

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**OBJECT DETECTION AND MAPPING USING
LIDAR FOR A MOBILE ROBOT**



by
Başak YILDIZ

August, 2016
İZMİR

OBJECT DETECTION AND MAPPING USING LIDAR FOR A MOBILE ROBOT

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of
Science in Mechatronics Engineering Program**

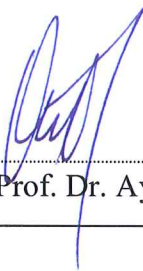


**by
Başak YILDIZ**

**August, 2016
İZMİR**

M.Sc THESIS EXAMINATION RESULT FORM


We have read the thesis entitled “**OBJECT DETECTION AND MAPPING USING LIDAR FOR A MOBILE ROBOT**” completed by **BAŞAK YILDIZ** under supervision of **ASSIST. PROF. DR. AYTAÇ GÖREN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Assist. Prof. Dr. Aytaç GÖREN

Supervisor


Prof. Dr. Zeki KIRAL

(Jury Member)


Assist. Prof. Dr. Özgür BAŞER

(Jury Member)


Prof. Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank to my thesis consultant Assist. Prof. Dr. Aytaç GÖREN helping me in completing my Master Thesis with his guidance during the development of this study. I would also like to thank to my family for standing by me throughout my life.

Başak YILDIZ



OBJECT DETECTION AND MAPPING USING LIDAR FOR A MOBILE ROBOT

ABSTRACT

Autonomous mobile robots should effectively complete tasks and be able to adapt to changing environmental conditions. In addition, mobile robots are required to explore their environment without colliding with any objects and obstacles. For meeting the needs of mobile robots, first step is creating a comprehensive map to model the environment. Then the objects in the environment should be detected in the created map. This study focuses on these two steps that are the essential parts of the automation process for mobile robots. The aims of this research are getting reliable data from LIDAR sensor, LIDAR based map building and object detection in indoor environments.

Keywords: Mapping, LIDAR, object detection

MOBİL ROBOT İÇİN LİDAR KULLANARAK NESNE ALGILAMA VE HARİTALAMA

ÖZ

Otonom mobil robotlar verimli bir şekilde görevlerini tamamlamalı ve deęişen çevre koşullarına uyum sağlayabilmelidir. Ayrıca çevrelerini nesnelere ve engellere çarpmadan keşfetmelidir. Mobil robotların ihtiyaçlarını karşılamak için ilk adım ortamın detaylı bir haritasını çıkarmaktır. Sonrasında ortamdaki nesnelere ve engelleri algılayıp haritada göstermektir. Bu çalışma, mobil robotların otonom olarak çalışması için gereken bu temel iki adıma odaklanmıştır. Bu çalışmanın amaçları LIDAR sensöründen güvenilir veri alınması, iç ortamlarda LIDARa dayalı haritalama ve nesne algılama yapılmasıdır.

Anahtar kelimeler: Haritalama, LIDAR, nesne algılama

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	v
LIST OF FIGURES	ix
LIST OF TABLES	xii
CHAPTER ONE - INTRODUCTION	1
CHAPTER TWO - METHODOLOGY.....	2
2.1 Lidar Technology	2
2.1.1 Lidar Types According to the Operating Principles	2
2.1.2 Applications of LIDAR	4
2.1.2.1 Airborne Laser Scanning	4
2.1.2.2 Terrestrial Laser Scanning	5
2.1.3 LIDAR Applications for Mobile Robots	6
2.2 Mapping.....	7
2.2.1 Robotic Mapping	7
2.2.2 Map Representation	8
2.2.2.1 Metric Map	8
2.2.2 Topographical Map.....	10
2.3 Filtering (Estimation) Theory.....	11
2.3.1 Kalman Filter	11
2.3.1.1 Extended Kalman Filter	14
2.3.1.2 Unscented Kalman Filter	15
2.3.2 Particle Filter	16
2.3.2.1 Rao-Blackwellized Particle Filter.....	19
2.4 Object Detection.....	21
2.4.1 Costmap	22

2.5 Robot Operating System (ROS)	22
CHAPTER THREE - EXPERIMENTAL SETUP	25
CHAPTER FOUR - EXPERIMENTAL WORK.....	33
4.1 Getting Reliable LIDAR Data	33
4.2 Mapping and Experimental Work in ROS Environment	42
4.2.1 Required Fundamental Knowledge for Creating System in ROS	42
4.2.1.1 Coordinate Frames for Mobile Platforms	42
4.2.1.2 Relations between Frames	43
4.2.1.3 The Transformation Library (tf) and tf Transforms	44
4.2.2 Mapping via Hector SLAM Algorithm	44
4.2.2.1 Theory of Hector SLAM Algorithm.....	45
4.2.2.2 Implementation of Hector SLAM and Experimental Work	48
4.2.2.3 Mapping Results with Hector SLAM.....	50
4.2.3 Mapping via Gmapping Algorithm	54
4.2.3.1 Theory of Gmapping Algorithm.....	54
4.2.3.2 Implementation of Gmapping and Experimental Work	60
4.2.3.3 Mapping Results with Gmapping	63
4.2.4 Comparison of Gmapping and Hector SLAM for Mapping.....	66
4.3 Object Detection & Costmaps	66
4.3.1 Theory of Costmap	67
4.3.2 Implementation of Costmap_2d and Experimental Work.....	71
CHAPTER FIVE - CONCLUSION	82
REFERENCES.....	83
APPENDICES	88
APPENDIX-1: Technical Specifications of LMS.....	88
APPENDIX-2: Hector_mapping_default.launch File	93

APPENDIX-3: Gmapping_params.yaml File 95



LIST OF FIGURES

	Page
Figure 2.1 Principles of operating for purpose propagation time measurement	3
Figure 2.2 Airborne laser scanner system	4
Figure 2.3 Terrastrial laser scanning (TLS)	5
Figure 2.4 Representation of occupancy grid map	9
Figure 2.5 Distinction between metric map and topographical map	10
Figure 2.6 Operation of Kalman filter	14
Figure 2.7 The operation of the extended Kalman filter	15
Figure 2.8 The operation of the unscented Kalman filter	16
Figure 2.9 Concept of ROS	23
Figure 3.1 Technical drawing of mobile robot	25
Figure 3.2 Top view of the mobile robot	25
Figure 3.3 Technical drawing of modified mobile robot	26
Figure 3.4 Experimental mobile robot	27
Figure 3.5 LMS100 from LMS1xx family	27
Figure 3.6 Scanning range of LMS100	28
Figure 3.7 Maximum instant area for map building with LMS100	29
Figure 3.8 Configuration of LIDAR	29
Figure 3.9 VirtualBox as a virtual machine	30
Figure 3.10 The area of the study, Automatic Control Laboratory of Mechanical Engineering Department, DEU	31
Figure 3.11 The area of the study, main corridor	31
Figure 3.12 The layout of the study area	32
Figure 4.1 Overview of the process for autonomous mobile robots	33
Figure 4.2 Monitoring laser data from LMS100 LIDAR	34
Figure 4.3 Real area and the observed area	34
Figure 4.4 Coordinate system transformation	35
Figure 4.5 Basic map of the corridor from one data set	36
Figure 4.6 Calculating RMSE	38
Figure 4.7 Root mean square errors (RMSE) of LIDAR	38

Figure 4.8 Root mean square errors (RMSE) of LIDAR with own mean filter.....	39
Figure 4.9 Control scheme of mobile robot	40
Figure 4.10 Calculating system resolution with DC motor	40
Figure 4.11 Scenario for calculating errors.....	41
Figure 4.12 Overview of mapping system with hector_slam	47
Figure 4.13 Original frame scheme of hector_slam.....	48
Figure 4.14 Modified frame scheme of hector_slam for our study.....	49
Figure 4.15 Graph of our hector_slam implementation	50
Figure 4.16 Built map of laboratory corridor.....	51
Figure 4.17 Measurements in the built map.....	52
Figure 4.18 Escape of laser beams while mapping of laboratory corridor	52
Figure 4.19 Built map of laboratory corridor and the main corridor	53
Figure 4.20 Built map of DEU Automatic Control Laboratory and its corridor.....	54
Figure 4.21 Graph of our gmapping implementation	63
Figure 4.22 Early stage of map building process.....	63
Figure 4.23 Built map of laboratory corridor.....	64
Figure 4.24 Built map of the Automatic Control Laboratory	65
Figure 4.25 Built map of the Automatic Control Laboratory and its corridor.....	65
Figure 4.26 Comparison of gmapping and hector slam	66
Figure 4.27 Representing mobile robot in an environment with obstacles.....	67
Figure 4.28 Cell types	69
Figure 4.29 Update algorithm Costmap layers	70
Figure 4.30 Mobile robot and obstacle interaction	71
Figure 4.31 Graph of our costmap implementation	74
Figure 4.32 tf tree of our implementation	74
Figure 4.33 Static layer	75
Figure 4.34 Obstacle layer	75
Figure 4.35 Inflation layer also it can be called master layer	76
Figure 4.36 Costmap while robot is exploring between tables in the laboratory.....	77
Figure 4.37 Dynamic objects and clearing function of costmap.....	78
Figure 4.38 Costmap of Automatic Control Laboratory	78
Figure 4.39 Mobile robot while passing the doorway.....	79

Figure 4.40 Costmap while mobile robot is passing the doorway	79
Figure 4.41 Minimum detectable object	80
Figure 4.42 Minimum detectable object in costmap.....	81



LIST OF TABLES

	Page
Table 3.1 Measures of the mobile robot.....	26
Table 3.2 Functional specifications of LMS100	28
Table 4.1 Errors of the mobile robot.....	41
Table 4.2 Distance traveled errors in the scenario	41



CHAPTER ONE

INTRODUCTION

Nowadays with the help of the technological developments and accumulating knowledge, mobile robots are widely used in numerous fields. Especially demand for autonomous mobile robots is increasing. In order to achieve autonomous operation of a mobile robot, in indoor and outdoor environments, some real-time systems must integrate with each other. These systems are environment perception, localization, planning, and control. In addition, a robust mobile robot platform with convenient sensors, computational hardware, networking, and software base is important.

Autonomous mobile robots should effectively complete tasks and be able to adapt to changing environmental conditions. In addition, mobile robots are required to explore their environment without colliding with any objects and obstacles. For meeting the needs of robot, first step is creating a comprehensive map to model the environment. Then the objects in the environment should be detected in the created map. This study focus on these two steps that are the essential parts of the further automation process for mobile robots.

In this study, LIDAR based mapping and object detection are aimed. LIDAR transmits signals to target objects as a series of laser pulses. The range is estimated when any solid object reflects back the emitted beam. Environment is represented with *Occupancy gridmap*, an approach which uses a probabilistic tessellated representation method. The occupancy grids maintain probabilistic estimates of the occupancy state of the cells on a world frame. For building a sensor derived map of the environment, the cell state estimates are obtained by interpreting the incoming range readings using probabilistic sensor models. After representing the environment with a map, object in environment is detected in an occupancy map which is called costmap. At the end of the study, the state of achievements is observed and interpreted.

CHAPTER TWO METHODOLOGY

2.1 LIDAR Technology

LIDAR is an acronym for Light Detection and Ranging. It is basically a system for measuring range with laser. LIDAR is one of the remote sensing technologies.

According to Siegwart and Nourbakhsh (2004), a LIDAR “consists of a transmitter which illuminates a target with a collimated beam (e.g., laser), and a receiver capable of detecting the component of light which is essentially coaxial with the transmitted beam” (p. 108). LIDARs generate a range estimate based on the time required for the light beam to reach the object and turn back. A mechanism consists of a mirror that sweeps the light beam to cover the required scene in a plane or even in three dimensions, using a rotating, nodding mirror (Siegwart & Nourbakhsh, 2004).

2.1.1 LIDAR Types According to the Operating Principles

LIDARs can be categorized into two types for operating principles.

Continuous wave (CW) LIDAR, transmits a continuous signal and ranging is carried out by modulating the intensity of the laser light. Travel time is directly proportional to the phase difference between the received and transmitted sinusoidal signal. CW LIDARs are generally used in wind measurement.

Pulse LIDAR which can be called flash laser, is most commonly used for ranging applications. It transmits signal consist of a series of laser pulses. The range is estimated base on the time taken by the transmitted pulse to the target and back. (Marcoe, 2007; Nayegandhi, 2007). Principle of operation for purpose propagation time measurement is shown in Figure 2.1.

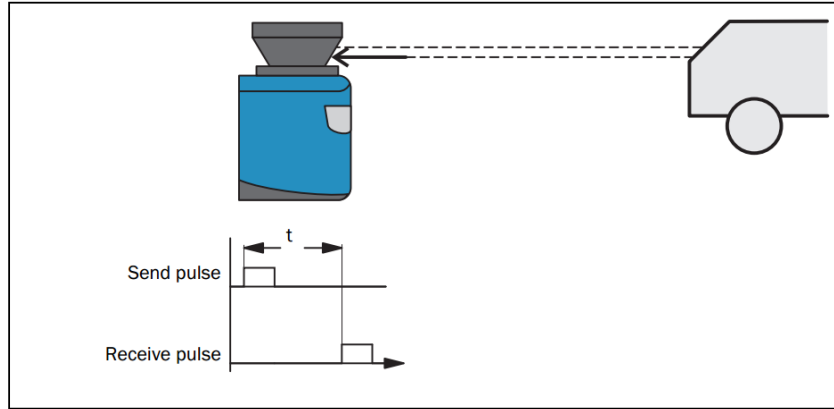


Figure 2.1 Principle of operation for purpose propagation time measurement

Range and range resolution formulas are given in the following equations for pulse laser and CW laser.

Pulse laser:

$$R = c \frac{t}{2}; \Delta R = c \frac{\Delta t}{2} \quad (2.1)$$

Where R is the range distance between sensor and object; ΔR is the range resolution, c is the speed of light (approximately 3.00×10^8 m/s), t is the time interval between sending and receiving a pulse (ns), Δt is the time resolution of time measurement (ns).

CW laser:

$$R = \frac{1}{4\pi} \frac{c}{f} \varphi; \Delta R = \frac{1}{4\pi} \frac{c}{f} \Delta \varphi \quad (2.2)$$

Where f is frequency (Hz); φ is the phase (rad); and $\Delta \varphi$ is the phase resolution (rad) (Baltsavias, 1999).

2.1.2 Applications of LIDAR

There are various applications of LIDAR systems. It can be divided into two which are terrestrial and airborne.

2.1.2.1 Airborne Laser Scanning

Airborne laser scanning (ALS) was first demonstrated in 1993 and was introduced to the mapping industry in Australia in 1998. ALS is a system when a laser scanner attached to a plane during flight. It creates a 3D point cloud model of the landscape. Beside the main part of system which is LIDAR, this system has two more components. GPS is required for determining the position of aircraft. Inertial navigation or measuring system (INS/IMU) is required for monitoring the altitude of aircraft (Crowe & Riley, 2006).

An example of ALS is shown in Figure 2.2.

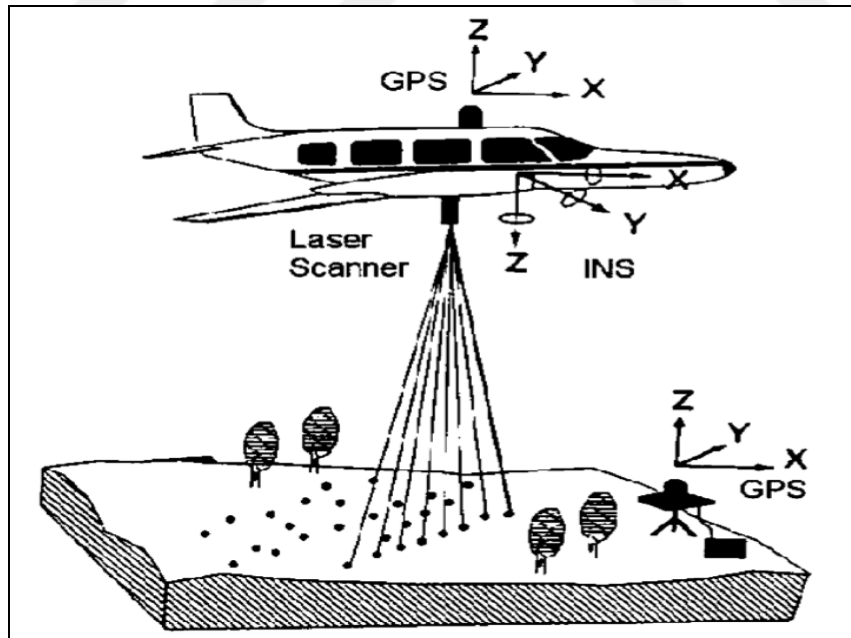


Figure 2.2 Airborne laser scanning system (Kao, Kramer, Luo, Dungan & Pang, 2005)

2.1.2.2 Terrestrial Laser Scanning

It is a ground based system. Locating laser scanner on the ground provides some specific advantages for apprehending discrete objects from various angles. *Terrestrial laser scanning* is most useful for capturing small (relative to those captured from an aircraft) irregular objects such as buildings, earthworks and landforms such as cliff faces (Crowe & Riley, 2006).

