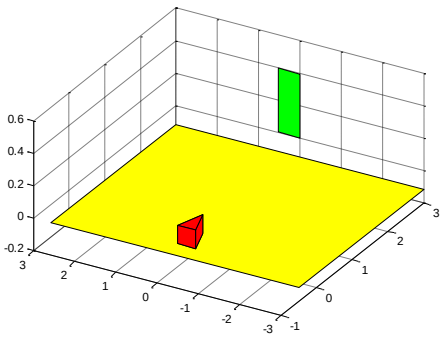
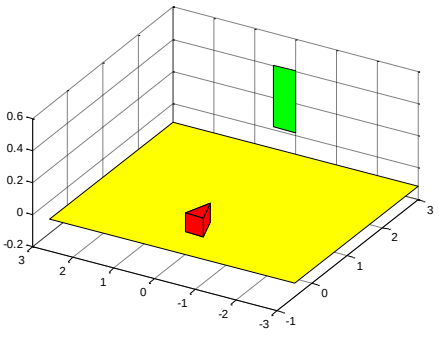
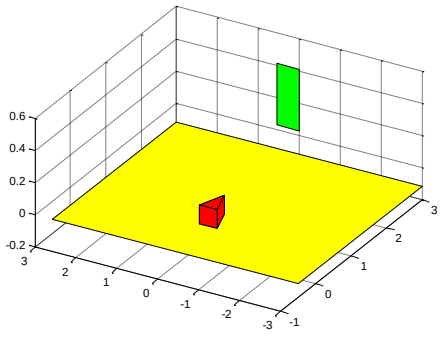
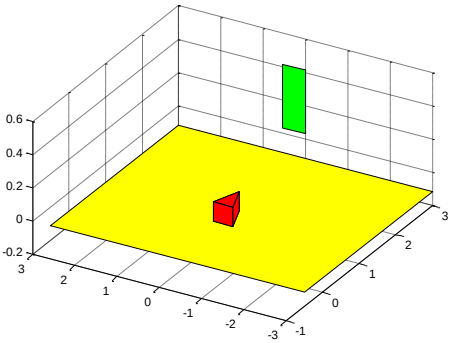
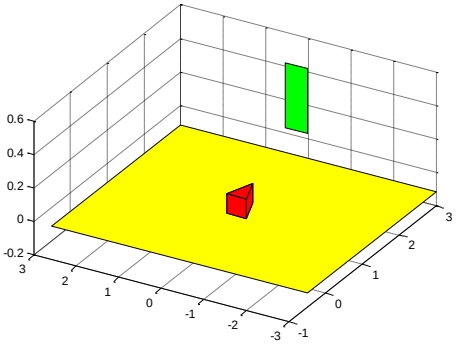
	
time = 1	time = 2
	
time = 3	time = 4
	
time = 5	time = 6

Table 43 – extracted data with background truth for plane landmarks experiment#2move#1

time = 1	x	y	z	θ	height	width
robot real	-27	0	0(115)	-1.30	-	-
robot belief	-27	0	0	-1.30	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2694	8	157	-0.64	390	537

Table 44 – extracted data with background truth for plane landmarks experiment#2move#2

time = 2	x	y	z	θ	height	width
robot real	273	0	0(115)	-0.75	-	-
robot belief	273	-5	0	-0.52	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2704	5	153	-0.67	393	530

Table 45 – extracted data with background truth for plane landmarks experiment#2move#3

time = 3	x	y	z	θ	height	width
robot real	573	0	0(115)	-0.93	-	-
robot belief	573	-8	0	-0.66	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2696	5	166	-0.16	385	531

Table 46 – extracted data with background truth for plane landmarks experiment#2move#4

time = 4	x	y	z	Θ	height	width
robot real	873	0	0(115)	-1.13	-	-
robot belief	873	-11	0	-0.71	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2684	4	177	0.47	382	534

Table 47 – extracted data with background truth for plane landmarks experiment#2move#5

time = 5	x	y	z	θ	height	width
robot real	1173	0	0(115)	-1.13	-	-
robot belief	1173	-15	0	-0.51	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2687	1	181	0.92	383	535

Table 48 – extracted data with background truth for plane landmarks experiment#2move#6

time = 6	x	y	z	θ	height	width
robot real	1473	0	0(115)	-0.56	-	-
robot belief	1473	-17	0	-0.31	-	-
lm#1 real	2700	0	210(325)	2	420	545
lm#1 belief	2674	0	186	1.44	389	533

Table 49 – map constructed at each step with point landmarks for experiment#2

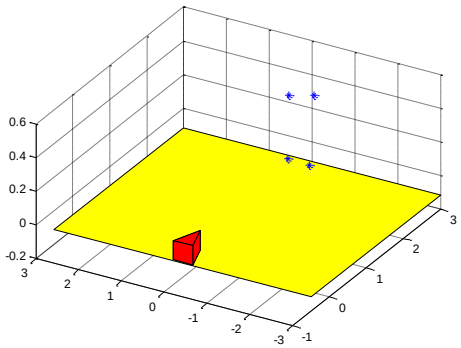
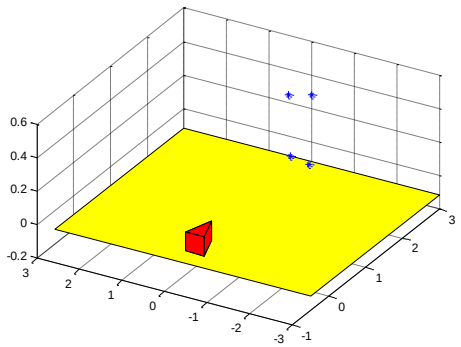
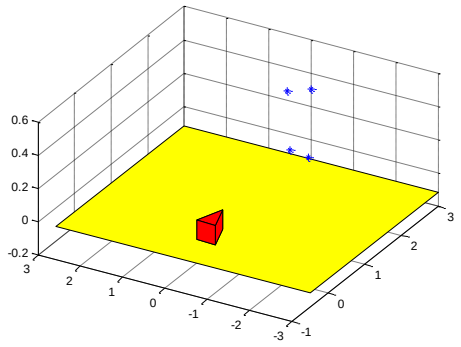
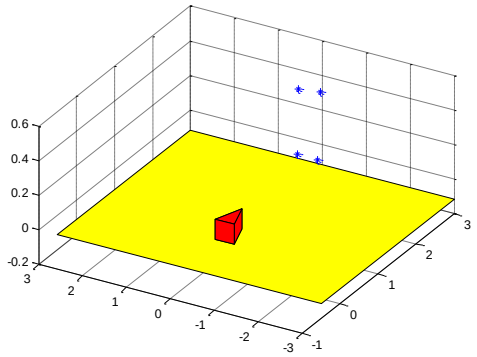
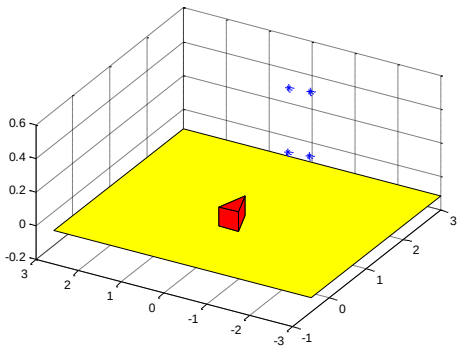
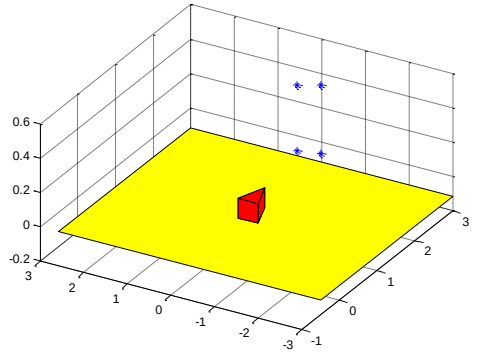
	
time = 1	time = 2
	
time = 3	time = 4
	
time = 5	time = 6

Table 50 – extracted data with background truth for point landmarks experiment#2move#1

time = 1	x	y	z	θ	height	width
robot real	-27	0	0(115)	-1.30	-	-
robot belief	-27	0	0	-1.30	-	-
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2741	-263	341	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2644	-245	-6	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2675	275	-6	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2709	287	306	-	-	-

Table 51 – extracted data with background truth for point landmarks experiment#2move#2

time = 2	x	y	z	θ	height	width
robot real	273	0	0(115)	-0.75	-	-
robot belief	273	-5	0	-0.48	-	-
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2713	-259	348	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2639	-247	-55	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2744	270	-58	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2695	286	317	-	-	-

Table 52 – extracted data with background truth for point landmarks experiment#2move#3

time = 3	x	y	z	θ	height	width
robot real	573	0	0(115)	-0.93	-	-
robot belief	573	-7	0	-0.56		-
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2700	-251	371	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2628	-250	-22	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2737	271	-35	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2672	284	331	-	-	-

Table 53 – extracted data with background truth for point landmarks experiment#2move#4

time = 4	x	y	z	θ	height	width
robot real	873	0	0(115)	-1.13	-	-
robot belief	837	-12	0	-0.62		
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2668	-251	381	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2583	-253	0	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2655	261	-14	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2696	278	354	-	-	-

Table 54 – extracted data with background truth for point landmarks experiment#2move#5

time = 5	x	y	z	θ	height	width
robot real	1173	0	0(115)	-1.13	-	-
robot belief	1173	-14	0	-0.4	-	-
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2652	-251	383	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2623	-258	5	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2640	260	-9	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2688	276	362	-	-	-

Table 55 – extracted data with background truth for point landmarks experiment#2move#6

time = 6	x	y	z	θ	height	width
robot real	1473	0	0(115)	-0.56	-	-
robot belief	1473	-14	0	-0.07	-	-
pnt#1 real	2709	-271	420(535)	-	-	-
pnt#1 belief	2688	-250	400	-	-	-
pnt#2 real	2709	-271	0(115)	-	-	-
pnt#2 belief	2669	-261	7	-	-	-
pnt#3 real	2691	271	0(115)	-	-	-
pnt#3 belief	2657	265	-6	-	-	-
pnt#4 real	2691	271	420(115)	-	-	-
pnt#4 belief	2674	275	371	-	-	-

Experiment#3

Figure 109 – object of experiment#3

In this experiment the object shown in Figure 109 is placed at x=2100mm, y=300mm, α=0°. IRSCAN, moves through x axis and collected data from following poses:

step#1: x=-27mm, y=-600mm, α=-0.75°; step#2: x=273mm, y=-600mm, α=-0.94°; step#3: x=573mm, y=-600mm, α=-0.94°; step#4: x=873mm, y=-600mm, α=-1.13°; step#5: x=1173mm, y=-600mm, α=30.18°.