These systems can be mobile or stationary. An example of stationary terrestrial laser scanning is shown in Figure 2.3.

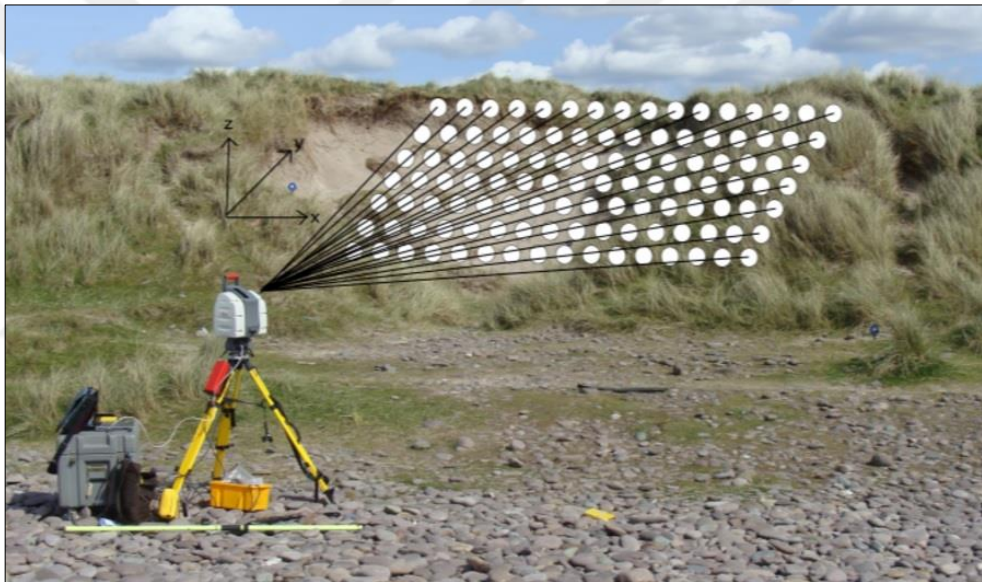


Figure 2.3 Terrestrial Laser Scanning (TLS) (Kandrot, 2013)

Terrestrial laser scanning applications are various. In a study about assessing forest metrics, the bottom and top points of trees are trees are extracted by ground-based scanning LIDAR. With help of these points, measures of tree location, tree height and stem density can be more comparable than traditional techniques (Hopkinson, Chasmer, Young-Pow & Treitz, 2004).

2.1.3 LIDAR Applications for Mobile Robots

Nowadays, LIDAR technology is widely used in autonomous systems. Mobile robots equipped with LIDARs can obtain to perform automatic acquisition and 3-D reconstruction of terrain, buildings, large infrastructures or using this information to safely navigate through unknown environments.

LIDAR sensors are used for mapping and localization when GPS signals are weak or don't exist in the urban areas.

BigDog is a well-known rough-terrain quadruped robot which developed by Boston Dynamics. LIDAR is integrated onto BigDog. The LIDAR is being used to allow BigDog to follow a human leader, without requiring the operator to drive continuously (Raibert, Blankespoor, Nelson, Playter & BigDog Team, 2008).

In another study which is about plant detection and mapping for agricultural robots, indicates that 3D LIDAR sensors are more accurate than the other sensors for working under variable weather conditions (Weiss & Biber, 2011).

LIDARs are commonly used in autonomous vehicle competitions such as DARPA. The third driverless car competition of the DARPA Grand Challenge was commonly known as the DARPA Urban Challenge. Tartan racing team's vehicle, Boss was named as a winner of this challenge. In Boss, LIDARs are mounted on the roof of the robot to evaluate the terrain around the vehicle. The following roles is filled by LIDARs; determining safe to cross/merge at intersection, determining safe to pass in oncoming traffic, detection & localize vehicles for separation, estimating road shape and lane locations, detection of static obstacles in the road (Urmson et al., 2007).

2.2 Mapping

The term of mapping in general is the process of making a map which represents the significant features of a piece of the surface of the Earth.

2.2.1 Robotic Mapping

Robotic mapping is a discipline which deals with the study and application of creating map or floor plan by the robot and to localize itself in it.

Robot has two sources of information; *idiothetic* and *allothetic* information. Idiothetic information source which corresponds *odometry* in robotics, supplies internal information about the movements of robot. It may concern wheel rotation for robots. In dead reckoning approach, by tracking the number of revolutions of its wheels, data can be gathered and robot's position can be known. Allothetic information source which corresponds perception and sensor data, supplies external information about the environment. The information may be derived from laser sensors, sonar or vision for robots (Filliat & Meyer, 2003).

In mobile robots, various sensors can define the surroundings. Ultrasonic sensors, LIDARs, radars and vision sensors like cameras are the common types. Most sensors have range limitations. For instance, sound and light are not able to go through walls. These range limitations make it necessary for a robot to navigate in its environment when creating a map (Thrun, 2002).

Accuracy of map depends on accuracy of sensor data. Sensors have errors which can be called as measurement noise. Especially in multi-sensor systems measurement noise escalates by multiple sensors.

A crucial challenge in robotic mapping appears from the nature of the measurement noise. Problems caused by measurement noise are generally easy to solve if the noise in various measurements is statistically independent. In that case,

for eliminating the effects of the noise, a robot could take more measurements. Unfortunately, the measurement noises are statistically dependent in robotic mapping. This is due to errors in control accumulate over time so they affect the way forthcoming sensor measurements are interpreted. Dealing with such systematic errors is key to building maps successfully (Thrun, 2002).

Sensors can be used individually or combined with each other for more robust systems. Mapping via camera is a high level application. Cameras provide data with color and high resolution. Because of these characteristics, modeling surroundings is superior than the lower resolution LIDAR. On the other hand, mapping via camera is related with image processing algorithms and they are required high processing performance.

All state-of-the-art algorithms for robotic mapping in the literature have one common feature: They are probabilistic.

2.2.2 Map Representations

The internal representation of the map can be topological or metric.

2.2.2.1 Metric Map

The environment is represented as a group of objects with coordinates in a two-dimensional space in the metric framework. *Metric maps* display the layout of the environment in a way similar to an architectural sketch, which is less demanding to comprehend for people. This objective view of the environment, which is rather independent of any given robot, likewise makes it easy for various robots to reuse such maps. Metric maps are easier to create than *topological maps* as a result of the unquestionable definition of locations managed by their coordinates (Filliat & Meyer, 2003; Thrun, 1999).

In the map, rather than utilizing a group of features to represent objects, it is likely to represent the part of the environment which is approachable to the robot. This method can be called *Occupancy grid map*. In that method, the metric map represents the environment with fixed resolution grids. Information of the observed environment is hold by these grids. Each grid cell represents some amount of space in the real world. For building map, objects and obstacles are detected by sensors and the collected sensor data are used to determine the occupancy value of each grid. An example of occupancy map is shown in Figure 2.4 (Filliat & Meyer, 2003; Habib, 2007).

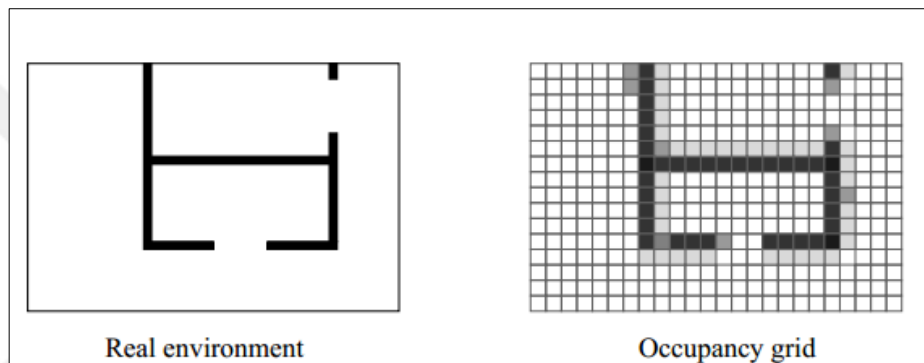


Figure 2.4 Representation of occupancy grid map (Filliat & Meyer, 2003)

Metric maps are uncomplicated to build since grids recreate explicitly the metrical structure of the environment. In addition, the grid cells geometry corresponds directly to the real environment, so that the robot orientation and position within its model can be determined by its orientation and position in the real world. Metric maps have the advantage of its high resolution and of being well suited for robot navigation tasks.

There are some disadvantages of using solely metric maps such as being exposed to errors in both map-making and dead-reckoning abilities of the robot. In large scale environments, odometry drifts cause problems for map maintenance. It makes the global stability of the map hard to sustain. Metric maps are containing no information about the various types of objects or places in the environment (Habib, 2007).

Illustration of distinction between metric map and topographical map is shown in Figure 2.5.

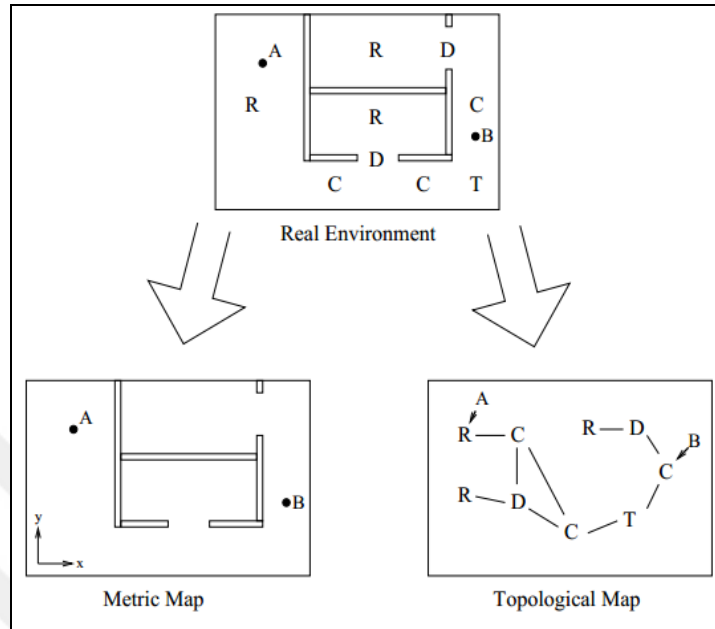


Figure 2.5 Distinction between metric map and topographical map (Filliat & Meyer, 2003)

2.2.2.2 Topographical Map

In the topographical framework, the environment represented by a group of peculiar places and relations between them. Topographical maps define the connectivity of various places and it can be automatically extracted while creating metric maps. These maps can also be enhanced by geometrical relations between places to support tasks such as planning and navigation.

Advantages of topographical maps are that they are less sensitive to measurement noises, less complex and enable more efficient planning than metric maps. Nonetheless, they are incompetent to describe individual objects in the environment. Moreover, topographical maps are usually difficult to learn and maintain in large scale environments (Habib, 2007).

2.3 Filtering (Estimation) Theory

Filtering theory which is also known as estimation theory, has gained a lot of attention as a result of its practical importance in solving problems in scientific and engineering fields. Due to the combined research attempts of many scientists, various algorithms have been developed. These algorithms can be divided into two main categories which are linear and nonlinear filtering algorithms, corresponding to linear or linearized dynamic models with Gaussian noise and to nonlinear or non-Gaussian models.

Recursive Bayesian estimation which can be called as a *Bayes filter*, is a stochastic method for estimating unknown density of a continuous random variable or *probability density function* (PDF) recursively over time using arriving measurements and a mathematical model.

State estimation problems can be dealt with the Bayesian filters. In state estimation problems, the available measurement data is used along with prior knowledge about the physical phenomena and the measuring devices, in order to consecutively generate estimates of the demanded dynamic variables. This is succeeded in such approach that the error is statistically minimized.

Kalman filter and *particle filter* are the two well-known Bayes estimation applications.

2.3.1 Kalman Filter

The Kalman filter is a group of mathematical equations that supplies an efficient computational means to estimate the state of a process, in a way that minimizes the mean of the squared error. The Kalman filter is very effective in several perspectives. The filter provides estimations of present, past, and even future states. It can do so even when the precise character of the system model is unknown (Welch & Bishop, 1995).

Kalman filters focus on the general problem of trying to estimate the state $x \in R^n$ of a discrete time controlled process that is administrated by linear probabilistic difference equation with a measurement $z \in R^m$.

$$\begin{aligned}x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\z_k &= Hx_k + v_k\end{aligned}\tag{2.3}$$

The random variables and represent the process and measurement noise. They are assumed to be independent of each other and with normal probability distributions

$$p(w) \sim N(0, Q), \quad p(v) \sim N(0, R).\tag{2.4}$$

In practice, the process noise covariance and measurement noise covariance matrices might change with each time step or measurement, however here it is assumed that they are constant.

It is defined to be *a priori state* estimate at step k given knowledge of the process prior to step k, and to be *a posteriori state* estimate at step k given measurement. A priori and a posteriori estimate errors can be defined as

$$\begin{aligned}e_k^- &\equiv x_k - \hat{x}_k^-, \text{ and} \\e_k &\equiv x_k - \hat{x}_k.\end{aligned}\tag{2.5}$$

a priori estimate error covariance is then,

$$P_k^- = E[e_k^- e_k^{-T}]\tag{2.6}$$

and *a posteriori* estimate error covariance is

$$P_k = E[e_k e_k^T]\tag{2.7}$$

In deriving the equations for the Kalman filter, it is begun with the goal of finding an equation that computes an *a posteriori state* estimate as a linear combination of an *a priori estimate* and a weighted difference between an actual measurement and a measurement prediction as shown in Eq. (2.8). Some justification for Eq. (2.8) is given in “The Probabilistic Origins of the Filter” found below.

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

The difference $(z_k - H\hat{x}_k^-)$ in Eq. (2.8) is called the measurement *innovation*, or *the residual*. The residual reflects the discrepancy between $H\hat{x}_k^-$ the predicted measurement and the actual measurement z_k . A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix K in Eq. (2.8) is chosen to be *the gain* or *blending factor* that minimizes the *a posteriori* error covariance Eq. (2.7). This minimization can be accomplished by first substituting Eq. (2.8) into the above definition for e_k , substituting that into Eq. (2.7), performing the indicated expectations, taking the derivative of the trace of the result with respect to K , setting that result equal to zero, and then solving for K . One form of the resulting K that minimizes Eq. (2.7) is given by

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} = \frac{P_k^- H^T}{H P_k^- H^T + R} \quad (2.9)$$

In Figure 2.6, a complete picture of the operation of the Kalman filter is shown. Notice how the time update equations in Figure 2.6 project the state and covariance estimates forward from time step $k-1$ to step k . A and B are from Eq. (2.3), while Q is from Eq. (2.4). The first task during the measurement update is to compute the Kalman gain, K_k . It is from Eq. (2.9). The next step is to actually measure the process to obtain z_k , and then to generate an *a posteriori* state estimate by incorporating the measurement as simply Eq. (2.8) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via $P_k = (1 - K_k H) P_k^-$ (Welch & Bishop, 1995).

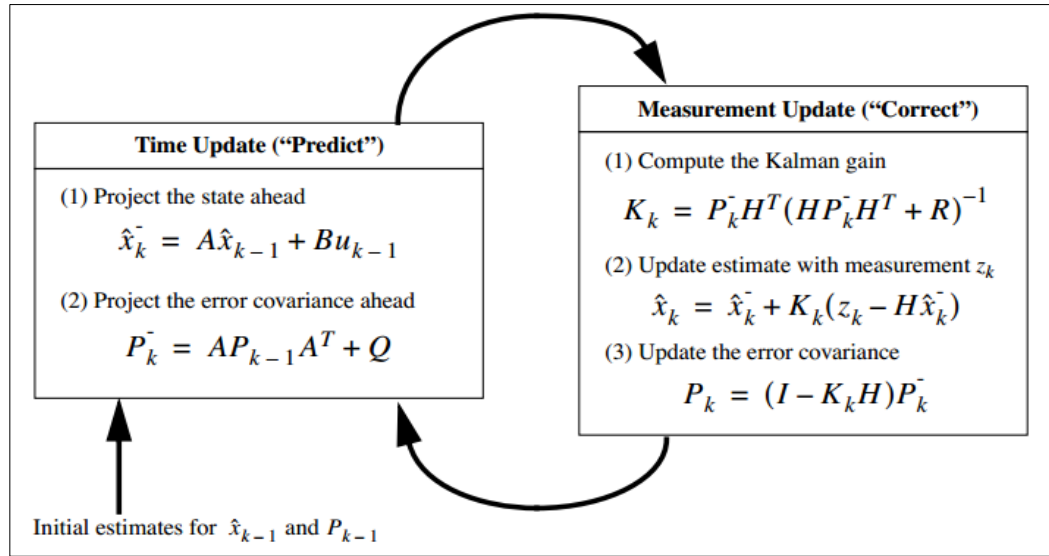


Figure 2.6 Operation of Kalman filter (Welch & Bishop, 1995)

The Kalman Filter widely used in linear systems with Gaussian noise. The first application that publicly known was done at NASA Ames Research Center during Apollo program in early 1960s (McGee and Schmidt, 1985; Schmidt, 1981).

The fields in which the Kalman filter is widely used include not only aerospace engineering but also robotics, economics, mechanical engineering, electrical engineering, chemical engineering, biology and many others.

The applications of the Kalman filter are limited to linear models with Gaussian noises. The Kalman filter extensions were developed for less limiting cases by using linearization methods.

2.3.1.1 Extended Kalman Filter (EKF)

Extended Kalman filter can be referred as nonlinear Kalman filter. When the process to be estimated or the measurement relationship to the process is non-linear, Kalman filter cannot deal with it. For these non-linear problems, another approach is required. According to Welch and Bishop, "A Kalman filter that linearizes about the

current mean and covariance is referred to as an extended Kalman filter or EKF” (Welch & Bishop, 1995, p. 7).

The extended Kalman filter process is shown in Figure 2.7.

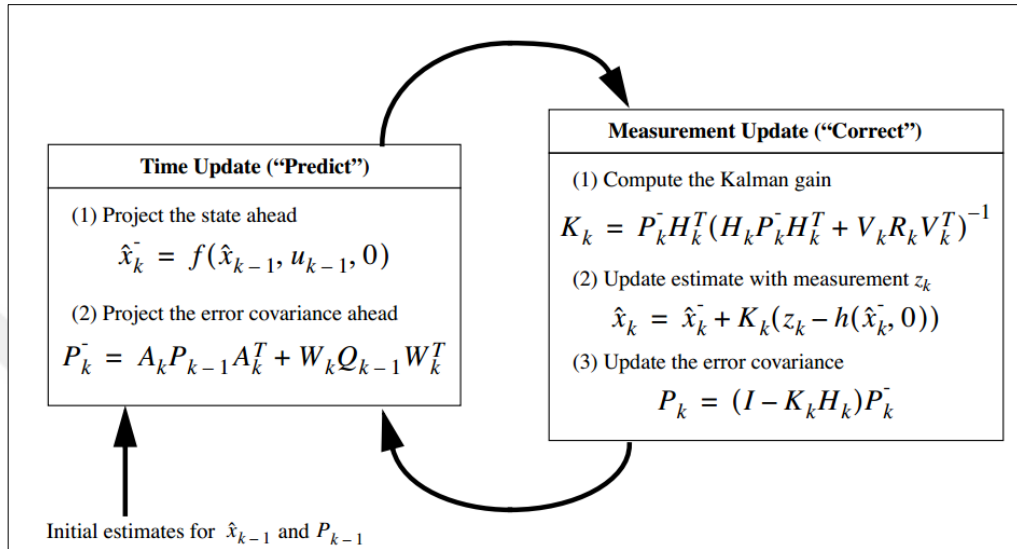


Figure 2.7 The operation of the *extended* Kalman filter (Welch & Bishop, 1995)

2.3.1.2 Unscented Kalman Filter (UKF)

Kalman Filter with the Unscented transformation can be called as *Unscented Kalman Filter*.

Unscented Kalman Filter was introduced in 1997 by Julier and Uhlmann. They indicated that “Using the principle that a set of discretely sampled points can be used to parameterise mean and covariance, the estimator yields performance equivalent to the KF for linear systems yet generalises elegantly to nonlinear systems without the linearisation steps” which is required by the Extended Kalman filter (Julier & Uhlmann, 1997, p. 182).

The process of unscented Kalman filter is shown in Figure 2.8 which is composed of two main parts, similar to the Kalman Filter. Time update is the first part of this process, in where the initial state estimate is computed via selecting sigma points,

solving for its mean and covariance. Also, the observation is propagated in this step and its mean and covariance are also calculated. The measurement update is the second part of this process. The Kalman gain and cross-covariance of the propagated state and the propagated observation are calculated. They used to update the state and states covariance. Difference from Kalman Filter is that the initial state estimate is obtained from the sigma (particle) propagation (Causeo et al., 2010).

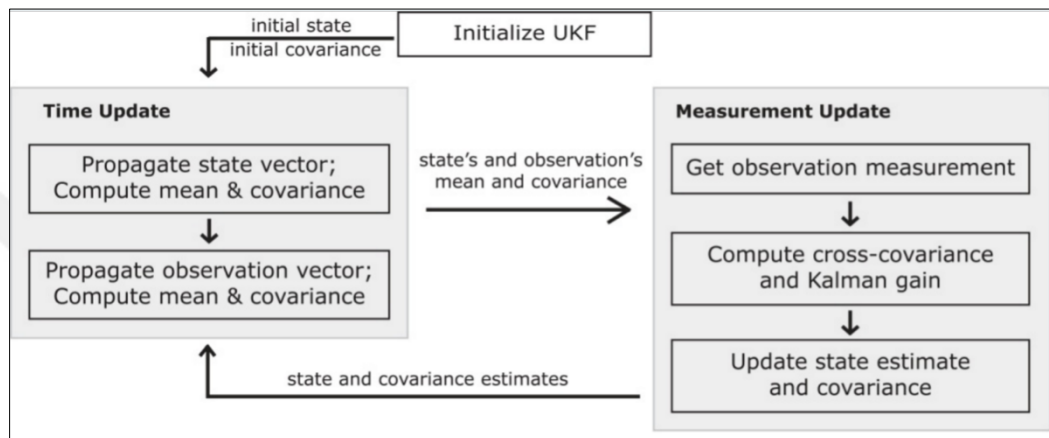


Figure 2.8 The operation of the *unscented* Kalman filter (Causeo et al., 2010)

If the system nonlinearities are severe the extended Kalman filter can be difficult to tune and often gives unreliable estimates. So, in those applications, unscented Kalman filter can be preferred (Simon, 2006).

Despite the fact the unscented Kalman filter is a relatively recent development, it is quickly finding applications in areas such as aircraft model estimation, neural network training, financial forecasting and motor state estimation.

2.3.2 Particle Filter

Particle filters are also known as Sequential Monte Carlo (SMC) methods. Particle filter was introduced as a numerical approximation for the nonlinear Bayesian filtering problem.

According to Salmond and Gordon, “the particle filter is not another variant of the EKF: it does not stem from linear-Gaussian or least-squares theory” (Salmond & Gordon, 2005, p.22).

Main concept of particle filter is finding an approximate solution using a complex model instead of an exact solution using a simplified model.

Particle filter is an implementation of the recursive Bayesian filter using sequential Monte Carlo approach. Instead of portraying the required probability density function as a functional form, in this approach it is represented approximately as a group of random samples of the probability density function. These random samples are the particles of the filter which are propagated and updated in accordance with the dynamics and measurement models. In contrast to the Kalman filter, this method is not limited by linear Gaussian assumptions. So, various extending the range of problems can be succeeded. The basic form of the particle filter is very simple. Computationally tractable for large or high dimensional problems. However, it can be computationally expensive (Salmond & Gordon, 2005).

The particle filter can be summarized as follows. The system and measurement equations are given as:

$$\begin{aligned}x_{k+1} &= f_k(x_k, w_k) \\ y_k &= h_k(x_k, v_k)\end{aligned}\tag{2.10}$$

where $\{w_k\}$ and $\{v_k\}$ are independent white noise processes with known pdf's.

Assuming that the pdf $p(x_0)$ of the initial state $p(z_0)$ is known, randomly generate N initial particles on the basis of the pdf $p(z_0)$. These particles are denoted $x_{0,i}^+(i = 1, \dots, N)$. The parameter N is chosen by the user as a trade-off between computational effort and estimation accuracy.

For $k = 1, 2, \dots$, the following is done.

(a) Performing the time propagation step to obtain a priori particles $x_{k,1}^-$ using the known process equation and the known pdf of the process noise:

$$x_{k,1}^- = f_{k-1}(x_{k-1}^+, w_{k,1}^+)(i = 1, \dots, N) \quad (2.11)$$

(b) Computing the relative likelihood q_i of each particle $x_{k,1}^-$ conditioned on the measurement y_k . This is done by evaluating the pdf $p(y_k|x_{k,1}^-)$ on the basis of the nonlinear measurement equation and the pdf of the measurement noise.

(c) Scaling the relative likelihoods obtained in the previous step as follows:

$$q_i = \frac{q_i}{\sum_j^N q_j} \quad (2.12)$$

Now the sum of all the likelihoods is equal to one.

(d) Generating a set of a posteriori particles $x_{k,1}^-$ on the basis of the relative likelihoods q_i . This is called the resampling step.

(e) Now that we have a set of particles $x_{k,1}^-$ that are distributed according to the pdf $p(x_k|y_k)$, any desired statistical measure of this pdf can be computed. We typically are most interested in computing the mean and the covariance.

Particle filters are used in various fields especially wherever there is a necessity to estimate the state of a probabilistic evolving system using ambiguous and noisy measurement data. These fields can be mentioned as robotics, computer vision, econometrics, numerical weather prediction etc.

In robotics, some of the applications can be mentioned as robot fault diagnosis, robot localization, SLAM, tracking with non-standard sensors, tracking and

navigation with a bounded support, multiple object tracking and association uncertainty.

In localization, particle filters are also known as *Monte Carlo Localization (MCL)*. This method represents the probability density involved by maintaining a group of samples which are randomly drawn from it. Localization approach which can represent arbitrary distributions is obtained via a sampling-based representation. The resulting method is capable to competently localize mobile robots without knowing of their starting location. In comparison with prior approaches such as *Markov localization*, this method is more accurate, less memory-intensive and easier to implement due to sampling based representation (Dellaert, Fox, Burgard, & Thrun, 1999; Thrun, 2002).

2.3.2.1 The Rao-Blackwellized or Marginalized Particle Filter

Rao-Blackwellized particle filter is an extension of the particle filter that uses Rao-Blackwellization approach. Rao-Blackwellization is a technique that marginalize out some of the variables.

In some cases, it is possible to divide the problem into two parts; linear-Gaussian and non-linear parts. Assume that the state vector may be partitioned as $x_k = \begin{pmatrix} x_k^L \\ x_k^N \end{pmatrix}$ so the required posterior may be factorized into Gaussian and non-Gaussian terms:

$$p(x_k|Z_k) = p(C, x_k^N|Z_k) = p(x_k^L|x_k^N, Z_k) p(x_k^N|Z_k) \quad (2.13)$$

where $p(x_k^L, x_k^N|Z_k)$ is Gaussian (conditional on x_k^N) and $p(x_k^N|Z_k)$ is non-Gaussian. In other words, the linear component of the state vector x_k^L can be “marginalized out”. Essentially, the term $p(x_k^L, x_k^N|Z_k)$ may be obtained from a Kalman filter while the non-Gaussian part $p(x_k^N|Z_k)$ is given by a particle filter. The scheme requires that a Kalman filter update be performed for each x_k^N particle.

Rao-Blackwellized particle filter has a major advantage that the dimension of the particle filter state x_k^N is smaller than the full state vectors dimension. Therefore, less particles are needed for satisfying filter performance. That comes at the amount of a more complicated algorithm, in spite of the operation count of the marginalized filter for a given number of particles can really be less than that of the regular algorithm (Salmond & Gordon, 2005).

Rao-Blackwellized particle filters have a big role to play in computer vision where mixtures of Gaussians arise commonly, robotics dynamic factor analysis (Doucet, De Freitas, Murphy & Russel, 2000).

Gmapping which is used in this study, is an improved implementation of Rao-Blackwellized Particle Filter. This approach uses a particle filter in which each particle carries an individual map of the environment.

Gmapping was presented as an improved method for learning grid-maps with the help of Rao-Blackwellized particle filters. This method computes an immensely accurate proposal distribution based on the observation possibility of the most recent sensor information, a scan-matching process and the odometry. This allows us to draw particles in a more precise way that really lessens the quantity of required samples. In addition, selective resampling strategy based on the effective sample size was applied. This way lessens the quantity of unnecessary resampling actions in the particle filter and so substantially decrease the risk of particle depletion. Gmapping approach has been implemented and evaluated on data acquired with different mobile robots that equipped with LIDARs. Tests performed with gmapping algorithm in various large scale environments have demonstrated its robustness and the capability of creating maps with high quality. In gmapping experiments, the quantity of particles needed by gmapping method often was one order of magnitude smaller compared to previous approaches (Grisetti, Stachniss & Burgard, 2005, 2007).

2.4 Object Detection

In autonomous systems, path planning is a crucial task to accomplish. Generally, both in indoor or outdoor environment, obstacles are existed. While mobile robots are fulfilling the given tasks, obstacles in the environment are required to avoid or overcome.

Object detection is the key part of the obstacle avoidance task. For detection two main method is available; image-based detection and range-based detection. Various sensors can be used for this purpose. LIDAR, infrared sensors, ultrasonic sensors can be used in range-based detection. Vision sensors such as camera or kinect sensor can be used in image-based detection.

Obstacles in environment represent a fundamental hindrance as natural hazards. They can prevent mobile robots from achieving specific tasks by abolishing path plans. As a worst case, they may do physical harm to the mobile robot. Nowadays, obstacle avoidance systems are able to detect most of the obstacles by looking for signs such as gradient of range.

LIDAR measures the exact distance to the object. However, it cannot able to detect overhanging obstacles and low objects due to its constant, defined, scanning angle and height. This problem can be accomplished by mechanic systems such as tilting or moving LIDAR mechanisms. Also it can be dealt with a similar case of us, a mobile robot which can overcomes some defined-height obstacles with its mechanical design.

In Tartan racing team's vehicle which was the winner of DARPA Urban Challenge 2007, detection algorithm for static obstacles uses LIDARs mounted on the roof of the vehicle evaluate the field around the vehicle. For representing the *traversability* of the field, a costmap is computed by this algorithm. The root of the algorithm based on comparisons of couples of laser points. Each arriving laser point is compared to its neighbours which contain points. The maximum cost is registered

as the cost of that point. The maximum cost for all points in a cell is maintained and placed in a cost map. The obstacle placement accuracy is apparently dependent on the cost map resolution and the sensor calibration. Tartan racing team indicated that, “algorithm properly places cones within 25 [cm] of their location. In most cases this is sufficient to navigate through obstacle fields with cones and walls” (Urmson et al., 2007, p. 15).