Table 56 –visual data collected for experiment#3move#1

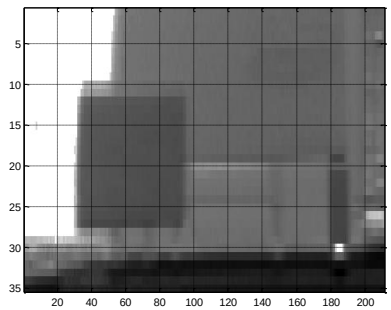
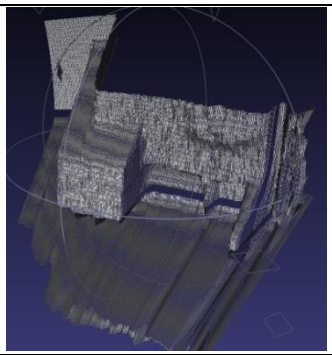
	
Distance Image	Constructed Point Cloud

Table 56 (continued)

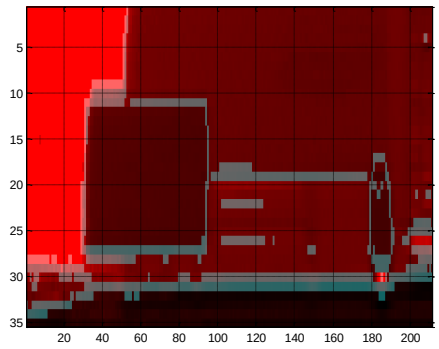
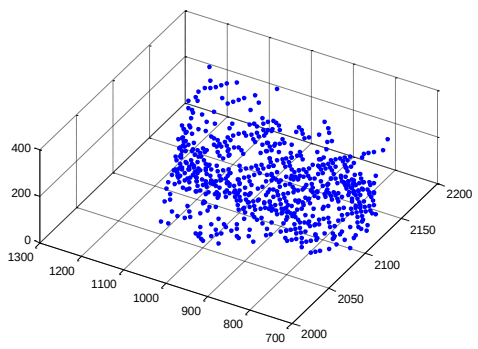
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 57 – background truth and measured parameters for experiment#3move#1

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	-27	-600	115	-	-	-	-0.8	-	-
Lm#1 real pose relative to sensor	2115	928	193	2318	23,7	4,8	0,8	420	544
Lm#1 measured pose relative to sensor	2102	928	176	2305	23,8	4,4	0,8	336	457
Pnt#1 real pose relative to sensor	2118	678	385	2257	17,7	9,8	-	-	-
Pnt#1 measured pose relative to sensor	2099	689	346	2236	18,2	8,9	-	-	-
Pnt#2 real pose relative to sensor	2118	678	0	2224	17,7	0	-	-	-
Pnt#2 measured pose relative to sensor	2125	714	25	2241	18,6	0,6	-	-	-
Pnt#3 real pose relative to sensor	2112	1178	0	2418	29,1	0	-	-	-
Pnt#3 measured pose relative to sensor	2066	1157	26	2368	29,3	0,6	-	-	-
Pnt#4 real pose relative to sensor	2112	1178	385	2448	29,1	9	-	-	-
Pnt#4 measured pose relative to sensor	2093	1154	374	2420	28,9	8,9	-	-	-

Table 58 –visual data collected for experiment#3move#2

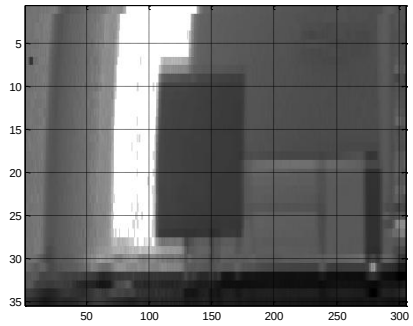
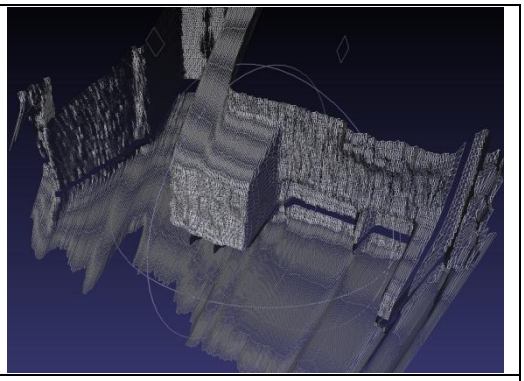
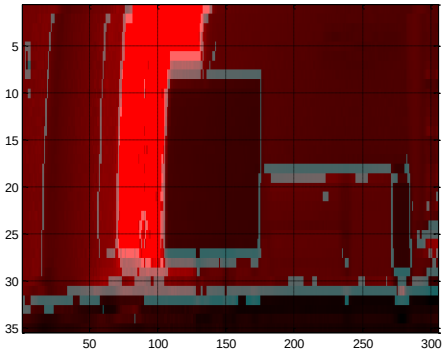
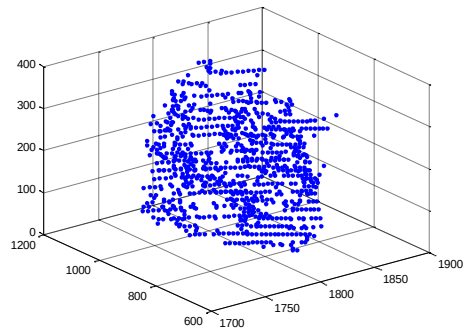
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 59 – background truth and measured parameters for experiment#3move#2

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	273	-600	115	-	-	-	-0.9	-	-
Lm#1 real pose relative to sensor	1812	930	193	2046	27,2	5,4	0,9	420	544
Lm#1 measured pose relative to sensor	1803	920	181	2033	27	5,1	3,2	367	450
Pnt#1 real pose relative to sensor	1816	680	385	1977	20,5	11,2	-	-	-
Pnt#1 measured pose relative to sensor	1833	690	373	1993	20,6	10,8	-	-	-
Pnt#2 real pose relative to sensor	1816	680	0	1939	20,5	0	-	-	-
Pnt#2 measured pose relative to sensor	1832	703	22	1963	21	0,6	-	-	-

Table 59 (continued)

Pnt#3 real pose relative to sensor	1808	1180	0	2159	33,1	0	-	-	-
Pnt#3 measured pose relative to sensor	1769	1149	23	2110	33	0,6	-	-	-
Pnt#4 real pose relative to sensor	1808	1180	385	2193	33,1	10,1	-	-	-
Pnt#4 measured pose relative to sensor	1788	1136	404	2157	32,4	10,8	-	-	-

Table 60 –visual data collected for experiment#3move#3

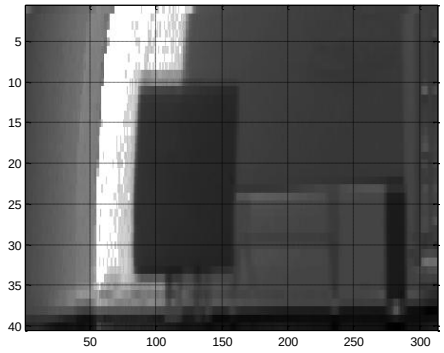
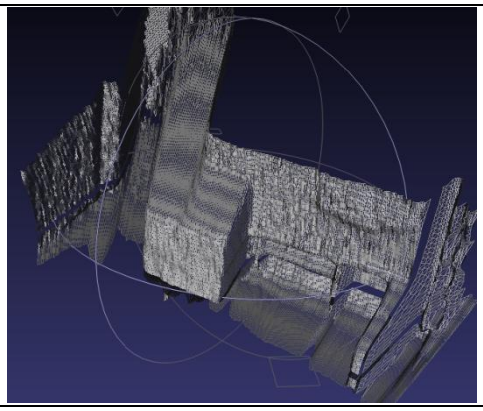
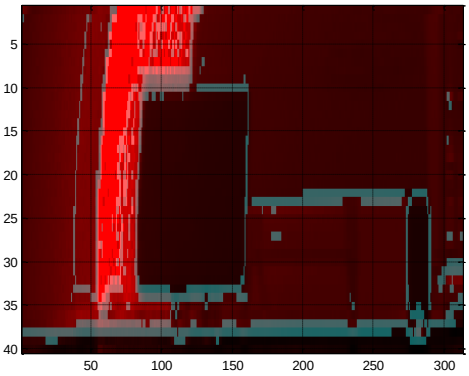
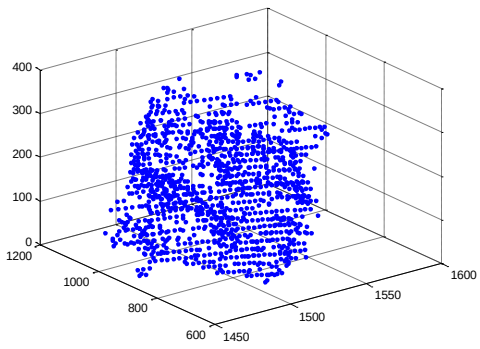
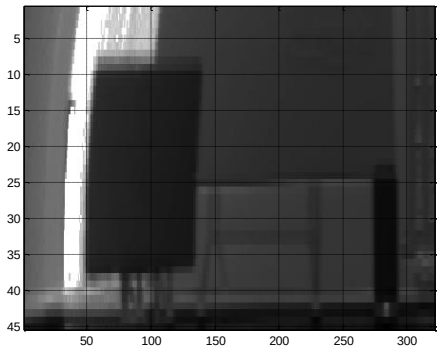
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

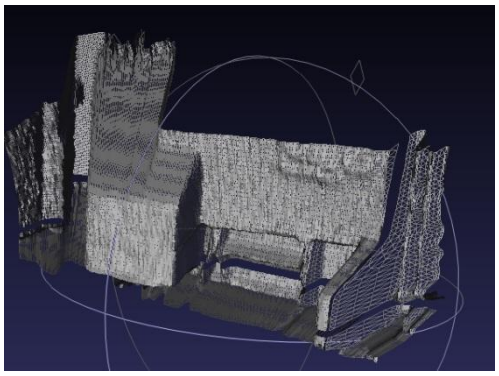
Table 61 – background truth and measured parameters for experiment#3move#3

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	573	-600	115	-	-	-	-0.9	-	-
Lm#1 real pose relative to sensor	1512	925	193	1783	31,5	6,2	0,9	420	544
Lm#1 measured pose relative to sensor	1511	911	165	1772	31,1	5,4	3,4	392	440
Pnt#1 real pose relative to sensor	1516	675	385	1704	24	13,1	-	-	-
Pnt#1 measured pose relative to sensor	1548	683	381	1734	23,8	12,7	-	-	-
Pnt#2 real pose relative to sensor	1516	675	0	1660	24	0	-	-	-
Pnt#2 measured pose relative to sensor	1533	707	0	1688	24,8	0	-	-	-
Pnt#3 real pose relative to sensor	1508	1175	0	1912	37,9	0	-	-	-
Pnt#3 measured pose relative to sensor	1473	1130	0	1857	37,5	0	-	-	-
Pnt#4 real pose relative to sensor	1508	1175	385	1950	37,9	11,4	-	-	-
Pnt#4 measured pose relative to sensor	1494	1131	400	1916	37,1	12,1	-	-	-

Table 62 –visual data collected for experiment#3move#4



Distance Image



Constructed Point Cloud

Table 62 (continued)

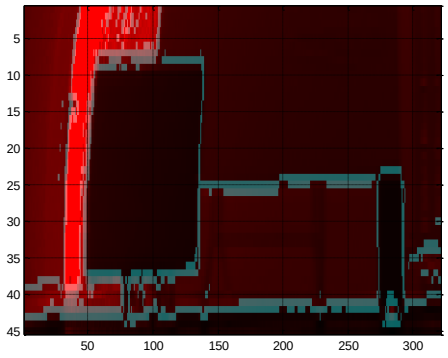
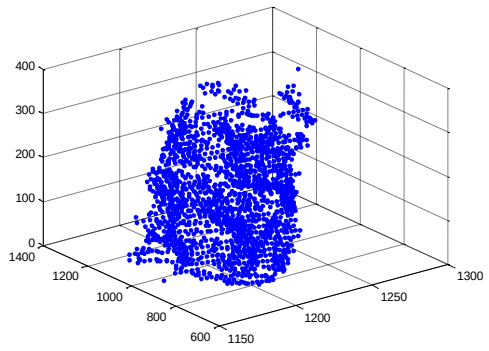
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 63 – background truth and measured parameters for experiment#3move#4

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	873	-600	115	-	-	-	-1.1	-	-
Lm#1 real pose relative to sensor	1209	924	193	1534	37,4	7,2	1,1	420	544
Lm#1 measured pose relative to sensor	1200	910	186	1517	37,2	7	2,6	432	461
Pnt#1 real pose relative to sensor	1214	674	385	1441	29	15,5	-	-	-
Pnt#1 measured pose relative to sensor	1220	667	429	1455	28,7	17,1	-	-	-
Pnt#2 real pose relative to sensor	1214	674	0	1389	29	0	-	-	-
Pnt#2 measured pose relative to sensor	1215	691	31	1398	29,6	1,3	-	-	-
Pnt#3 real pose relative to sensor	1204	1174	0	1682	44,3	0	-	-	-
Pnt#3 measured pose relative to sensor	1172	1157	18	1647	44,6	0,6	-	-	-
Pnt#4 real pose relative to sensor	1204	1174	385	1725	44,3	12,9	-	-	-
Pnt#4 measured pose relative to sensor	1184	1116	482	1697	43,3	16,5	-	-	-

Table 64 –visual data collected for experiment#3move#5

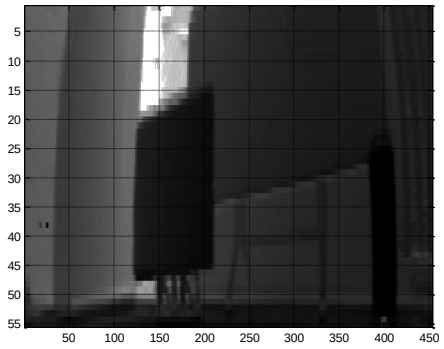
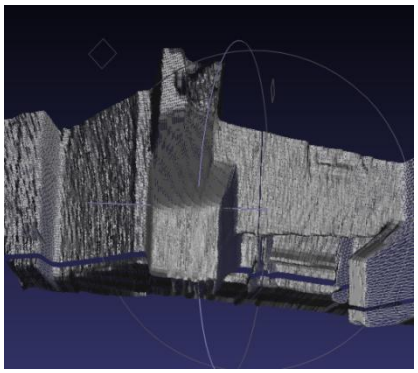
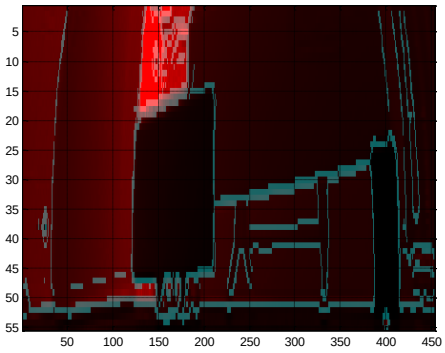
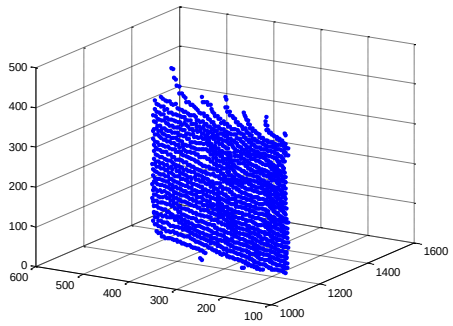
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature

Table 65 – background truth and measured parameters for experiment#3move#5

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	1173	-600	115	-	-	-	30.2	-	-
Lm#1 real pose relative to sensor	1254	312	193	1306	14	8,5	-30,2	420	544
Lm#1 measured pose relative to sensor	1205	299	200	1257	14	9,2	-29,1	380	416
Pnt#1 real pose relative to sensor	1128	96	385	1196	4,9	18,8	-	-	-
Pnt#1 measured pose relative to sensor	1115	117	401	1191	6	19,7	-	-	-
Pnt#2 real pose relative to sensor	1128	96	0	1132	4,9	0	-	-	-
Pnt#2 measured pose relative to sensor	1118	118	37	1125	6	1,9	-	-	-

Table 65 (continued)

Pnt#3 real pose relative to sensor	1379	528	0	1477	20,9	0	-	-	-
Pnt#3 measured pose relative to sensor	1296	507	15	1392	21,4	0,6	-	-	-
Pnt#4 real pose relative to sensor	1379	528	385	1526	20,9	14,6	-	-	-
Pnt#4 measured pose relative to sensor	1301	475	411	1445	20,1	16,5	-	-	-

Table 66 – map constructed at each step with plane landmarks for experiment#3

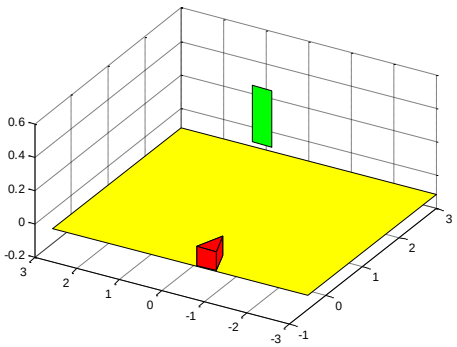
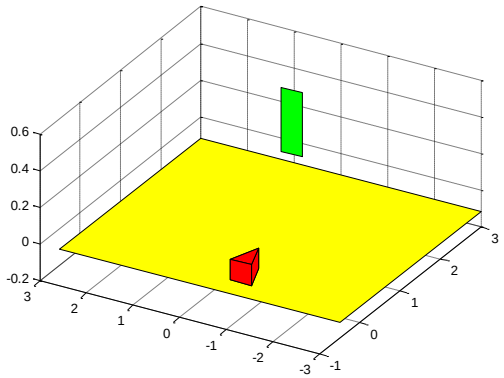
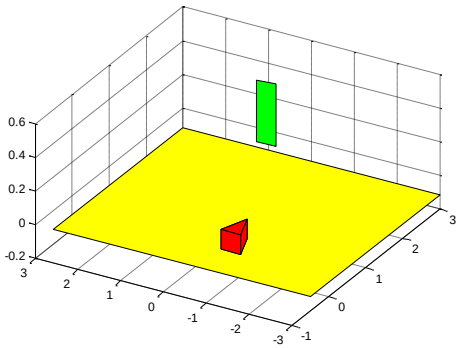
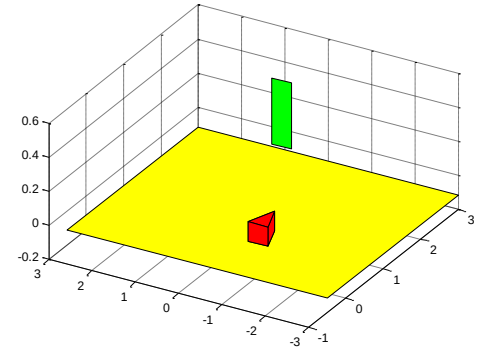
	
time = 1	time = 2
	
time = 3	time = 4

Table 66 (continued)

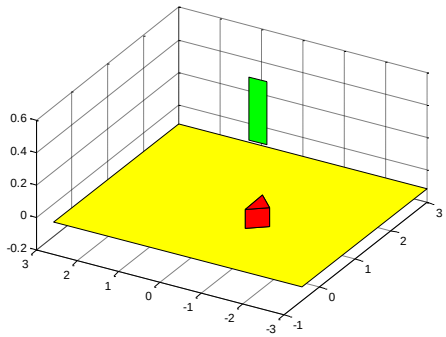
	
time = 5	

Table 67 – extracted data with background truth for plane landmarks experiment#3move#1

time = 1	x	y	z	θ	height	width
robot real	-27	-600	0(115)	-0.75	-	-
robot belief	-27	-600	0	-0.75	-	-
lm#1 real	2100	300	193(308)	0	385	500
lm#1 belief	2087	301	176	0.09	336	457

Table 68 – extracted data with background truth for plane landmarks experiment#3move#2

time = 2	x	y	z	θ	height	width
robot real	273	-600	0(115)	-0.94	-	-
robot belief	273	-604	0	-0.079	-	-
lm#1 real	2100	300	193(308)	0	385	500
lm#1 belief	2146	322	184	1.38	352	453