2.4.1 Costmap

Costmap is a 2D occupancy grid map which includes information about objects/obstacles. This approach is generally used for avoiding obstacles and performs precise path-planning.

Various navigation systems perform obstacle avoidance in path-planning stage, on a single costmap. In a single costmap, larger part of data is stored in a single grid. Layered Costmaps method is efficient at creating collision-free paths of minimal length. However, this approach may trouble in dynamic environments when the values in the costmap expand beyond occupied or free space (Lu, Hershberger & Smart, 2014).

2.5 Robot Operating System (ROS)

Robot operating system which is mostly known as ROS is a framework for writing robot software. ROS is an open source environment. It has large ecosystem of developers and existing code.

For understanding and working in ROS environment, the basic definitions should be known. The fundamental concepts of the ROS are nodes, topics, messages and services. This concept is shown in Figure 2.9 (*ROS concepts*, n.d.).

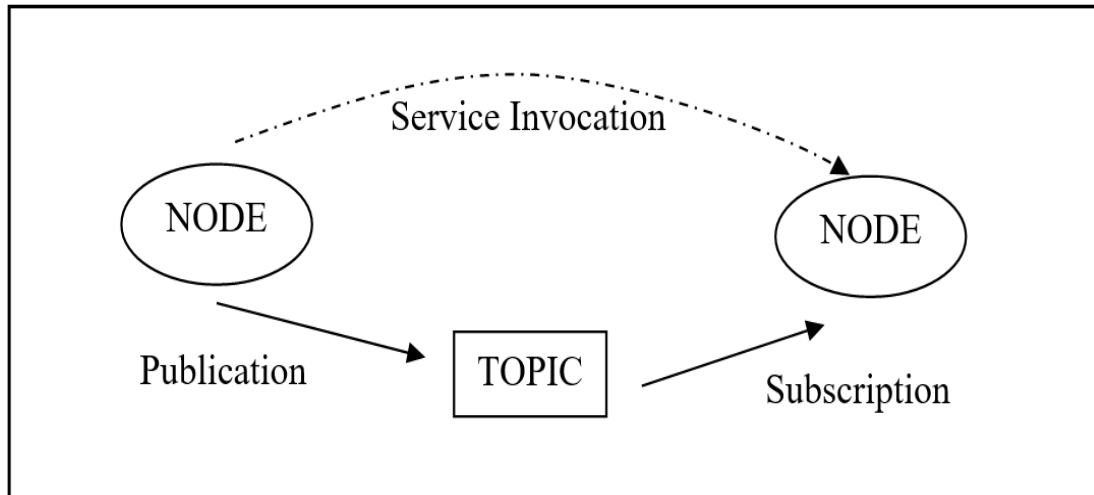


Figure 2.9 Concept of ROS

Nodes are processes that perform computation. A node is an executable that uses ROS to communicate with other nodes. Each node performs a single task. Nodes use a publish-subscribe model for communicating with other nodes on a named topic.

Messages are simple data structures. Nodes communicate with each other by passing messages.

Topics are streams of data. Topics publish streams of structured messages, supplying standardized interfaces for frequently used data structures. Nodes can publish data to a topic or subscribe to data published onto it. Data can come consistently such as laser scans or intermittently such as email.

Services are like a function call or an RPC (remote procedure call). The publish-subscribe model is a very flexible communication paradigm but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.

ROS Master is a program that stores information about the network. Nodes register themselves with the Master on startup and nodes ask the Master where to find other nodes after that, nodes established peer-to-peer communication with each other. The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages or invoke services.

Bags are a format for saving and playing back ROS message data. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.

CHAPTER THREE

EXPERIMENTAL SETUP

The mobile robot has seven wheels, a steering wheel in the front, two wheels are arranged on a *bogie mechanism* (parallel arm) on each side and two wheels in the rear. Five *DC motors* are used for propulsion. They are located in the wheels of bogie mechanisms and the front arm. In addition, the mobile robot has a *servo motor* which angular position changes depending on the angle of the front wheels. It is located in wheel of front arm. This arm has a spring suspension to guarantee optimal ground contact of all wheels at any time. The technical drawings of mobile robot are shown in Figure 3.1 and 3.2 (Yıldız & Gören, 2012).

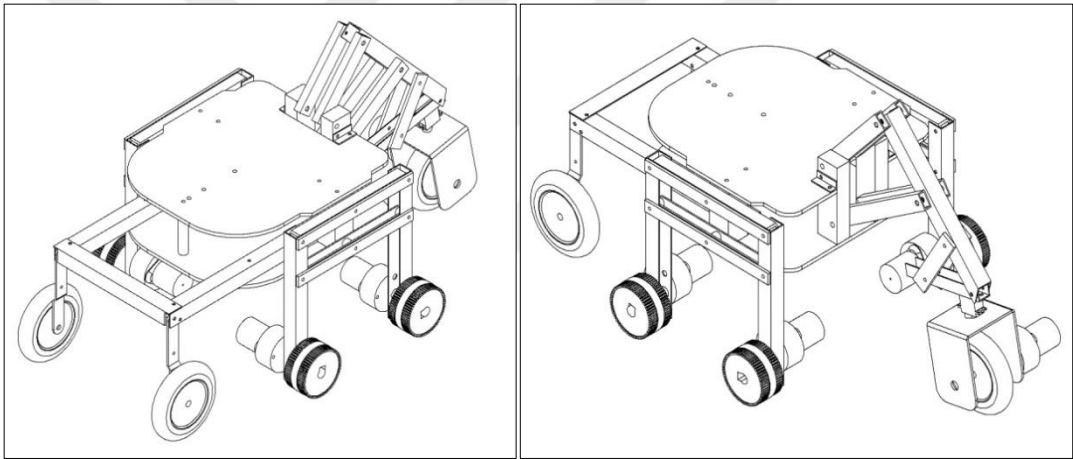


Figure 3.1 Technical drawing of mobile robot

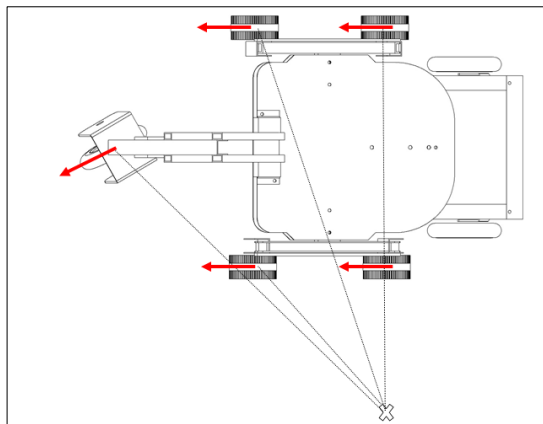


Figure 3.2 Top view of the mobile robot

Essential measures of the mobile robot are given in the Table 3.1.

Table 3.1 Measures of the mobile robot

Parameter	Value [mm]
Height	465
Distance between bogies	175
Distance between the rear wheel and the bogie wheel	155
Distance between the front wheel and the bogie wheel	190
Distance between the middle of the rear wheels	235
Distance between the middle of the bogie wheels	340
Diameter of front wheel	108
Diameter of rear wheel	80
Diameter of bogie wheel	60
Maximum width dimension	380
Maximum length dimension	620

The mobile robot is modified for this study which is shown in Figure 3.4. The LIDAR is built in mobile robot as a main part of the perception system. The modified technical drawing of the mobile robot is given Figure 3.3.

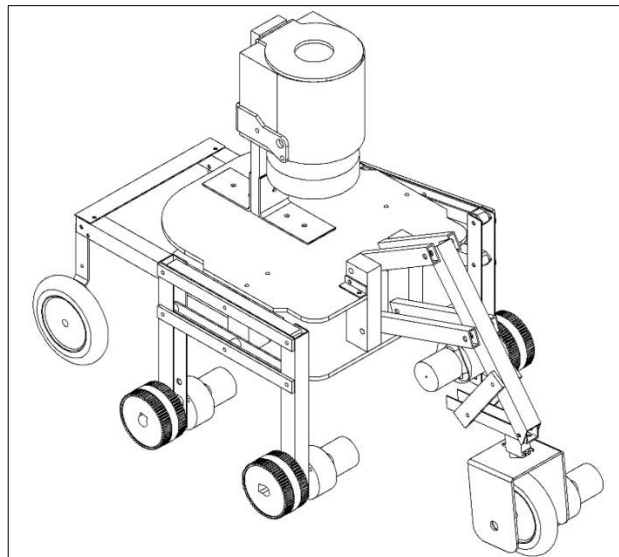


Figure 3.3 Technical drawing of modified mobile robot

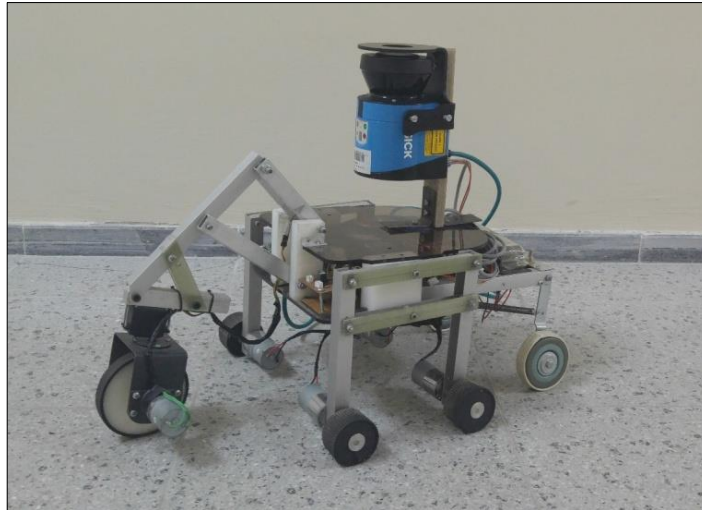


Figure 3.4 Experimental mobile robot

LMS is an electro-optical laser measurement system that scans the perimeter of its surroundings in a plane with the help of laser beams. The LMS measures its surroundings in two-dimensional polar coordinates. When a beam of laser is occurrence on an object, the position is decided in the form of direction and distance (SICK AG, 2010).

For this study, LMS100 from SICK AG LMS1xx laser scanner family is used. LMS100 is shown in Figure 3.5.



Figure 3.5 LMS100 from LMS1xx family

Data acquisition is performed using this LMS100 LIDAR that features a range between 0.5 m and 20 m. Functional specification of the laser scanner LMS100 is shown in the Table 3.2 and the scanning range of LMS100 is shown in Figure 3.6. For further information about LMS1xx, Appendix-1 can be examined.

Table 3.2 Functional specifications of LMS100 (Sick AG, 2010)

Parameter	Value
Scanning frequency	25 Hz – 50 Hz
Scan angle	270°
Distance measuring range	0.5 m – 20 m
Angular resolution with 25 Hz	0.25° – 0.5°
Angular resolution with 50 Hz	0.5°
Measuring error:	
Systematic error	±30 mm (max ±50 mm)
Statistical error (1 σ)	12 mm (max 20 mm)

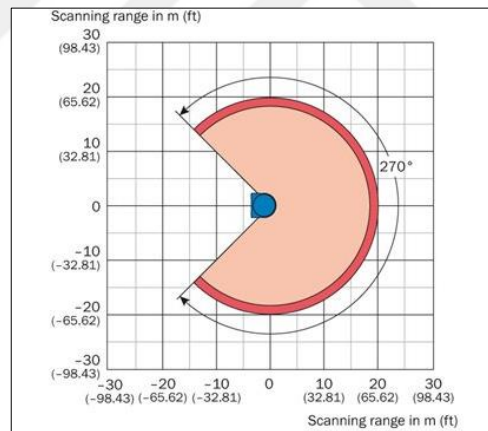


Figure 3.6 Scanning range of LMS100 (SICK AG, 2010)

As seen above the LMS100 has a maximum range which is 20 meters. So that, maximum area that can be built in map for an instant, can be calculated as 28 m x 28 m. This area can be seen as blue in Figure 3.7.

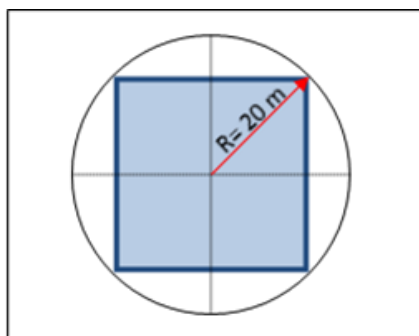


Figure 3.7 Maximum instant area for map building with LMS100

Configuration of LMS100 should be adjusted in SICK's own program SOPAS Engineering Tool (SOPAS ET). Angular resolution, scanning frequency and scan angles can be modified within limits of LMS100. This configuration screen is shown in Figure 3.8.

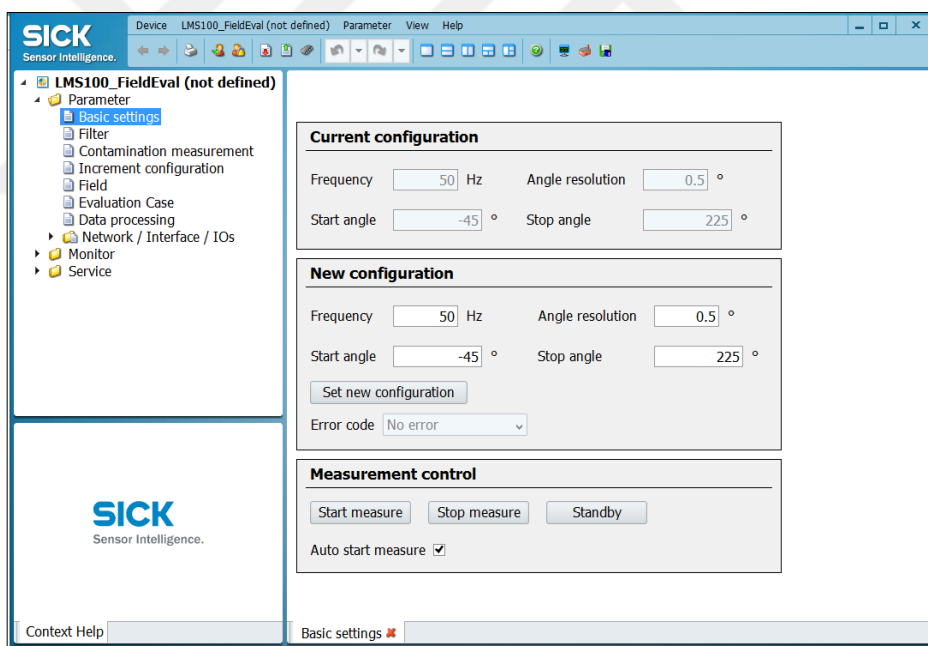


Figure 3.8 Configuration of LIDAR

The host computer has Windows operating system and ROS only has experimental Windows support. Due to that reason, Linux operating system based Ubuntu which is officially supported by ROS is decided to use. Virtual machines are adequate to run different operating systems at one computer. A virtual machine

program called VirtualBox is installed. It is shown in Figure 3.9. Via this application Ubuntu could be installed and work properly.

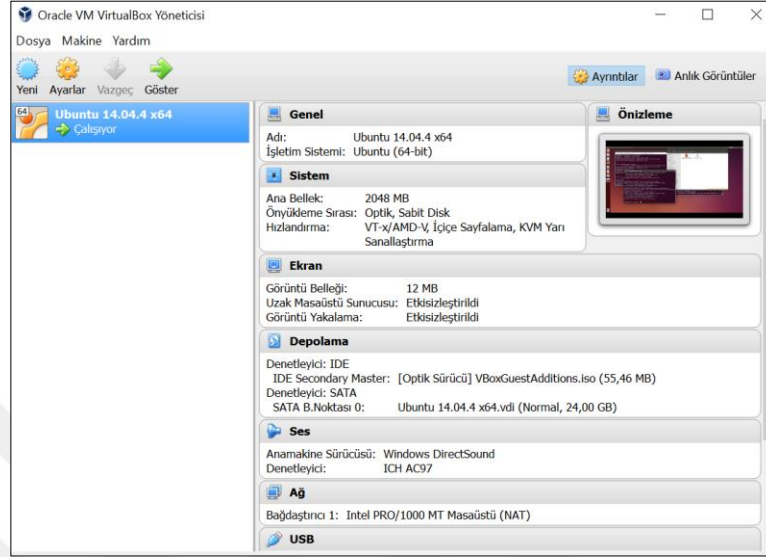


Figure 3.9 VirtualBox as a virtual machine

It is only possible to output all measured values of a scan in real-time using the Ethernet interface. So in this study, connection between LIDAR and the host computer is provided via Ethernet interface that has a data transmission rate of 10/100 MBit. The interface is a TCP/IP interface. The host computer has 2.3 GHz Intel Core processor and located on the mobile robot.