Table 69 – extracted data with background truth for plane landmarks experiment#3move#3

time = 3	x	y	z	θ	height	width
robot real	573	-600	0(115)	-0.94	-	-
robot belief	573	-606	0	-0.54	-	-
lm#1 real	2100	300	193(308)	0	385	500
lm#1 belief	2156	330	179	2.05	367	448

Table 70 – extracted data with background truth for plane landmarks experiment#3move#4

time = 4	x	y	z	θ	height	width
robot real	873	-600	0(115)	-1.13	-	-
robot belief	873	-608	0	-0.61	-	-
lm#1 real	2100	300	193(308)	0	385	500
lm#1 belief	2115	313	181	2.07	387	452

Table 71 – extracted data with background truth for plane landmarks experiment#3move#5

time = 5	x	y	z	θ	height	width
robot real	1173	-600	0(115)	30.18	-	-
robot belief	1173	-598	0	30.42	-	-
lm#1 real	2100	300	193(308)	0	385	500
lm#1 belief	2090	301	187	1.89	385	442

Table 72 – map constructed at each step with point landmarks for experiment#3

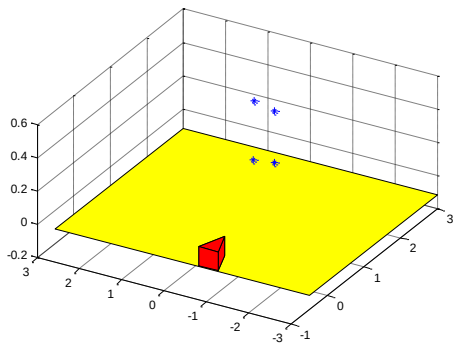
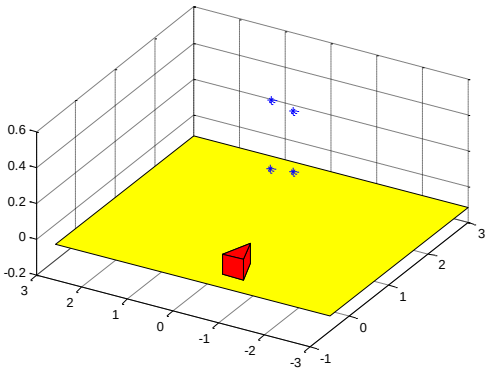
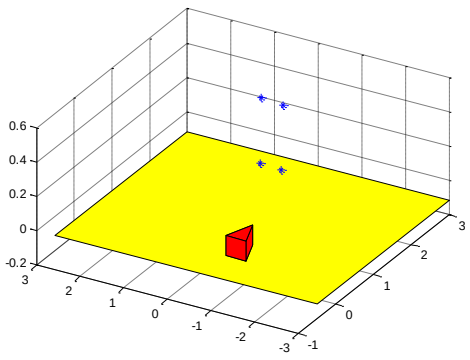
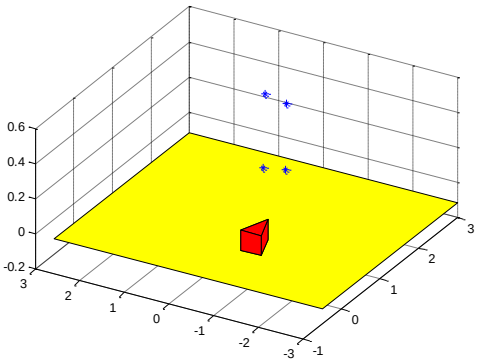
	
time = 1	time = 2
	
time = 3	time = 4

Table 72 (continued)

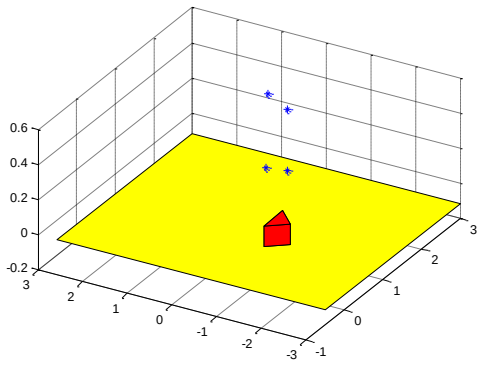
	
time = 5	time = 6

Table 73 – extracted data with background truth for point landmarks experiment#3move#1

time = 1	x	y	z	θ	height	width
robot real	-27	-600	0(115)	-0.75	-	-
robot belief	-27	-600	0	-0.75	-	-
pnt#1 real	2100	50	385(500)	-	-	-
pnt#1 belief	2080	62	345	-	-	-
pnt#2 real	2100	50	0(115)	-	-	-
pnt#2 belief	2107	86	25	-	-	-
pnt#3 real	2100	550	0(115)	-	-	-
pnt#3 belief	2054	530	26	-	-	-
pnt#4 real	2100	550	385(500)	-	-	-
pnt#4 belief	2081	527	374	-	-	-

Table 74 – extracted data with background truth for point landmarks experiment#3move#2

time = 2	x	y	z	θ	height	width
robot real	273	-600	0(115)	-0.94	-	-
robot belief	273	-604	0	-0.70	-	-
pnt#1 real	2100	50	385(500)	-	-	-
pnt#1 belief	2135	74	367	-	-	-
pnt#2 real	2100	50	0(115)	-	-	-
pnt#2 belief	2155	96	23	-	-	-
pnt#3 real	2100	550	0(115)	-	-	-
pnt#3 belief	2081	541	25	-	-	-
pnt#4 real	2100	550	385(500)	-	-	-
pnt#4 belief	2116	540	398	-	-	-

Table 75 – extracted data with background truth for point landmarks experiment#3move#3

time = 3	x	y	z	θ	height	width
robot real	573	-600	0(115)	-0.94	-	-
robot belief	573	-606	0	-0.49	-	-
pnt#1 real	2100	50	385(500)	-	-	-
pnt#1 belief	2165	84	380	-	-	-
pnt#2 real	2100	50	0(115)	-	-	-
pnt#2 belief	2108	83	13	-	-	-
pnt#3 real	2100	550	0(115)	-	-	-
pnt#3 belief	2114	560	16	-	-	-
pnt#4 real	2100	550	385(500)	-	-	-
pnt#4 belief	2108	537	401	-	-	-

Table 76 – extracted data with background truth for point landmarks experiment#3move#4

time = 4	x	y	z	θ	height	width
robot real	873	-600	0(115)	-1.13	-	-
robot belief	873	-608	0	-0.48	-	-
pnt#1 real	2100	50	385(500)	-	-	-
pnt#1 belief	2133	72	396	-	-	-
pnt#2 real	2100	50	0(115)	-	-	-
pnt#2 belief	2113	86	20	-	-	-
pnt#3 real	2100	550	0(115)	-	-	-
pnt#3 belief	2041	518	16	-	-	-
pnt#4 real	2100	550	385(500)	-	-	-
pnt#4 belief	2092	526	428	-	-	-

Table 77 – extracted data with background truth for point landmarks experiment#3move#5

time = 5	x	y	z	θ	height	width
robot real	1173	-600	0(115)	30.18	-	-
robot belief	1173	-599	0	30.28	-	-
pnt#1 real	2100	50	385(500)	-	-	-
pnt#1 belief	2057	44	383	-	-	-
pnt#2 real	2100	50	0(115)	-	-	-
pnt#2 belief	2098	80	26	-	-	-
pnt#3 real	2100	550	0(115)	-	-	-
pnt#3 belief	2059	530	16	-	-	-
pnt#4 real	2100	550	385(500)	-	-	-
pnt#4 belief	2104	533	431	-	-	-

Experiment#4

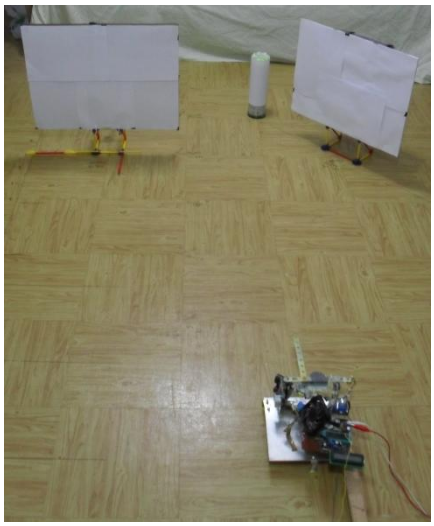


Figure 110 – objects of experiment#4 and sensor system (photo taken at move#6)

In this experiment the objects shown in Figure 110 are placed at x_1=2700mm, y_1=300mm, α_1=2° and x_2=2700mm, y_2=-600mm, α_2=-47.5°. IRSCAN, moves through x axis and collected data from following poses::

step#1: x=-27mm, y=-300mm, α= -1.3°; step#2: x=273mm, y=-300mm, α= -0.94°; step#3: x=573mm, y=-300mm, α= -0.75°; step#4: x=873mm, y=-300mm, α= -0.94°; step#5: x=1173mm, y=-300mm, α= -1.3°; step#5: x=1473mm, y=-300mm, α= -1.13.

Table 78 –visual data collected for experiment#4move#1

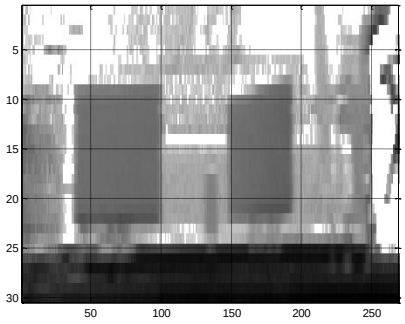
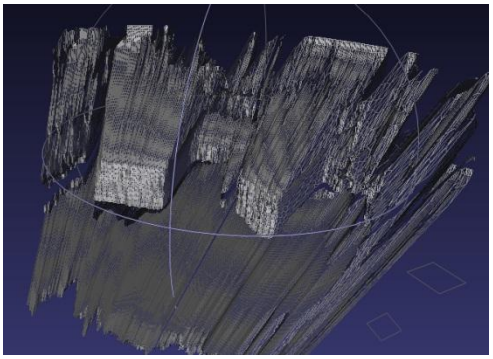
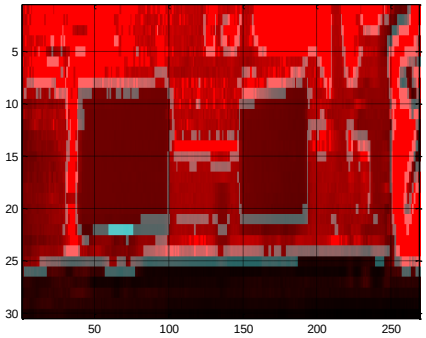
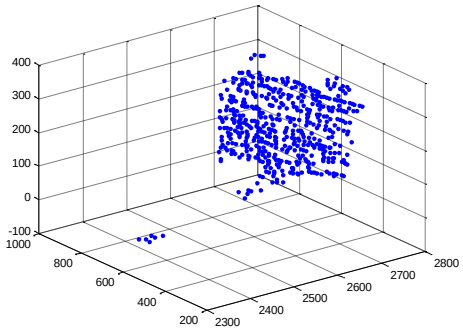
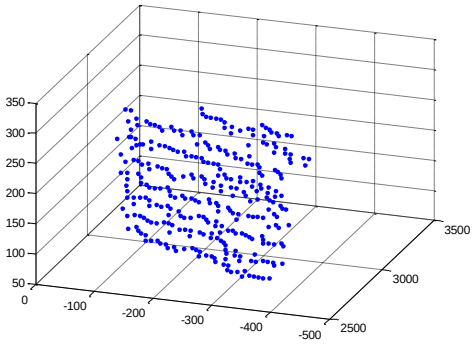
	
Distance Image	Constructed Point Cloud
	
Edge Image - Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 79 – background truth and measured parameters for experiment#4move#1

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	-27	-300	115	-	-	-	-1.31	-	-
Lm#1 real pose relative to sensor	2713	662	210	2800	13,7	4,3	3,3	420	545
Lm#1 measured pose relative to sensor	2688	655	179	2773	13,7	3,7	3,9	326	495
Lm#2 real pose relative to sensor	2733	-237	193	2750	-5	4	-46,2	385	500
Lm#2 measured pose relative to sensor	2716	-258	197	2735	-5,4	4,1	-50,1	277	389
Pnt#1 real pose relative to sensor	2728	393	420	2788	8,2	8,7	-	-	-
Pnt#1 measured pose relative to sensor	2723	404	368	2778	8,4	7,6	-	-	-
Pnt#2 real pose relative to sensor	2728	393	0	2756	8,2	0	-	-	-
Pnt#2 measured pose relative to sensor	2737	406	61	2768	8,4	1,3	-	-	-
Pnt#3 real pose relative to sensor	2676	933	0	2834	19,2	0	-	-	-
Pnt#3 measured pose relative to sensor	2653	900	0	2801	18,8	0	-	-	-
Pnt#4 real pose relative to sensor	2676	933	420	2865	19,2	8,4	-	-	-
Pnt#4 measured pose relative to sensor	2662	884	344	2826	18,4	7	-	-	-
Pnt#5 real pose relative to sensor	2553	-411	385	2614	-9,1	8,5	-	-	-
Pnt#5 measured pose relative to sensor	2611	-422	354	2669	-9,2	7,6	-	-	-
Pnt#6 real pose relative to sensor	2553	-411	0	2586	-9,1	0	-	-	-
Pnt#6 measured pose relative to sensor	2586	-392	58	2616	-8,6	1,3	-	-	-
Pnt#7 real pose relative to sensor	2914	-64	0	2914	-1,3	0	-	-	-
Pnt#7 measured pose relative to sensor	2824	-74	63	2826	-1,5	1,3	-	-	-
Pnt#8 real pose relative to sensor	2914	-64	385	2940	-1,3	7,5	-	-	-
Pnt#8 measured pose relative to sensor	2828	-111	315	2848	-2,3	6,4	-	-	-

Table 80 –visual data collected for experiment#4move#2

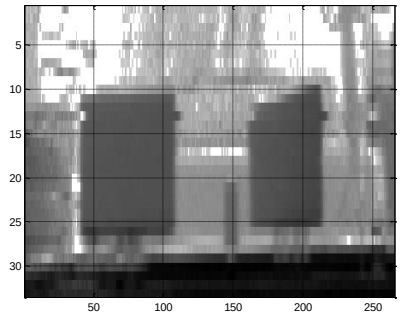
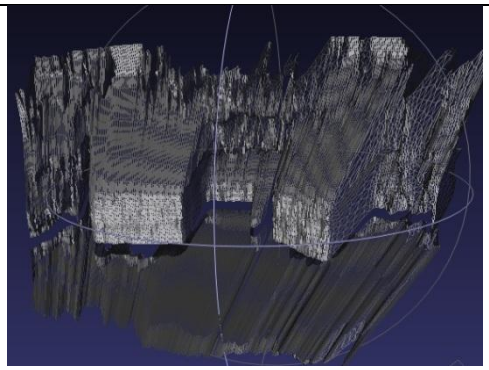
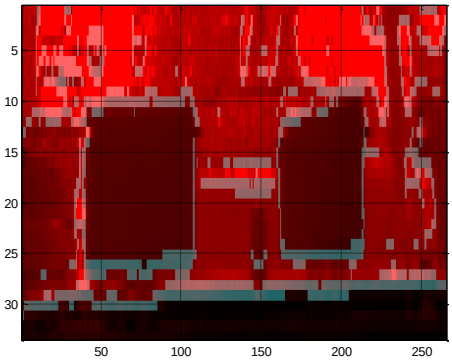
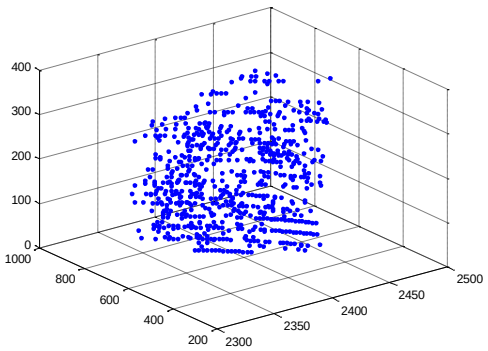
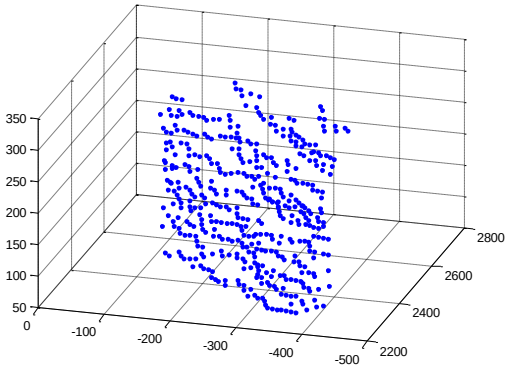
	
Distance Image	Constructed Point Cloud
	
Edge Image	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 81 – background truth and measured parameters for experiment#4move#2

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	273	-300	115	-	-	-	-0.94	-	-
Lm#1 real pose relative to sensor	2417	640	210	2509	14,8	4,8	2,9	420	545
Lm#1 measured pose relative to sensor	2398	644	173	2489	15	4	3,3	332	514
Lm#2 real pose relative to sensor	2432	-260	193	2453	-6,1	4,5	-46,6	385	500
Lm#2 measured pose relative to sensor	2408	-271	182	2430	-6,4	4,3	-46,3	298	428
Pnt#1 real pose relative to sensor	2431	370	420	2494	8,7	9,7	-	-	-
Pnt#1 measured pose relative to sensor	2403	381	353	2459	9	8,3	-	-	-
Pnt#2 real pose relative to sensor	2431	370	0	2459	8,7	0	-	-	-
Pnt#2 measured pose relative to sensor	2428	385	27	2459	9	0,6	-	-	-
Pnt#3 real pose relative to sensor	2382	911	0	2550	20,9	0	-	-	-
Pnt#3 measured pose relative to sensor	2372	911	0	2541	21	0	-	-	-
Pnt#4 real pose relative to sensor	2382	911	420	2585	20,9	9,4	-	-	-
Pnt#4 measured pose relative to sensor	2351	876	336	2531	20,4	7,6	-	-	-
Pnt#5 real pose relative to sensor	2250	-432	385	2323	-10,9	9,5	-	-	-
Pnt#5 measured pose relative to sensor	2283	-439	364	2353	-10,9	8,9	-	-	-
Pnt#6 real pose relative to sensor	2250	-432	0	2291	-10,9	0	-	-	-
Pnt#6 measured pose relative to sensor	2258	-426	51	2298	-10,7	1,3	-	-	-
Pnt#7 real pose relative to sensor	2613	-88	0	2615	-1,9	0	-	-	-
Pnt#7 measured pose relative to sensor	2534	-100	28	2536	-2,3	0,6	-	-	-
Pnt#8 real pose relative to sensor	2613	-88	385	2643	-1,9	8,4	-	-	-
Pnt#8 measured pose relative to sensor	2529	-91	310	2549	-2,1	7	-	-	-

Table 82 – visual data collected for experiment#4move#3

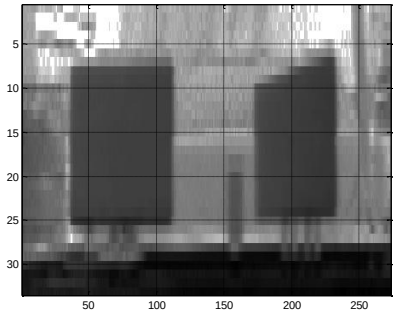
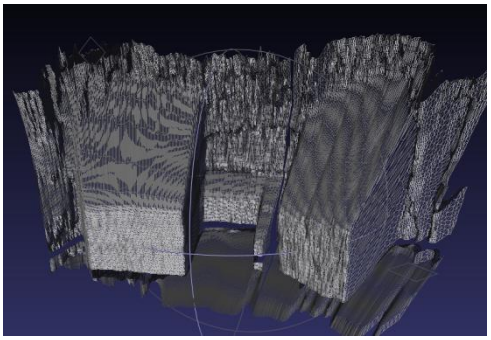
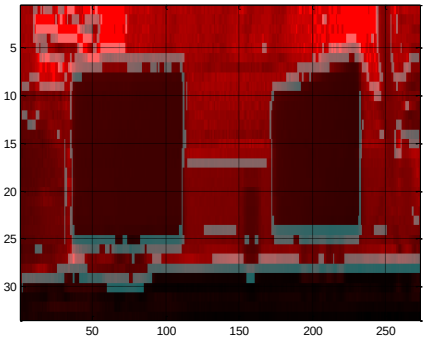
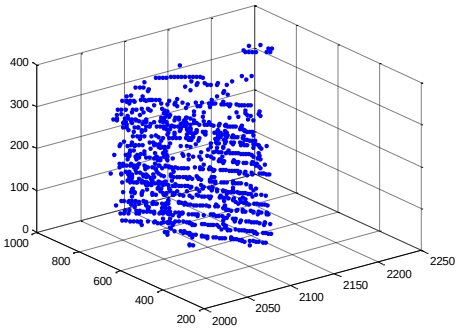
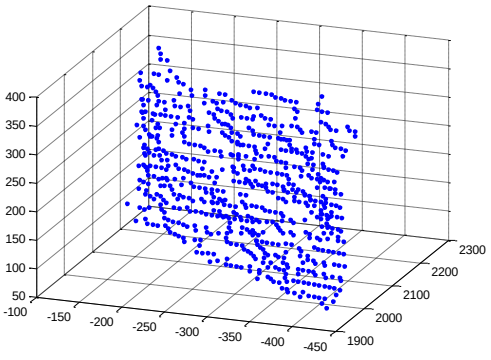
	