The area of study is the Automatic Control and Robotics Laboratory of Mechanical Engineering building at Dokuz Eylül University.

Automatic Control and Robotics Laboratory is shown in Figure 3.10. Part of the study area, the main corridor is shown in Figure 3.11. Layout of the study area is shown in Figure 3.12.



Figure 3.10 The area of the study, Automatic Control Laboratory of Mechanical Engineering Department, DEU



Figure 3.11 The area of the study, main corridor

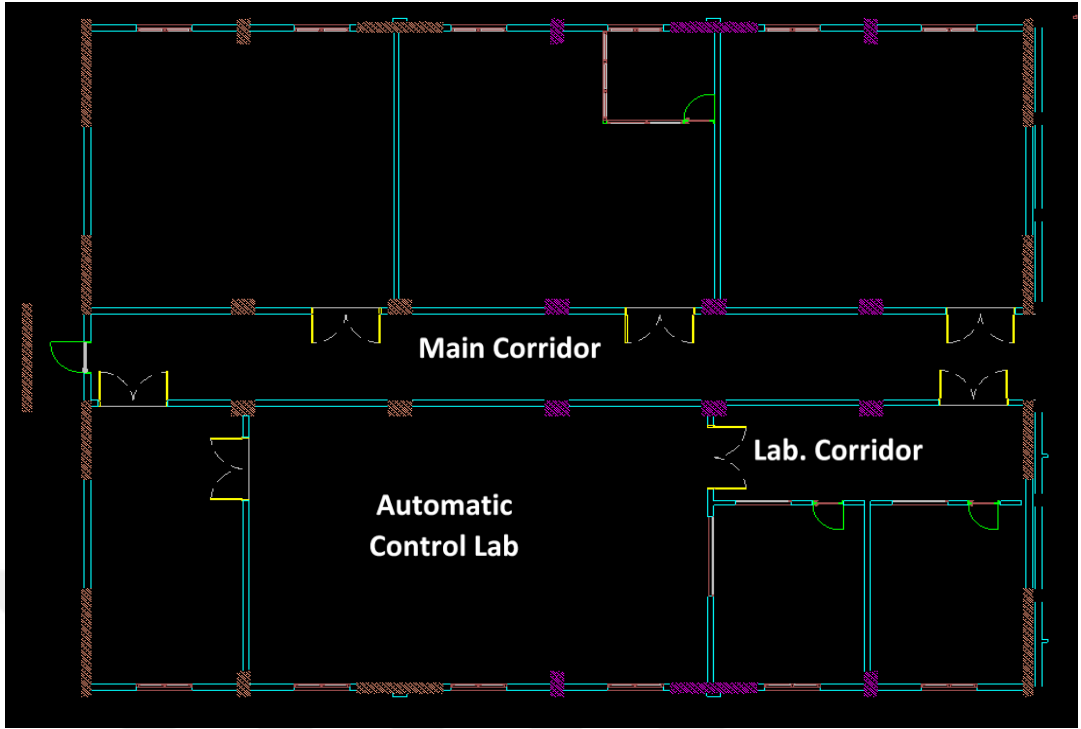


Figure 3.12 The layout of the study area

CHAPTER FOUR

EXPERIMENTAL WORK

As mentioned before, for operating autonomously in mobile robots, some steps should be achieved. In Figure 4.1, it is shown the process of an autonomous mobile robot. This thesis is aimed to examine the process from LIDAR data acquisition to creating costmaps for further parts of the automation for mobile robots.

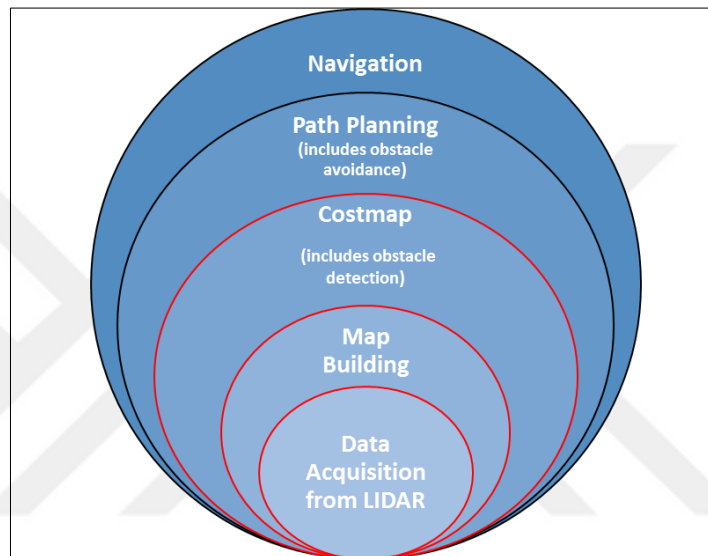


Figure 4.1 Overview of the process for autonomous mobile robots

4.1 Getting Reliable LIDAR Data

Robotic mapping addresses the problem of obtaining spatial models of the physical environments. For this problem reliable data acquisition is essential. Robustness of the control depends on the reliability of the sensor measurement.

In the early state of this study with LIDAR, laser scanner data are monitored in SOPAS ET. Monitor screen is shown in Figure 4.2. Via this monitor screen, other than regular laser scan data, mirrored laser scan data can be shown, also scanning angle can be changed. Desired point's distance from the origin and its coordinates also can be shown in this monitor.

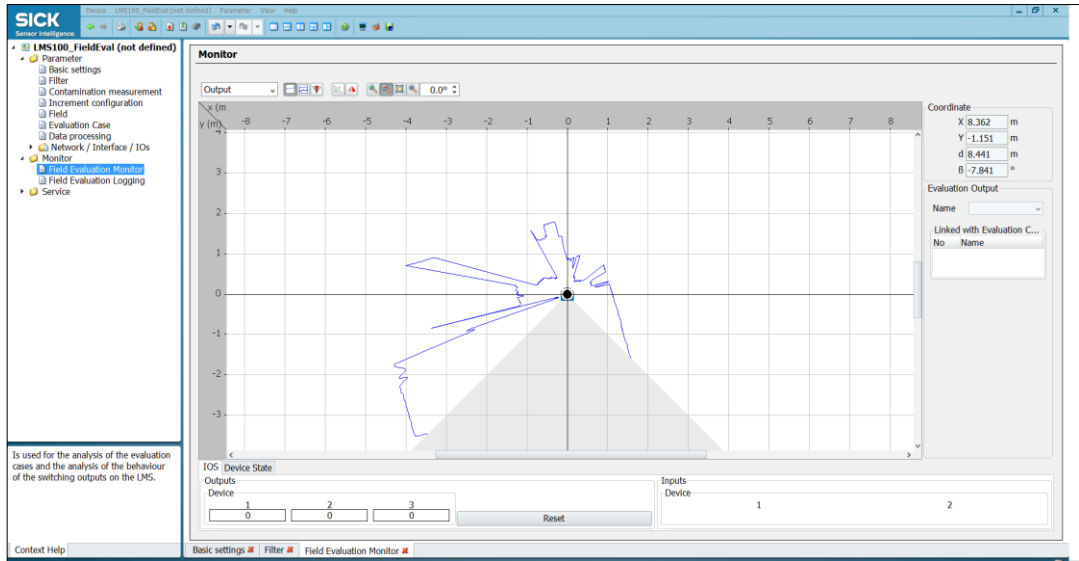


Figure 4.2 Monitoring laser data from LMS100 LIDAR

A small and basic experimental area is created. The area is compared with the scan data from LIDAR. The results are shown in Figure 4.3.

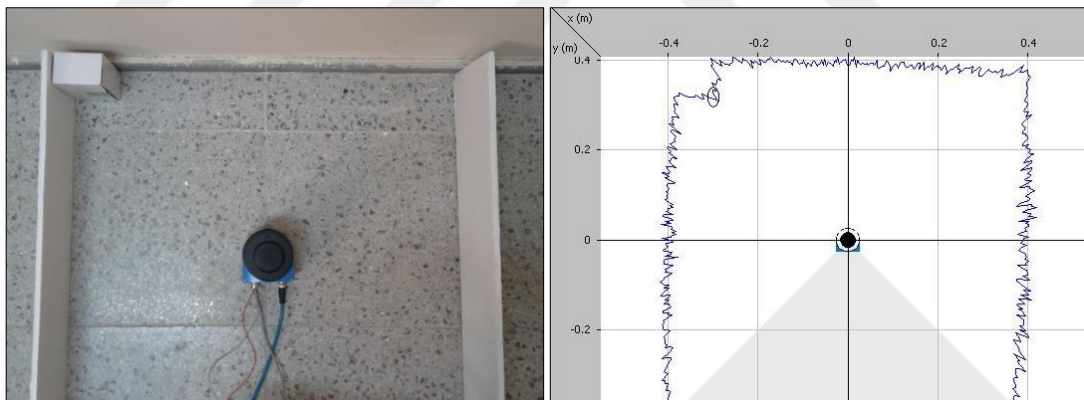


Figure 4.3 Real area and the observed area

SOPAS ET allows monitoring the laser scan data in real time. For further studies, proper data understanding is required. For understanding LIDAR data further, data are examined in Excel. For this part of experiments laser data are recorded in own program of LMS100, SOPAS ET. When recording scan data, a few points should be kept in mind. Configuration parameters and desired file extension should be chosen properly. After this configuration and data recording procedure, recorded data are

transferred to Excel framework. In our case, ScanData is chosen as a recorded parameter.

Usually each measurement is expressed in the polar coordinates (r, theta) as mentioned before. Also, the laser scan can be expressed in its equivalent Cartesian form (x, y). For this, a transformation needs to be done. It is shown in Figure 4.4. Transformation formulas are given in the following equations.

$$x = r * \text{Cos}(\theta * \frac{\pi}{180}) \quad (4.1)$$

$$y = r * \text{Sin}(\theta * \frac{\pi}{180}) \quad (4.2)$$

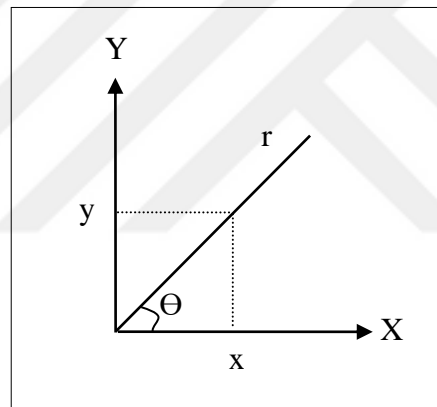


Figure 4.4 Coordinate system transformation

As given in Table 3.2 before, LIDAR's angular resolution with 50 Hz, is 0.5° and scan angle is 270° . So, for one LIDAR measurement we have 541 range values (r). These 541 data correspond to -45° to 225° with 0.5° interval. Since it is more easier to understand and examined data in Cartesian coordinates, transformation is applied and for a one LIDAR measurement, we have 541 (x,y) values of range.

One data set from transformed data is plotted in MatLab. This is shown in Figure 4.5.

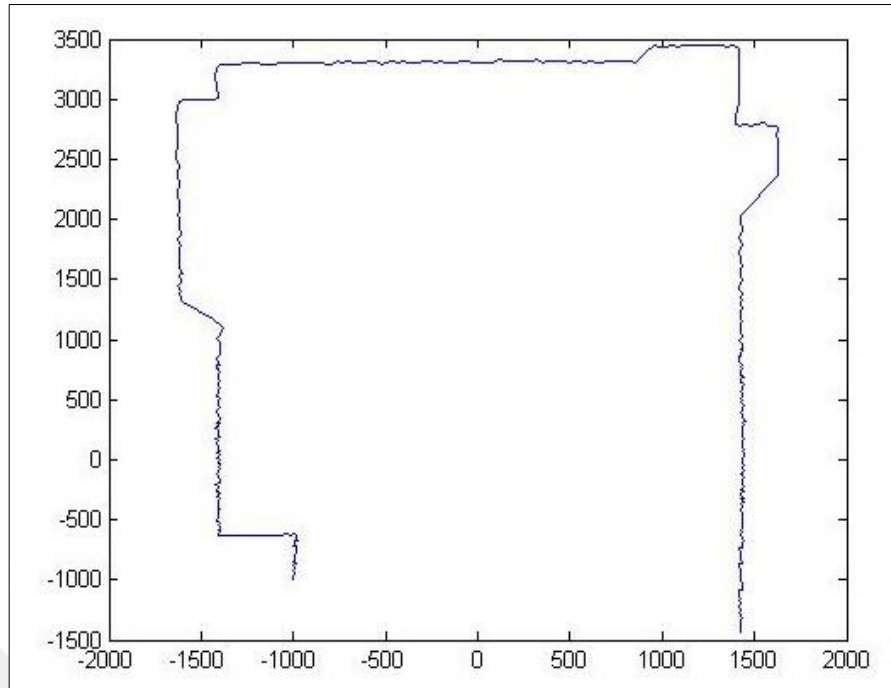


Figure 4.5 Basic map of the corridor from one data set

Statistical error means that when sensor performs several measurements, it will not output the same value every time. This is similar to when a measuring tape is used to measure two points several times, the result may not be exactly same value every time. For minimizing the statistical error, to obtaining several measurements together and build a mean value over these scans is the easiest solution.

LMS of SICK has a filter called a *mean filter*. When this mean filter is initiated, a number of scans can be adjusted. These scans are taken together and from those scans, a mean value from every scan degree over all the scans will be calculated. Then, one data string is given out with these mean values. This also implies that there is a reduction of data throughput, because only every 10th or 100th scan is provided. To eliminate the statistical error, a recommended number of scans is 100 (Alexander, 2015).

This formula is shown as following equations,

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (4.3)$$

where N is sampling quantity of scan; x is the range of a scan (mm); and \bar{x} is the average scan value (mm).

Root Mean Square Error (RMSE) which also known as Root Mean Square Deviation is one of the most commonly used statistics while considering accuracy. RMSE measures how much error there is between two sets of data. RMSE generally compares an observed value and a predicted value. Square of RMSE shows the *Quadratic Error* also known as *Mean Square error (MSE)*.

For our case, known measures as predicted value might be compared with LIDAR data as observed value. Root mean square error takes the difference for each known value and LIDAR value. After that, divide the sum of all values by the number of observations. This is how RMSE is calculated. Formulation of RMSE is given in (4.4),

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2} \quad (4.4)$$

where N is sampling quantity of scan; x_i is LIDAR scan data (mm); and \hat{x}_i is known measures (mm).

For calculating RMSE, an experimental stage is created. Figure 4.6 which is given below, shows how RMSE is calculated in the created stage. LIDAR is placed with a known distance from the wall. The known distance is measured before with both the ultrasonic range sensor and tape measure. LIDAR scan data are recorded and transferred to Excel framework. After the coordinate frame transformation is applied, sub data set from transformed Y values selected as a LIDAR data.