Distance Image	Constructed Point Cloud
	
Edge Image - Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 83 – Background truth and measured parameters for experiment#4move#3

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	573	-300	115	-	-	-	-0.75	-	-
Lm#1 real pose relative to sensor	2119	628	210	2220	16,5	5,4	2,8	420	545
Lm#1 measured pose relative to sensor	2086	626	199	2187	16,7	5,2	4,2	356	510
Lm#2 real pose relative to sensor	2131	-272	193	2157	-7,3	5,1	-46,8	385	500
Lm#2 measured pose relative to sensor	2086	-283	204	2115	-7,7	5,5	-50,2	319	429
Pnt#1 real pose relative to sensor	2132	358	420	2202	9,5	11	-	-	-
Pnt#1 measured pose relative to sensor	2091	366	380	2157	9,9	10,2	-	-	-
Pnt#2 real pose relative to sensor	2132	358	0	2162	9,5	0	-	-	-
Pnt#2 measured pose relative to sensor	2118	385	24	2153	10,3	0,6	-	-	-
Pnt#3 real pose relative to sensor	2085	899	0	2271	23,3	0	-	-	-
Pnt#3 measured pose relative to sensor	2062	878	25	2241	23,1	0,6	-	-	-
Pnt#4 real pose relative to sensor	2085	899	420	2309	23,3	10,5	-	-	-
Pnt#4 measured pose relative to sensor	2072	890	378	2287	23,3	9,5	-	-	-
Pnt#5 real pose relative to sensor	1949	-443	385	2035	-12,8	10,9	-	-	-
Pnt#5 measured pose relative to sensor	1970	-446	385	2056	-12,8	10,8	-	-	-
Pnt#6 real pose relative to sensor	1949	-443	0	1998	-12,8	0	-	-	-
Pnt#6 measured pose relative to sensor	1978	-441	45	2027	-12,6	1,3	-	-	-
Pnt#7 real pose relative to sensor	2313	-101	0	2315	-2,5	0	-	-	-
Pnt#7 measured pose relative to sensor	2233	-95	50	2236	-2,4	1,3	-	-	-
Pnt#8 real pose relative to sensor	2313	-101	385	2347	-2,5	9,4	-	-	-
Pnt#8 measured pose relative to sensor	2213	-94	346	2241	-2,4	8,9	-	-	-

Table 84 – Visual data collected for experiment#4move#4

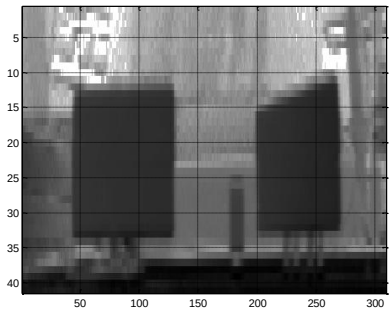
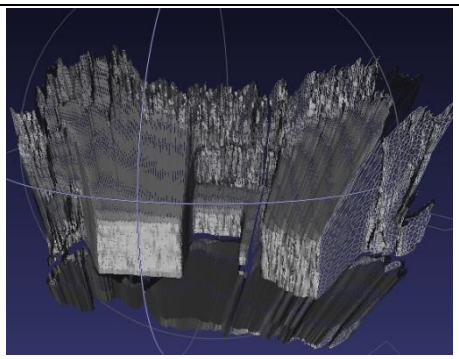
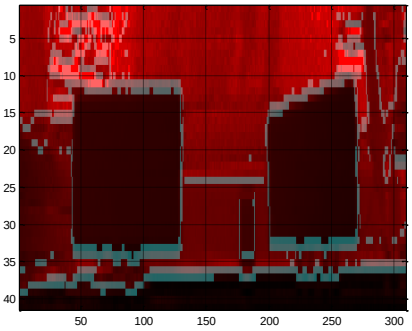
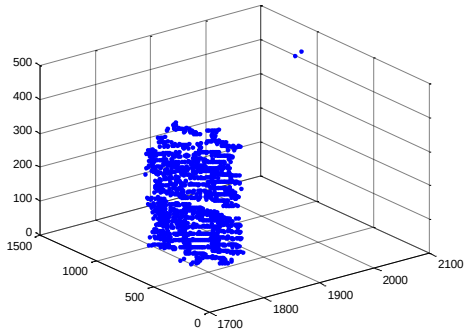
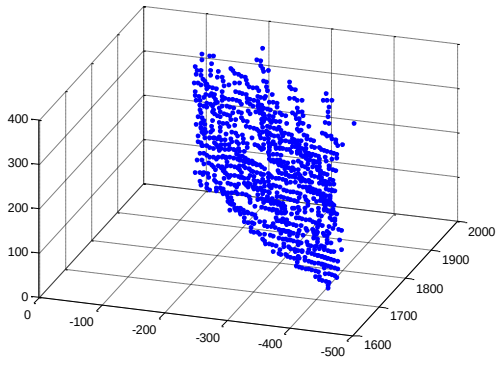
	
Distance Image	Constructed Point Cloud
	
Edge Image - Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 85 – background truth and measured parameters for experiment#4move#4

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	873	-300	115	-	-	-	-0.94	-	-
Lm#1 real pose relative to sensor	1817	630	210	1934	19,1	6,2	2,9	420	545
Lm#1 measured pose relative to sensor	1801	625	200	1917	19,1	6	5	403	524
Lm#2 real pose relative to sensor	1832	-270	193	1861	-8,4	5,9	-46,6	385	500
Lm#2 measured pose relative to sensor	1802	-289	206	1836	-9,1	6,4	-49,7	340	438
Pnt#1 real pose relative to sensor	1831	360	420	1913	11,1	12,7	-	-	-
Pnt#1 measured pose relative to sensor	1812	361	395	1890	11,3	12,1	-	-	-
Pnt#2 real pose relative to sensor	1831	360	0	1866	11,1	0	-	-	-
Pnt#2 measured pose relative to sensor	1829	370	21	1866	11,4	0,6	-	-	-
Pnt#3 real pose relative to sensor	1782	901	0	1997	26,8	0	-	-	-
Pnt#3 measured pose relative to sensor	1765	878	22	1971	26,4	0,6	-	-	-
Pnt#4 real pose relative to sensor	1782	901	420	2041	26,8	11,9	-	-	-
Pnt#4 measured pose relative to sensor	1796	893	452	2056	26,4	12,7	-	-	-
Pnt#5 real pose relative to sensor	1650	-442	385	1751	-15	12,7	-	-	-
Pnt#5 measured pose relative to sensor	1687	-440	413	1792	-14,6	13,3	-	-	-
Pnt#6 real pose relative to sensor	1650	-442	0	1708	-15	0	-	-	-
Pnt#6 measured pose relative to sensor	1677	-449	38	1736	-15	1,3	-	-	-
Pnt#7 real pose relative to sensor	2013	-98	0	2016	-2,8	0	-	-	-
Pnt#7 measured pose relative to sensor	1941	-95	43	1944	-2,8	1,3	-	-	-
Pnt#8 real pose relative to sensor	2013	-98	385	2052	-2,8	10,8	-	-	-
Pnt#8 measured pose relative to sensor	1934	-89	347	1967	-2,6	10,2	-	-	-

Table 86 –visual data collected for experiment#4move#5

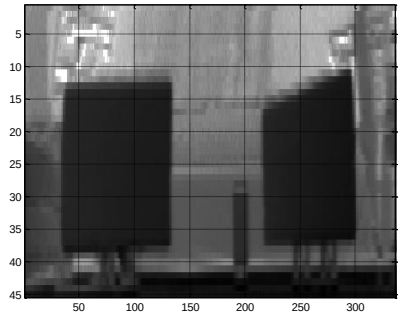
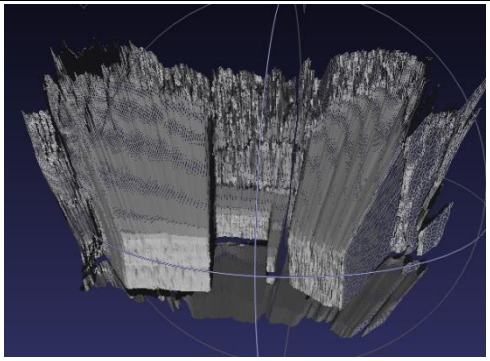
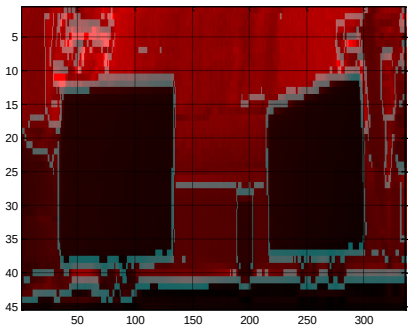
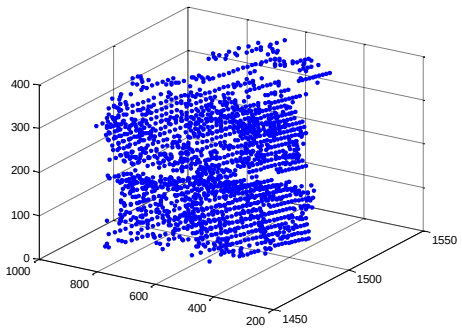
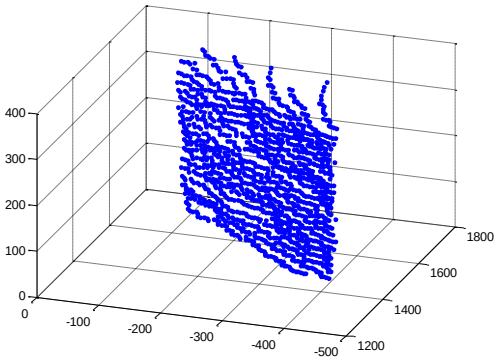
	