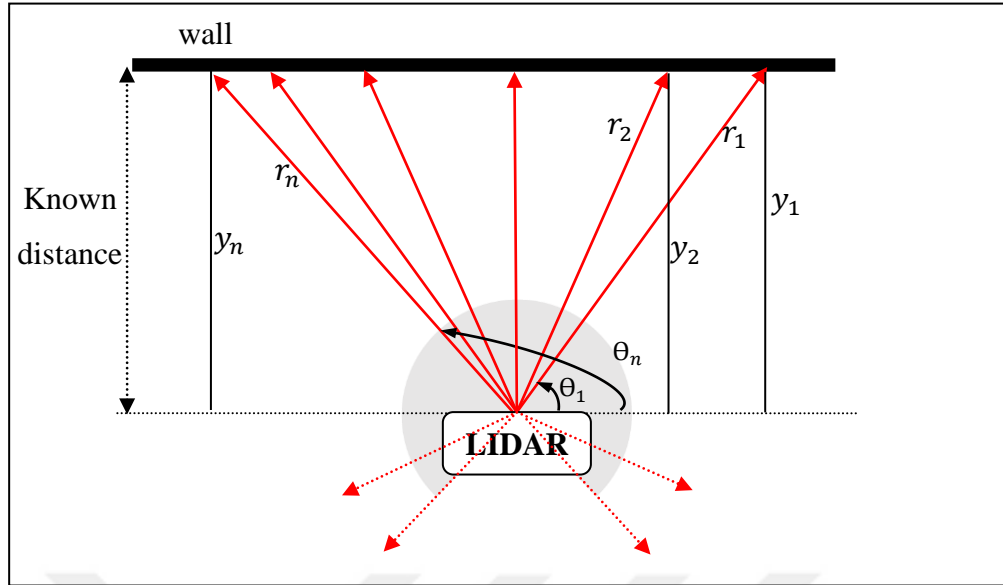


Figure 4.6 Calculating RMSE

Root mean square errors are calculated in MatLab by implementing the RMSE formula which is given in equation (4.4) above. Where θ_1 is 73.5° , θ_n is 114.5° and N is 83.

While recording LIDAR scan data, sampling rate is chosen as 1000 [ms]. The graph of RMSE is shown in Figure 4.7.

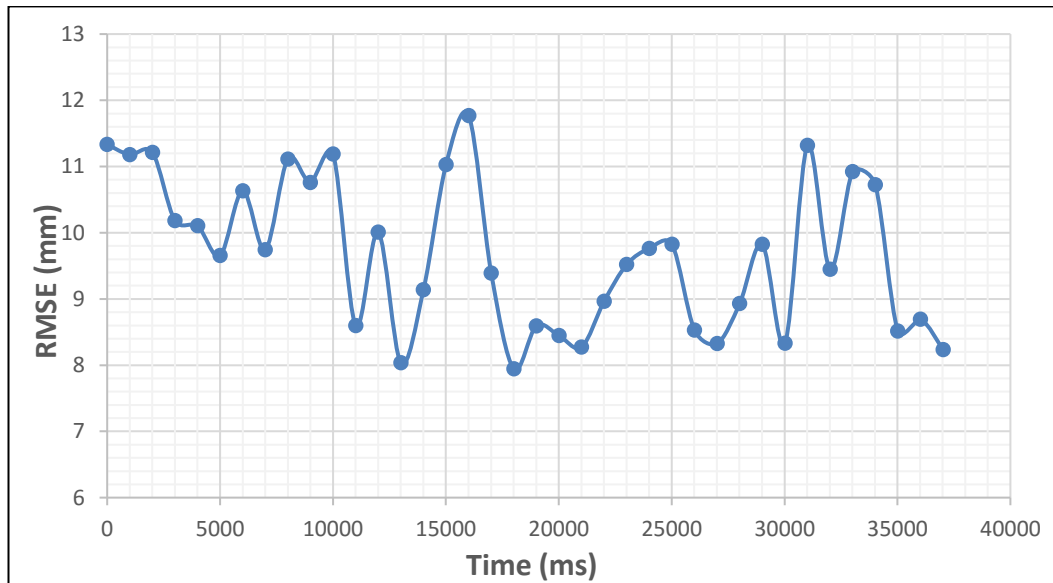


Figure 4.7 Root mean square error (RMSE) of LIDAR

Process is repeated with the mean filter of LMS100 activated. In the mean filter, scan number is chosen as 100. The results of RMSE graph is shown in Figure 4.8.

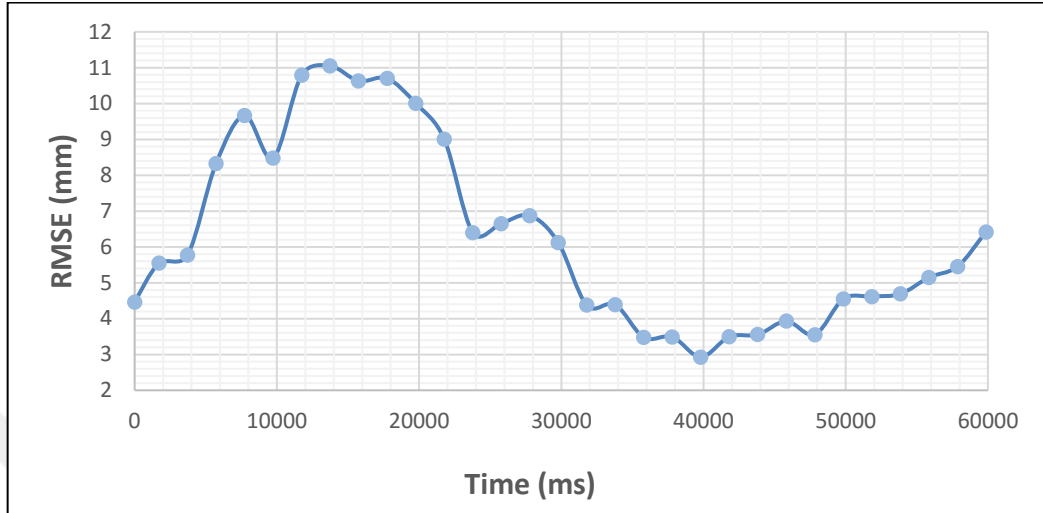


Figure 4.8 Root mean square error (RMSE) of LIDAR with own mean filter

As seen above, root mean square error of LIDAR changes in the range of 8-12 [mm] with a minimum value of 8 [mm] (Pls. see Fig. 4.7). When the mean filter is activated the root mean square error drops to minimum 3 [mm]. Root mean square error changes in the range of 3-11 [mm] (Pls. see Fig. 4.8). It is seen that, without mean filter average RMSE value of the sensor data is 10 [mm] whereas with mean filter activated average RMSE value of the sensor data decreases to 6 [mm]. On the other hand because of the mean filter process, sampling period increases up to twice of the unfiltered sampling period.

Robustness of the control depends on the reliability of the sensor measurement. Measurement quality should be increased and the error of the system should be decreased. The control scheme of the mobile robot is shown in the Figure 4.9.

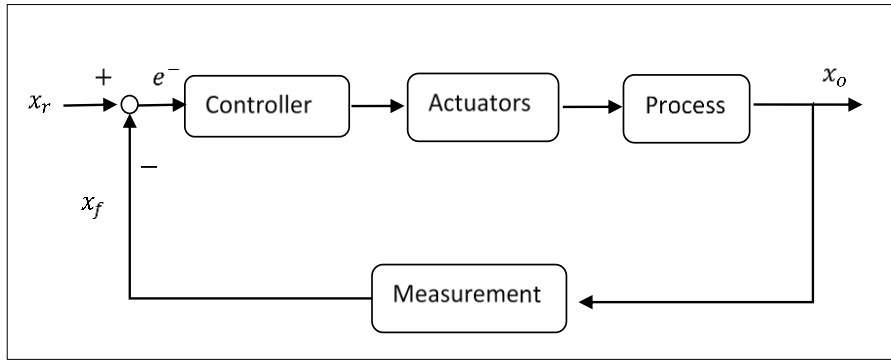


Figure 4.9 Control scheme of the mobile robot

Distance traveled by the mobile robot can be sensed by encoders, however this information is not reliable. For instance, wheels may slip on slippery surfaces so that mobile robot will travel less than the distance that encoders indicate. These errors are cumulative. They will grow while the mobile robot keeps moving. For eliminating those kind of errors, various sensors can be used depending on the working environment. Calculating error of the system resolution is shown in the Figure 4.10 below.

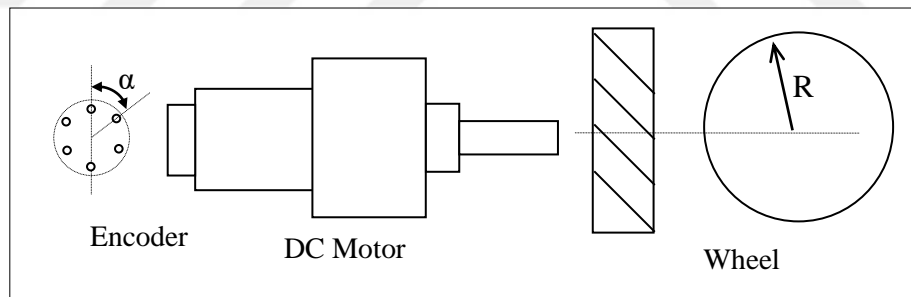


Figure 4.10 Calculating system resolution with DC motor

Gear ratio is 36, speed of dc motor is 102 RPM, α is 60° , diameter of the rear wheel, R is 60 [mm]. When a mobile robot is travelling for a one straight line, the system resolution error of the mobile robot is calculated as 1.75 [mm]. For the total traveled distance which is shown in the scenario below, system resolution error is calculated as 5.24 [mm].

In an outdoor environment includes grass has 0.32 optimal slip ratio. In a scenario where the mobile robot is travelling in three straight lines that makes totally 10 [m], the dead reckoning position error became 3.2 [m]. The scenario for calculating errors is shown in the Figure 4.11 (Kim & Lee, 2013).

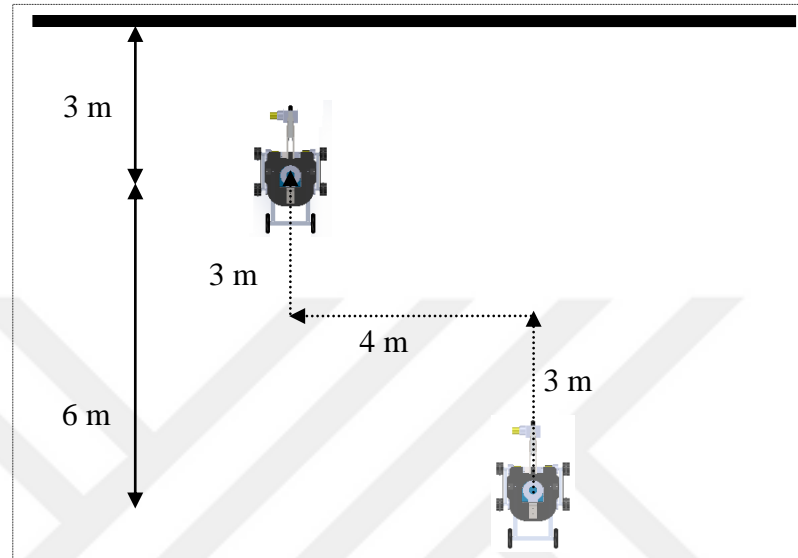


Figure 4.11 Scenario for calculating errors

Table 4.1 Errors of the mobile robot

	Error (mm)
Raw Data from LIDAR	+10 -26
LIDAR Data with Mean Filter Activated	-15
System Resolution	±1.75

Table 4.2 Errors in distance traveled in the scenario

Distance Traveled Error via Dead Reckoning (Encoder) Technique (mm)	Distance Traveled Error via LIDAR Data (mm)	Distance Traveled Error via Mean Filtered LIDAR Data (mm)
10000± (0.32*10000 ± 5.24)	6000±26	6000±15

In the Table 4.1, the errors from LIDAR, LIDAR with mean filter and system resolution are shown. In the Table 4.2 above, errors in the distance traveled in the scenario are shown. Horizontally distance is considered when calculating the distance travelled errors via LIDAR. As seen in the tables, without any external source like LIDAR, the error associated with odometry and encoder is excessive and grows unbounded as the mobile robot moves. So that some sort of external localization like LIDAR is required for accurate control of the mobile robot.

4.2 Mapping and Experimental Work in ROS Environment

This study builds upon the ROS. This open source environment is maintained by a large community of individuals and even companies. Content of ROS is highly adaptable, reusable and it's open to developments. This system provides us practicable solutions.

With ROS, a lot of application can be made such as simultaneous mapping and localization (SLAM), navigation, working with various sensors, face recognition and many other.

This section aims map building with two different algorithms with the help of the ROS.

4.2.1 Required Fundamental Knowledge for Creating System in ROS

Before starting to work with mobile robots in ROS and creating our system in it, there are some fundamental things to be understood thoroughly.

4.2.1.1 Coordinate Frames for Mobile Platforms

Map is a world fixed frame with its Z axis pointing upwards. The pose of a mobile platform, relative to the map frame, should not significantly drift over time. The map frame is not continuous, meaning the pose of a mobile platform in the map frame can

change in discrete jumps at any time. In a typical setup, a localization component constantly re-computes the robot pose in the map frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives. The map frame is useful as a long-term global reference, but discrete jumps make it a poor reference frame for local sensing and acting.

Odom is a world-fixed frame. The pose of a mobile platform in the *odom* frame can drift over time, without any bounds. This drift makes the *odom* frame useless as a long-term global reference. However, the pose of a robot in the *odom* frame is guaranteed to be continuous, meaning that the pose of a mobile platform in the *odom* frame always evolves in a smooth way, without discrete jumps. In a typical setup the *odom* frame is computed based on an odometry source, such as wheel odometry, visual odometry or an inertia measurement unit. The *odom* frame is useful as an accurate, short-term local reference, but drift makes it a poor frame for long-term reference.

base_link is rigidly attached to the mobile robot base. The *base_link* can be attached to the base in any arbitrary position or orientation; for every hardware platform there will be a different place on the base that provides an obvious point of reference (Meeussen, 2010).

4.2.1.2 Relationship between Frames

A tree representation is chosen to attach all coordinate frames in a robot system to each other. Therefore, each coordinate frame has one parent coordinate frame, and any number of child coordinate frames. The frames described are attached as follows:

map --> odom --> base_link

The map frame is the parent of odom, and odom is the parent of base_link. Although intuition would say that both map and odom should be attached to base_link, this is not allowed because each frame can only have one parent.

4.2.1.3 The Transformation Library (tf) and tf Transforms

According to Foote (2013), “The *tf* library was designed to provide a standard way to keep track of coordinate frames and transform data within the entire system.”

As robotic systems are becoming more complicated, being able to focus on the task frame and only the relevant coordinate frames become crucial. The *tf* library was developed as ROS package to provide this capability (Foote, 2013).

tf maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.

Broadcaster and Listener are the two standard modules which exist in the *tf* library. *Listening for transforms* module is responsible for receiving and buffering all coordinate frames that are broadcasted in the system and query for specific transforms between frames. *Broadcasting transforms* module is responsible for sending out the relative pose of coordinate frames to the rest of the system. A system can have many broadcasters that each provides information about a different part of the robot.

4.2.2 Mapping via Hector SLAM Algorithm

In this section, building map via hector slam algorithm is aimed. Before the experimental work, theory of hector slam is investigated and implementation is applied to our case.

4.2.2.1 Theory of Hector SLAM

Hector SLAM method is based on optimization of the alignment of beam endpoints with the map learnt so far. The basic idea using a Gauss-Newton approach was inspired by work in computer vision. The work was Lucas and Kanade's "An iterative image registration technique with an application to stereo vision" which issued in 1981 at DARPA Image Understanding Workshop. Using this approach, there is no need for a data association search between beam endpoints or an exhaustive pose search. As scans get aligned with the existing map, the matching is implicitly performed with all preceding scans (Kohlbrecher, von Oskar, Meyer & Klingauf, 2011; Lucas, 1981).

In Hector SLAM method, only filtering based on the endpoint z coordinate is used, so that only endpoints within a threshold of the intended scan plane are used in the scan matching proceeding. The following equations are taken from Kohlbrecher, von Oskar, Meyer & Klingauf for understanding the hector slam algorithm better.