Distance Image	Constructed Point Cloud
	
Edge Image - Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 87 – background truth and measured parameters for experiment#4move#5

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	1173	-300	115	-	-	-	-1.31	-	-
Lm#1 real pose relative to sensor	1513	635	210	1654	22,8	7,3	3,3	420	545
Lm#1 measured pose relative to sensor	1495	629	191	1633	22,8	6,7	2	405	515
Lm#2 real pose relative to sensor	1533	-265	193	1568	-9,8	7,1	-46,2	385	500
Lm#2 measured pose relative to sensor	1515	-274	196	1552	-10,3	7,3	-47,5	348	446
Pnt#1 real pose relative to sensor	1529	365	420	1627	13,4	15	-	-	-
Pnt#1 measured pose relative to sensor	1506	362	404	1601	13,5	14,6	-	-	-
Pnt#2 real pose relative to sensor	1529	365	0	1572	13,4	0	-	-	-
Pnt#2 measured pose relative to sensor	1511	368	17	1555	13,7	0,6	-	-	-
Pnt#3 real pose relative to sensor	1476	906	0	1732	31,5	0	-	-	-
Pnt#3 measured pose relative to sensor	1471	888	0	1719	31,1	0	-	-	-
Pnt#4 real pose relative to sensor	1476	906	420	1782	31,5	13,6	-	-	-
Pnt#4 measured pose relative to sensor	1460	869	423	1751	30,8	14	-	-	-
Pnt#5 real pose relative to sensor	1353	-438	385	1473	-17,9	15,1	-	-	-
Pnt#5 measured pose relative to sensor	1376	-417	392	1490	-16,9	15,2	-	-	-
Pnt#6 real pose relative to sensor	1353	-438	0	1422	-17,9	0	-	-	-
Pnt#6 measured pose relative to sensor	1378	-433	32	1445	-17,4	1,3	-	-	-
Pnt#7 real pose relative to sensor	1714	-92	0	1716	-3,1	0	-	-	-
Pnt#7 measured pose relative to sensor	1661	-87	18	1663	-3	0,6	-	-	-
Pnt#8 real pose relative to sensor	1714	-92	385	1759	-3,1	12,6	-	-	-
Pnt#8 measured pose relative to sensor	1659	-76	355	1699	-2,6	12,1	-	-	-

Table 88 –visual data collected for experiment#4move#6

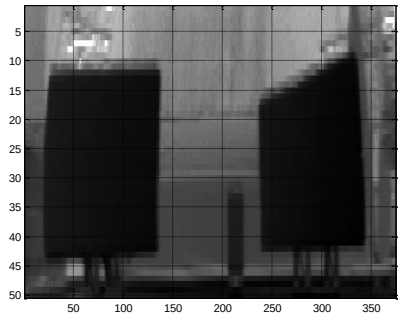
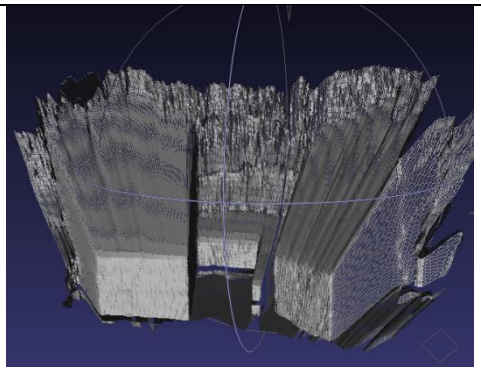
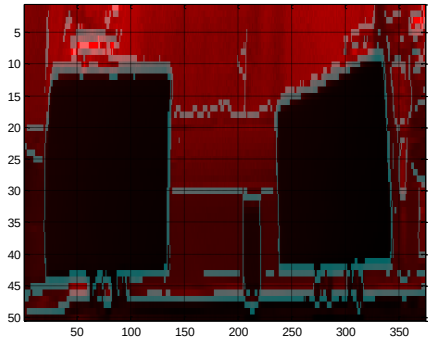
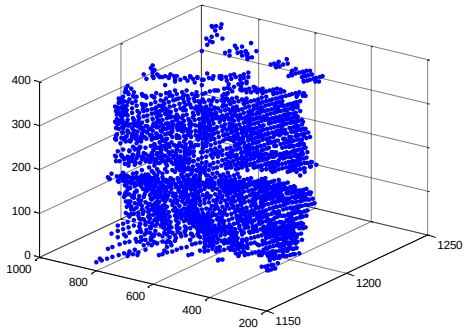
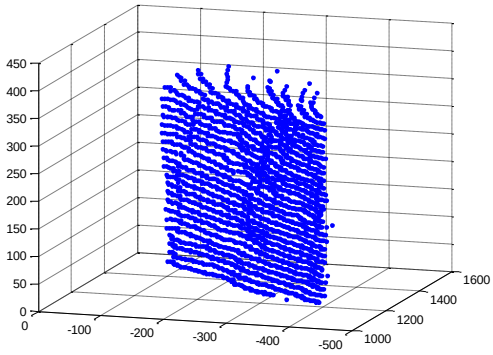
	
Distance Image	Constructed Point Cloud
	
Edge Image - Edges shown brighter are marked by supervisor	Point Cloud Of Depth Segmented Feature#1
	
Point Cloud Of Depth Segmented Feature#2	

Table 89 – background truth and measured parameters for experiment#4move#6

	x	y	z	R	α	β	θ	h	w
Robot Real Pose	1473	-300	115	-	-	-	-1.13	-	-
Lm#1 real pose relative to sensor	1215	624	210	1382	27,2	8,7	3,1	420	545
Lm#1 measured pose relative to sensor	1191	610	190	1352	27,1	8,1	2,3	446	497
Lm#2 real pose relative to sensor	1233	-276	193	1278	-12,6	8,7	-46,4	385	500
Lm#2 measured pose relative to sensor	1214	-286	196	1263	-13,2	9	-45,6	371	453
Pnt#1 real pose relative to sensor	1230	354	420	1347	16,1	18,2	-	-	-
Pnt#1 measured pose relative to sensor	1200	347	416	1317	16,1	18,4	-	-	-
Pnt#2 real pose relative to sensor	1230	354	0	1280	16,1	0	-	-	-
Pnt#2 measured pose relative to sensor	1197	363	0	1251	16,9	0	-	-	-
Pnt#3 real pose relative to sensor	1179	895	0	1480	37,2	0	-	-	-
Pnt#3 measured pose relative to sensor	1162	867	-16	1450	36,8	-0,6	-	-	-
Pnt#4 real pose relative to sensor	1179	895	420	1539	37,2	15,8	-	-	-
Pnt#4 measured pose relative to sensor	1160	831	458	1498	35,6	17,8	-	-	-
Pnt#5 real pose relative to sensor	1052	-448	385	1206	-23,1	18,6	-	-	-
Pnt#5 measured pose relative to sensor	1064	-413	408	1212	-21,2	19,7	-	-	-
Pnt#6 real pose relative to sensor	1052	-448	0	1143	-23,1	0	-	-	-
Pnt#6 measured pose relative to sensor	1064	-441	26	1152	-22,5	1,3	-	-	-
Pnt#7 real pose relative to sensor	1414	-103	0	1417	-4,2	0	-	-	-
Pnt#7 measured pose relative to sensor	1384	-95	15	1388	-3,9	0,6	-	-	-
Pnt#8 real pose relative to sensor	1414	-103	385	1469	-4,2	15,2	-	-	-
Pnt#8 measured pose relative to sensor	1366	-98	373	1419	-4,1	15,2	-	-	-

Table 90 – map constructed at each step with plane landmarks for experiment#4

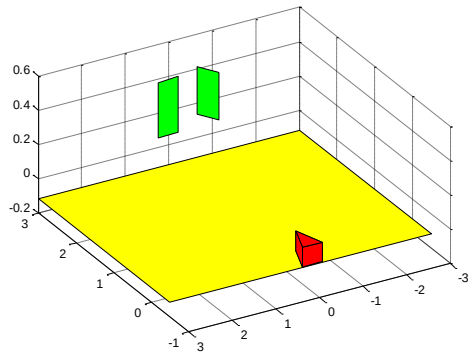
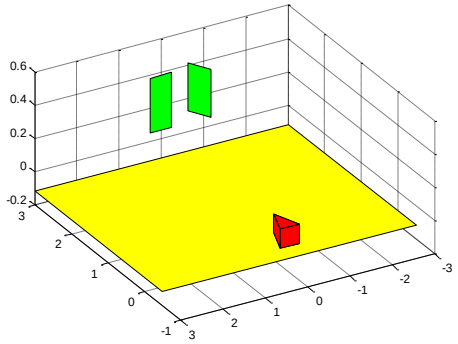
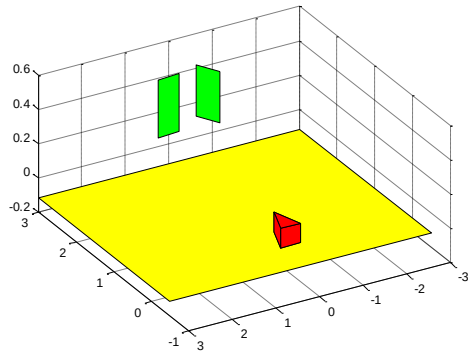
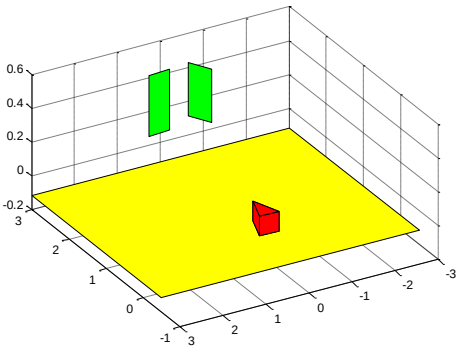
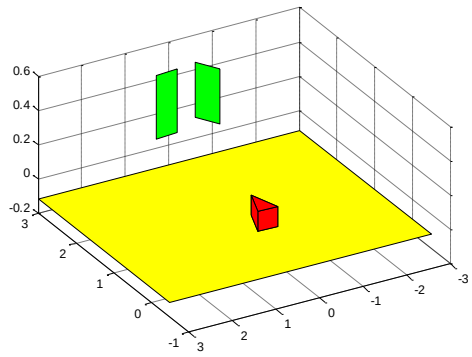
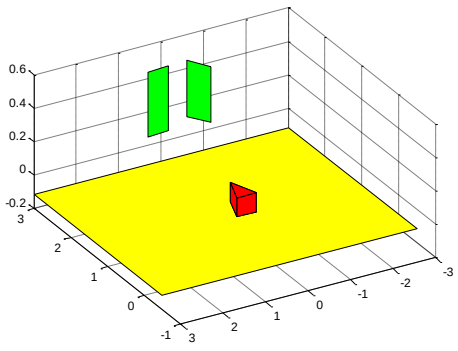
	
time = 1	time = 2
	
time = 3	time = 4
	
time = 5	time = 6

Table 91 – extracted data with background truth for plane landmarks experiment#4move#1

time = 1	x	y	z	θ	height	width
robot real	-27	-300	0(115)	-1.31	-	-
robot belief	-27	-300	0	-1.31	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2675	293	179	2.6	326	495
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2682	-620	198	-51.36	277	389

Table 92 – extracted data with background truth for plane landmarks experiment#4move#2

time = 2	x	y	z	θ	height	width
robot real	273	-300	0(115)	-0.94	-	-
robot belief	273	-306	0	-1.02	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2678	294	175	2.43	329	505
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2679	-620	189	-49.06	288	410

Table 93 – extracted data with background truth for plane landmarks experiment#4move#3

time = 3	x	y	z	θ	height	width
robot real	573	-300	0(115)	-0.75	-	-
robot belief	573	-310	0	-0.67	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2668	292	184	2.90	339	507
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2661	-618	194	-49.82	300	417

Table 94 – extracted data with background truth for plane landmarks experiment#4move#4

time = 4	x	y	Z	θ	height	width
robot real	873	-300	0(115)	-0.94	-	-
robot belief	873	-314	0	-0.63	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2686	294	191	3.4	358	512
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2654	-619	198	-50.01	312	424

Table 95 – extracted data with background truth for plane landmarks experiment#4move#5

time = 5	x	y	z	θ	height	width
robot real	1173	-300	0(115)	-1.30	-	-
robot belief	1173	-317	0	-0.92	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2707	303	194	2.66	371	513
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2687	-619	199	-49.43	321	429

Table 96 – extracted data with background truth for plane landmarks experiment#4move#6

time = 6	x	y	z	θ	height	width
robot real	1473	-300	0(115)	-1.13	-	-
robot belief	1472	-317	0	-0.49	-	-
lm#1 real	2700	300	210(325)	2.00	420	545
lm#1 belief	2717	311	195	2.49	389	509
lm#2 real	2700	-600	193(308)	-47.5	385	500
lm#2 belief	2684	-613	198	-48.29	333	435

Table 97 – map constructed at each step with point landmarks for experiment#4

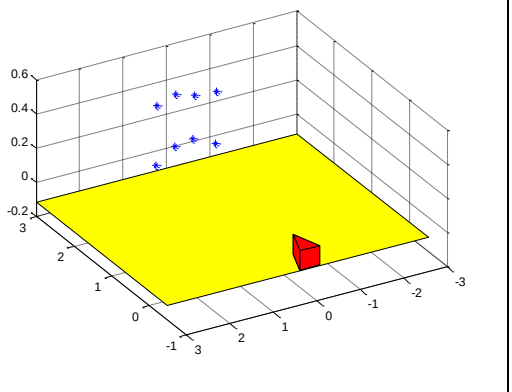
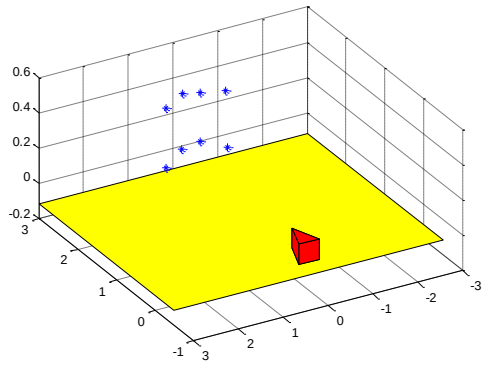
	
time = 1	time = 2

Table 97 (continued)

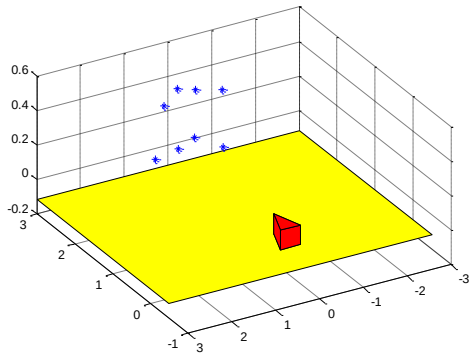
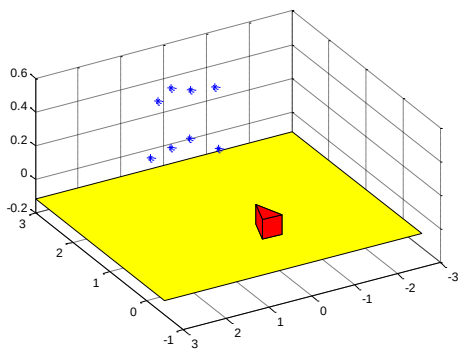
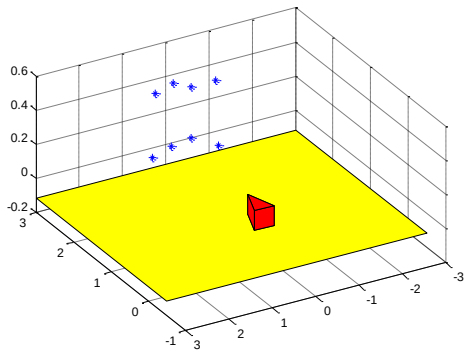
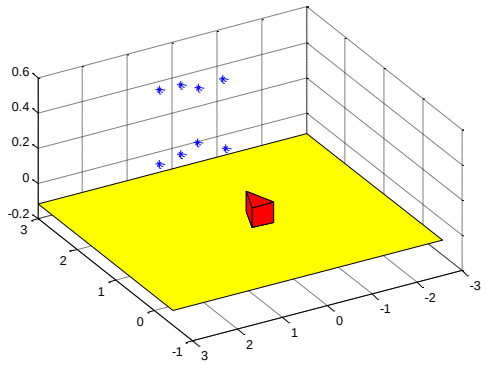
	
time = 3	time = 4
	
time = 5	time = 6

Table 98 – extracted data with background truth for point landmarks experiment#4move#1

time = 1	x	y	Z	θ	height	width
robot real	-27	-300	0(115)	-1.31	-	-
robot belief	-27	-300	0	-1.31	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2705	41	368	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2719	43	61	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-
pnt#3 belief	2646	540	0	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2654	523	344	-	-	-
pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2574	-782	354	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-

Table 98 (continued)

pnt#6 belief	2549	-751	58	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2795	-439	62	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2798	-476	315	-	-	-

Table 99 – extracted data with background truth for point landmarks experiment#4move#6

time = 2	x	y	Z	θ	height	width
robot real	273	-300	0(115)	-0.94	-	-
robot belief	273	-306	0	-0.97	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2710	41	362	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2729	43	43	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-
pnt#3 belief	2566	523	0	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2593	510	332	-	-	-
pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2527	-775	355	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-
pnt#6 belief	2443	-744	52	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2797	-443	43	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2830	-457	316	-	-	-

Table 100 – extracted data with background truth for point landmarks experiment#4move#3

time = 3	x	y	Z	θ	height	width
robot real	573	-300	0(115)	-0.75	-	-
robot belief	573	-313	0	-1.02	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2725	29	375	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2693	30	35	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-
pnt#3 belief	2691	551	10	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2535	484	337	-	-	-

Table 100 (continued)

pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2473	-776	358	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-
pnt#6 belief	2393	-748	46	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2807	-452	46	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2798	-460	326	-	-	-

Table 101 – extracted data with background truth for point landmarks experiment#4move#4

time = 4	x	y	Z	θ	height	width
robot real	873	-300	0(115)	-0.94	-	-
robot belief	873	-319	0	-1.03	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2758	29	389	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2764	36	31	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-
pnt#3 belief	2709	556	14	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2585	499	373	-	-	-
pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2519	-785	380	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-
pnt#6 belief	2375	-747	42	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2816	-452	45	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2804	-454	334	-	-	-

Table 102 – extracted data with background truth for point landmarks experiment#4move#5

time = 5	x	y	Z	θ	height	width
robot real	1173	-300	0(115)	-1.30	-	-
robot belief	1172	-321	0	-1.18	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2765	36	401	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2793	46	28	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-

Table 102 (continued)

pnt#3 belief	2704	562	10	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2656	532	396	-	-	-
pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2585	-790	396	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-
pnt#6 belief	2455	-757	39	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2829	-444	37	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2821	-443	342	-	-	-

Table 103 – extracted data with background truth for point landmarks experiment#4move#6

time = 6	x	y	Z	θ	height	width
robot real	1473	-300	0(115)	-1.13	-	-
robot belief	1472	-323	0	-0.68	-	-
pnt#1 real	2710	30	420(535)	-	-	-
pnt#1 belief	2737	34	407	-	-	-
pnt#2 real	2710	30	0(115)	-	-	-
pnt#2 belief	2727	41	19	-	-	-
pnt#3 real	2670	572	0(115)	-	-	-
pnt#3 belief	2682	559	3	-	-	-
pnt#4 real	2670	572	420(535)	-	-	-
pnt#4 belief	2699	553	423	-	-	-
pnt#5 real	2516	-768	385(500)	-	-	-
pnt#5 belief	2605	-787	409	-	-	-
pnt#6 real	2516	-768	0(115)	-	-	-
pnt#6 belief	2490	-760	35	-	-	-
pnt#7 real	2884	-431	0(115)	-	-	-
pnt#7 belief	2852	-438	31	-	-	-
pnt#8 real	2884	-431	385(500)	-	-	-
pnt#8 belief	2814	-436	348	-	-	-