Given a continuous map coordinate P_m , the occupancy value $M(P_M)$ as well as gradient $\nabla M(P_M) = \left(\frac{\partial M}{\partial x}(P_M), \frac{\partial M}{\partial y}(P_M) \right)$ can be approximated by using the four closest integer coordinates $P_{00..11}$. Linear interpolation along the x and y -axis then yields

$$\begin{aligned}
 M(P_m) \approx & \frac{y - y_0}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} M(P_{11}) + \frac{x_1 - x}{x_1 - x_0} M(P_{01}) \right) \\
 & + \frac{y_1 - y}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} M(P_{10}) + \frac{x_1 - x}{x_1 - x_0} M(P_{00}) \right)
 \end{aligned} \tag{4.5}$$

The derivatives can be approximated by:

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y - y_0}{y_1 - y_0} (M(P_{11}) + M(P_{01})) + \frac{y_1 - y}{y_1 - y_0} (M(P_{10}) + M(P_{00})) \tag{4.6}$$

$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_0}{x_1 - x_0} (M(P_{11}) + M(P_{10})) + \frac{x_1 - x}{x_1 - x_0} (M(P_{01}) + M(P_{00})) \quad (4.7)$$

This method pursues to find the rigid transformation $\xi = (p_x, p_y, \psi)^T$ that minimizes

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^n [1 - M(S_i(\xi))]^2 \quad (4.8)$$

that is, finding the transformation that gives the best alignment of the laser scan with the map is wanted. Here, $S_i(\xi)$ are the world coordinates of scan endpoint $s_i = (s_{i,x}, s_{i,y})^T$. They are a function of ξ , the pose of the robot in world coordinates:

$$S_i(\xi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \quad (4.9)$$

The function $M(S_i(\xi))$ returns the map value at the coordinates given by $S_i(\xi)$. Given some starting estimate of ξ , it is wanted to be estimated $\Delta\xi$ which optimizes the error measure according to

$$\sum_{i=1}^n [1 - M(S_i(\xi + \Delta\xi))]^2 \rightarrow 0. \quad (4.10)$$

By first order Taylor expansion of $M(S_i(\xi + \Delta\xi))$ we get:

$$\sum_{i=1}^n \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi \right]^2 \rightarrow 0. \quad (4.11)$$

This equation is minimized by setting the partial derivative with respect to $\Delta\xi$ to zero:

$$2 \sum_{i=1}^n \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta \xi \right] = 0 \quad (4.12)$$

Solving for $\Delta \xi$ yields the Gauss-Newton equation for the minimization problem:

$$\Delta \xi = H^{-1} \sum_{i=1}^n \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T [1 - M(S_i(\xi))] \quad (4.13)$$

with

$$H = \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \right] \quad (4.14)$$

An approximation for the map gradient $\nabla M(S_i(\xi))$ is provided in previous formulas. With equation (4.9) we get,

$$\frac{\partial S_i(\xi)}{\partial \xi} = \begin{pmatrix} 1 & 0 & -\sin(\psi)_{s_{i,x}} & -\cos(\psi)_{s_{i,y}} \\ 0 & 1 & \cos(\psi)_{s_{i,x}} & -\sin(\psi)_{s_{i,y}} \end{pmatrix} \quad (4.15)$$

Using $\nabla M(S_i(\xi))$ and $\frac{\partial S_i(\xi)}{\partial \xi}$, the Gauss-Newton equation (4.13) can now be evaluated, yielding a step $\Delta \xi$ towards the minimum.

Overall mapping process is shown in Figure 4.12.

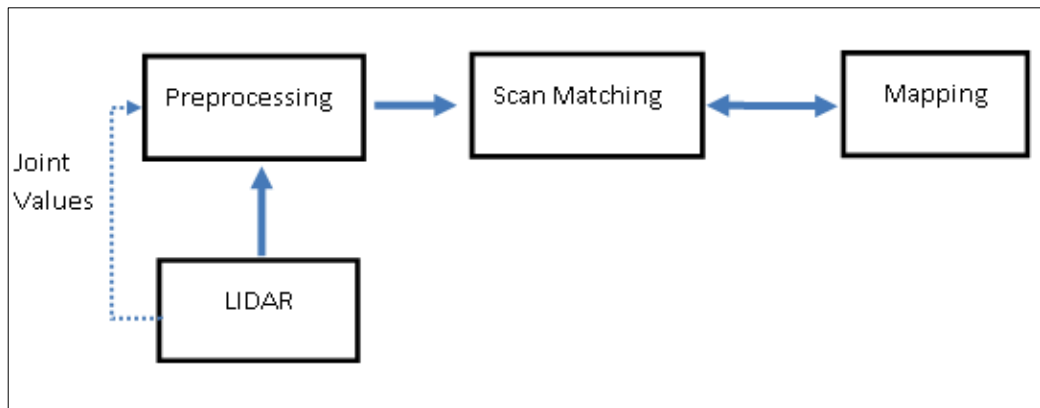


Figure 4.12 Overview of mapping system with hector_slam

All the coordinate frames which can be included in hector_slam are given in Figure 4.13.

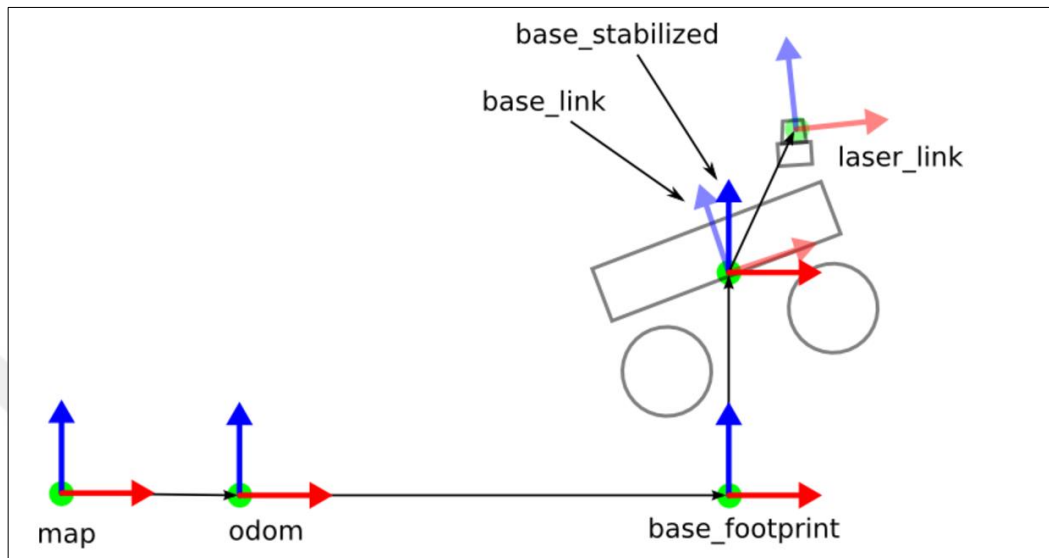


Figure 4.13 Original frame scheme of hector_slam

4.2.2.2 Implementation of Hector SLAM and Experimental Work

For adaptation of hector_slam algorithm to our study, modifications are made in hector_slam.

Odom frame is not supplied by our system, so odom is rearranged as base_link and also base_footprint is also rearranged as base_link. A transformation from map to base_link is directly published. Modified frame scheme of hector_slam for our study is shown in Figure 4.14.

```
<param name="map_frame" value="map" />
<param name="base_frame" value="base_link" />
<param name="odom_frame" value="base_link" />
```

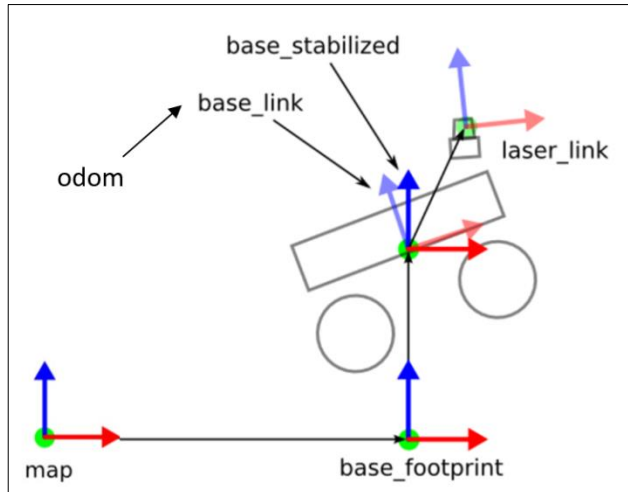


Figure 4.14 Modified frame scheme of hector_slam for our study

Main launch file is written and shown below. For sub-launch file which is hector_mapping_default.launch, Appendix-2 can be examined.

```

<launch>
  #### connect to lms100 LIDAR#####
  <param name="/use_sim_time" value="false"/>
  <node pkg="lmslxx" name="lmslxx" type="LMSlxx_node"
output="screen">
  <param name="host" value="169.254.10.200" />
  </node>
  #### run the visualizer: rviz #####
  <node pkg="rviz" type="rviz" name="rviz"
  args="-d $(find hector_slam_launch)/rviz_cfg/mapping_demo.rviz"/>
  #### publishing static transformation #####
  <node pkg="tf" type="static_transform_publisher"
name="map_baselink_broadcaster" args="0 0 0 0 0 0 map base_link
100"/>
  <node pkg="tf" type="static_transform_publisher"
name="baselink_laser_broadcaster" args="0 0 0 0 0 0 base_link laser
100"/>
  <include file="$(find bsk)/launch/hector_mapping_default.launch"/>
</launch>

```

After running the launch file, mapping building process is started. While map building process is proceeding, the system is checked whether everything is operating appropriately. For displaying the operation of the system, rqt_graph is used. rqt_graph is tool which creates a dynamic computation graph of what's going on in the system. Graph of our hector_slam implementation is shown in the Figure 4.15.

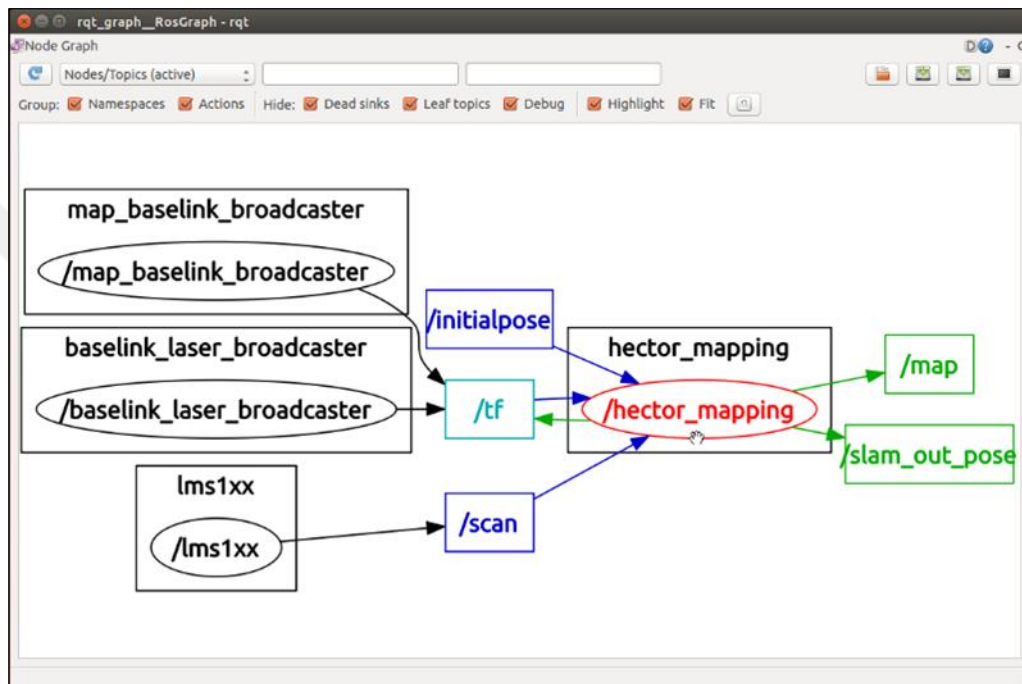


Figure 4.15 Graph of our hector_slam implementation

4.2.2.3 Mapping Results with Hector SLAM Algorithm

For first application, we built map of Automatic control laboratory's corridor. While map building process is proceeding, whenever the map is thought as enough, it can be saved with *map_saver* command-line utility of *map_server* tool. *Map_saver* allows dynamically generated maps to be saved to image file as *pgm* and data file as *yaml*.

Yaml format of the saved map is shown below.

```
image: hectormap1.pgm
resolution: 0.025000
origin: [-25.612499, -25.612499, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

where *image* is path the image file containing the occupancy data, *origin* is the 2-D pose of the lower left pixel in the map as (x, y, yaw) with yaw as counterclockwise rotation (yaw=0 means no rotation), *occupied_thresh* is pixels with occupancy probability greater than this threshold are considered completely occupied, *free_thresh* is pixels with occupancy probability less than this threshold are considered completely free, *negate* is whether the white/black free/occupied semantics should be reversed. *Resolution* points out there solution of the built map in meters/pixel, it also corresponds to meters/cell.

The built map of laboratory corridor is shown in Figure 4.16.

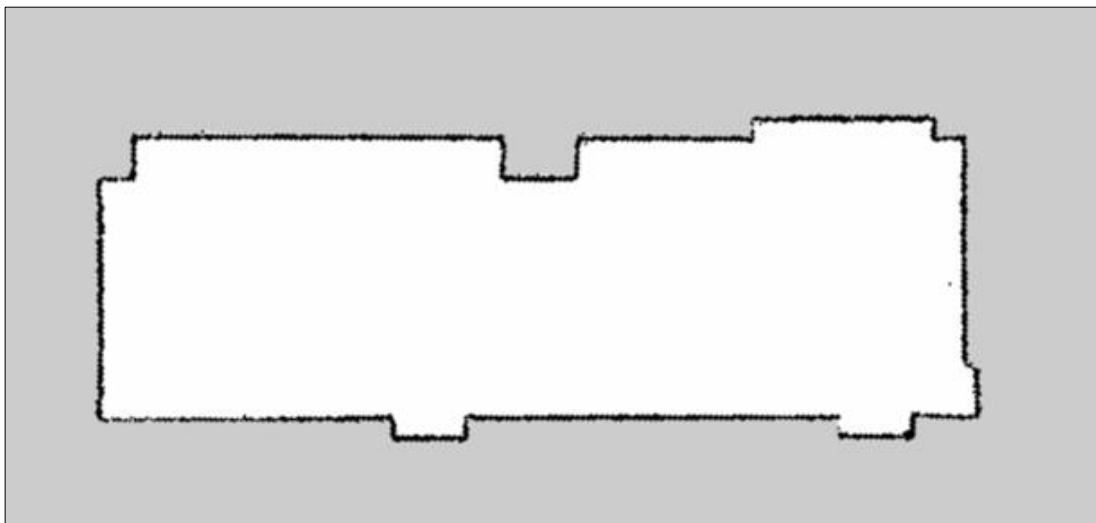


Figure 4.16 Built map of laboratory corridor

In a map building process, the desirable measurements or distances can be estimated by the algorithm. With help of the visualizer interface rviz in ROS, measurements are shown in Figure 4.17. All the shown distances are from real time sensor measurements.

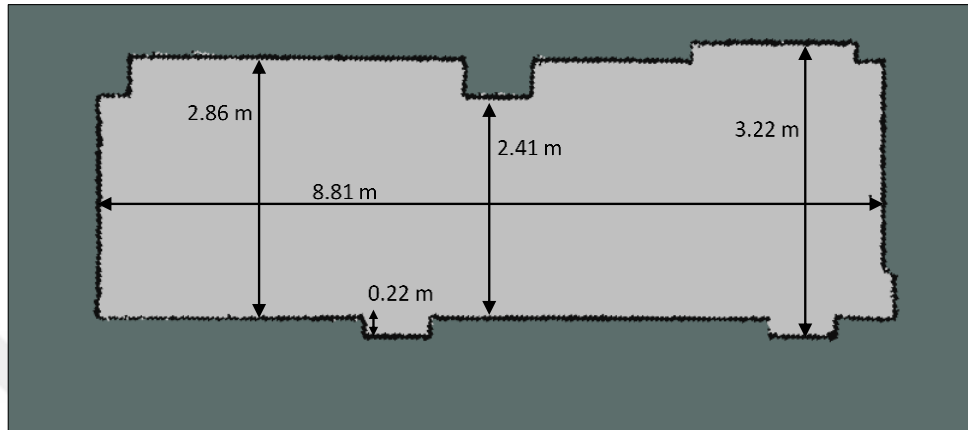


Figure 4.17 Measurements in the built map

While the map building process is continuing, the door is opened. As can be seen in Figure 4.18, laser beams escaped from doorway. While the mobile robot is not exploring the outside of the area which in it, robot cannot know what is in the outside of the explored area.

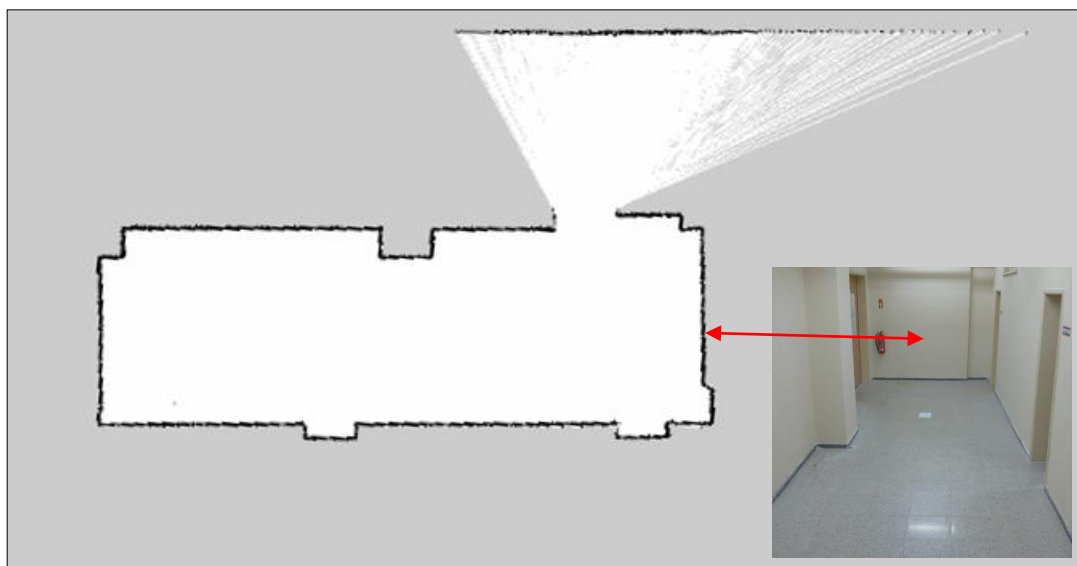


Figure 4.18 Escape of laser beams while mapping of laboratory corridor

After passing the doorway to explore the main corridor, now mobile robot could build the map of the new area. Built map can be seen in the Figure 4.19.

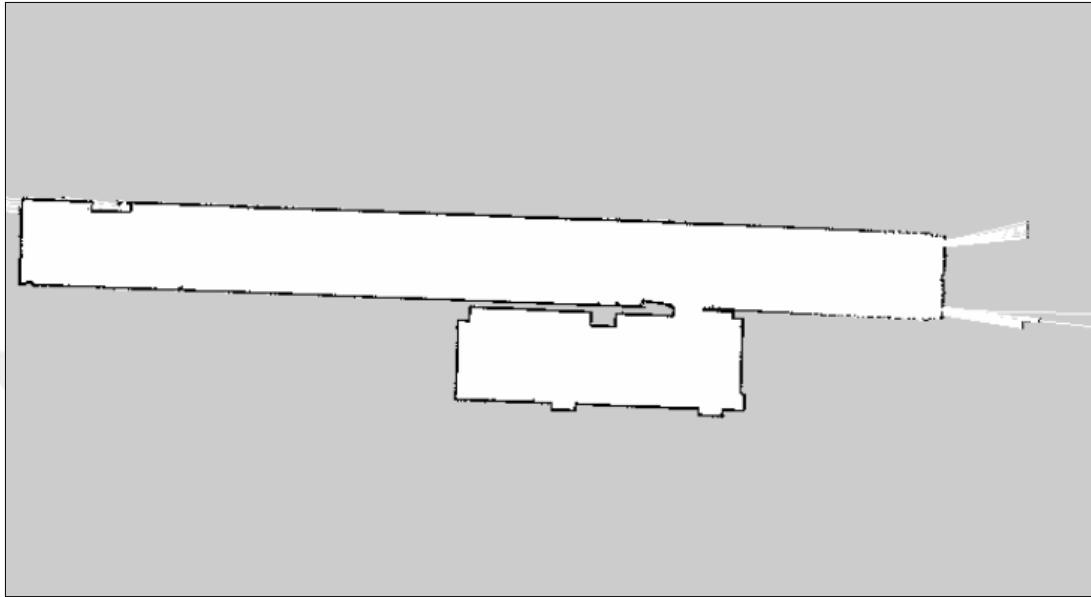


Figure 4.19 Built map of laboratory corridor and the main corridor

Map of corridors is smooth since they are not including numerous objects in it. But in real life scenarios, mobile robots can be required to evaluate complex indoor environments.

For the complex environments just like the Automatic Control Laboratory of Mechanical Engineering Department, DEU, the mapping results are not smooth as the empty corridor. It is shown in Figure 4.20. Also it can be seen that the laser beams escapes from windows.



Figure 4.20 Built map of DEU Automatic Control Laboratory and its corridor

4.2.3 Mapping via Gmapping Algorithm

In this section, building map via gmapping algorithm is aimed. Before the experimental work, theory of gmapping is investigated and implementation is applied to our case.

4.2.2.1 Theory of Gmapping

Gmapping is the most popular and publicly known simultaneous localization and mapping algorithm which was developed by Giorgio Grisetti, Wolfram Burgard and Cyrill Stachniss. Gmapping method is using Rao-Blackwellized particle filter to keep track of the likely positions of the robot, based on its sensor data and the parts of the map that have already been built.

Each particle = sample of history of robot poses + posterior over maps given the sample pose history; approximate posterior over maps by distribution with all probability mass on the most likely map whenever posterior is needed.

The main idea of Rao-Blackwellized particle filter for SLAM is to estimate a posterior $p(x_{1:t}, m | z_{1:t}, u_{1:t-1})$ about $x_{1:t}$ potential trajectories of the mobile robot given its observations $z_{1:t}$ and its odometry measurements $u_{1:t-1}$, for using this posterior to calculate a posterior over maps and trajectories:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) = p(m | x_{1:t}, z_{1:t})p(x_{1:t} | z_{1:t}, u_{1:t-1}) \quad (4.16)$$

Rao-Blackwellized sampling importance resampling filter for mapping incrementally processes the sensor observations and the odometry readings as they are available. It updates the set of samples that represents the posterior about the map and the trajectory of the vehicle. The process can be summarized by the following four steps:

- 1) *Sampling*: The next generation of particles $\{x_t^{(i)}\}$ is obtained from the generation $x_{t-1}^{(i)}$ by sampling from the proposal distribution π . Often, a probabilistic odometry motion model is used as the proposal distribution.
- 2) *Importance Weighting*: An individual importance weight $w_t^{(i)}$ is assigned to each particle according to the importance sampling principle

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})} \quad (4.17)$$

The weights account for the fact that the proposal distribution π is in general not equal to the target distribution of successor states.

- 3) *Resampling*: Particles are drawn with replacement proportional to their importance weight. This step is necessary since only a finite number of particles is used to approximate a continuous distribution. Furthermore, resampling allows us to apply a particle filter in situations in which the target

distribution differs from the proposal. After resampling, all the particles have the same weight.

- 4) *Map Estimation:* For each particle, the corresponding map estimate $p(m_t^{(i)} | x_{1:t}^{(i)}, z_{1:t})$ is computed based on the trajectory $x_{1:t}^{(i)}$ of that sample and the history of observations $z_{1:t}$

The implementation of this schema requires to evaluate the weights of the trajectories from scratch whenever a new observation is available. We obtain a recursive formulation to compute the importance weights by restricting the proposal π to fulfill the following assumption

$$\pi(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1}) = \pi(x_t | x_{1:t-1}, z_{1:t}, u_{1:t-1}) \cdot \pi(x_{1:t-1} | z_{1:t-1}, u_{1:t-2}) \quad (4.18)$$

Based on Equations (4.17) and (4.18), the weights are computed as

$$w_t^{(i)} = \frac{p(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})} \quad (4.19)$$

$$= \frac{\eta p(z_t | x_{1:t}^{(i)}, z_{1:t-1}) p(x_t^{(i)} | x_{1:t-1}^{(i)}, u_{t-1})}{\pi(x_t^{(i)} | x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t-1})} \quad (4.20)$$

$$\cdot w_t^{(i)} = \frac{p(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{(i)} | z_{1:t}, u_{1:t-1})}$$

$$\propto \frac{p(z_t | m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)}, u_{t-1})}{\pi(x_t | x_{1:t-1}^{(i)}, z_{1:t}, u_{1:t-1})} \cdot w_{t-1}^{(i)} \quad (4.21)$$

Here $\eta = 1/p(z_t|z_{1:t-1}, u_{1:t-1})$ is a normalization factor resulting from Bayes' rule that is equal for all particles. Most of the existing particle filter applications rely on the recursive structure of Eq. (4.21).

The importance weights are then computed according to the observation model $p(z_t|m, x_t)$. This becomes clear by replacing π in Eq. (4.21) by the motion model $p(x_t|x_{t-1}, u_{t-1})$

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})}{p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})} \quad (4.22)$$

$$\propto w_{t-1}^{(i)} \cdot p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) \quad (4.23)$$

This proposal distribution, however, is suboptimal especially when the sensor information is significantly more precise than the motion estimate of the robot based on the odometry, which is typically the case if a robot equipped with a laser range finder.

The most recent sensor observation z_t when generating the next generation of samples. By integrating z_t into the proposal one can focus the sampling on the meaningful regions of the observation likelihood. According to Doucet (1998), the distribution

$$p(x_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t|m_{t-1}^{(i)}, x_t) p(x_t|x_{t-1}^{(i)}, u_{t-1})}{p(z_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \quad (4.24)$$

is the optimal proposal distribution with respect to the variance of the particle weights. Using that proposal, the computation of the weights turns into

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{\eta p(z_t|m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)}, u_{t-1})}{p(x_t|m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} \quad (4.25)$$

$$\propto w_{t-1}^{(i)} \frac{p(z_t | m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t | m_{t-1}^{(i)}, x_t) p(x_t | x_{t-1}^{(i)}, u_{t-1})}{p(z_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} \quad (4.26)$$

$$= w_{t-1}^{(i)} \cdot p(z_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) \quad (4.27)$$

$$= w_{t-1}^{(i)} \cdot \int p(z_t | x') p(x' | x_{t-1}^{(i)}, u_{t-1}) dx' \quad (4.28)$$

$$L^{(i)} = \{x | p(z_t | m_{t-1}^{(i)}, x) > \epsilon\} \quad (4.29)$$

The Gaussian parameters are estimated as

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^K x_j \cdot p(z_t | m_{t-1}^{(i)}, x_j) \cdot p(x_j | x_{t-1}^{(i)}, u_{t-1}) \quad (4.30)$$

$$\Sigma_t^{(i)} = \frac{1}{\eta^{(i)}} \cdot \sum_{j=1}^K p(z_t | m_{t-1}^{(i)}, x_j) \cdot p(x_j | x_{t-1}^{(i)}, u_{t-1}) \cdot (x_j - \mu_t^{(i)})(x_j - \mu_t^{(i)})^T \quad (4.31)$$

with the normalization factor

$$\eta^{(i)} = \sum_{j=1}^K p(z_t | m_{t-1}^{(i)}, x_j) \cdot p(x_j | x_{t-1}^{(i)}, u_{t-1}) \quad (4.32)$$

In this way, we obtain a closed form approximation of the optimal proposal which enables us to efficiently obtain the next generation of particles. Using this proposal distribution, the weights can be computed as

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \cdot p(z_t | m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) \\ &= w_{t-1}^{(i)} \cdot \int p(z_t | x') p(x' | x_{t-1}^{(i)}, u_{t-1}) dx' \\ &\cong w_{t-1}^{(i)} \cdot \sum_{j=1}^K p(z_t | m_{t-1}^{(i)}, x_j) \cdot p(x_j | x_{t-1}^{(i)}, u_{t-1}) \end{aligned} \quad (4.33)$$

$$= w_{t-1}^{(i)} \cdot \eta^{(i)} \quad (4.34)$$

We compute this quantity according to the formulation of Doucet et al. (2001) as

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w^{(i)})^2}, \quad (4.35)$$

where $w^{(i)}$ refers to the normalized weight of particle i .

The overall process is summarized in Algorithm: Map Learning with RBPF below. The algorithm is taken from the works of Giorgia Grisetti, Wolfram Burgard and Cyrill Stachniss.

Algorithm: Map Learning with RBPF

Require:

S_{t-1} , the sample set of the previous time step
 z_t , the most recent laser scan
 u_{t-1} , the most recent odometry measurement

Ensure:

S_t , the new sample set
 $S_t = \{ \}$

```

for all  $S_{t-1}^{(i)} \in S_{t-1}$  do
   $\langle x_{t-1}^{(i)}, w_{t-1}^{(i)}, m_{t-1}^{(i)} \rangle = S_{t-1}^{(i)}$ 
  // scan-matching
   $x_{t-1}'^{(i)} = x_{t-1}^{(i)} \oplus u_{t-1}$ 
   $\hat{x}_t^{(i)} = \operatorname{argmax}_x p(x_t | m_{t-1}^{(i)}, z_t, x_{t-1}'^{(i)})$ 
  if  $\hat{x}_t^{(i)} = \text{failure}$  then
     $x_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$ 

```

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot p(z_t | m_{t-1}^{(i)}, x_t^{(i)})$$

Else

// sample around the mode

for $k = 1, \dots, K$ **do**

$x_k \sim \{x_j \mid |x_j - \hat{x}^{(i)}| < \Delta\}$

end for

// compute Gaussian proposal

$$\mu_t^{(i)} = (0, 0, 0)^T$$

$$\eta^{(i)} = 0$$

for all $x_j \in \{x_1, \dots, x_k\}$ **do**

$$\mu_t^{(i)} = \mu_t^{(i)} + x_j \cdot p(z_t | m_{t-1}^{(i)}, x_j) \cdot$$

$$p(x_t | x_{t-1}^{(i)}, u_{t-1})$$

$$\eta^{(i)} = \eta^{(i)} + p(z_t | m_{t-1}^{(i)}, x_j) \cdot$$

$$p(x_t | x_{t-1}^{(i)}, u_{t-1})$$

end for

$$\mu_t^{(i)} = \mu_t^{(i)} / \eta^{(i)}$$


```

 $\Sigma^{(i)}_t = 0$  // update sample set
for all  $x_j \in \{x_1, \dots, x_K\}$  do
     $\Sigma^{(i)}_t = \Sigma^{(i)}_t + (x_j - \mu^{(i)}_t)(x_j - \mu^{(i)}_t)^T \cdot p(z_t | m_{t-1}^{(i)}, x_j^{(i)}) \cdot p(z_t | x_{t-1}^{(i)}, u_{t-1})$ 
     $S_t = S_t \cup \{ \langle x_t^{(i)}, w_t^{(i)}, m_t^{(i)} \rangle \}$ 
end for
if  $N_{eff} < T$  then
     $N_{eff} = \frac{1}{\sum_{i=1}^N (w^{(i)})^2}$ 
     $S_t = \text{resample}(S_t)$ 
end if

end for
 $\Sigma^{(i)}_t = \Sigma^{(i)}_t / \eta^{(i)}$ 

// sample new pose
 $x_t^{(i)} \sim \mathcal{N}(\mu_t^{(i)}, \Sigma_t^{(i)})$ 

// update importance weights
 $w_t^{(i)} = w_{t-1}^{(i)} \cdot \eta^{(i)}$ 
end if

// update map
 $m_t^{(i)} = \text{integrateScan}(m_{t-1}^{(i)}, x_t^{(i)}, z_t)$ 

```

4.2.2.2 Implementation of Gmapping and Experimental Work

Before implementing gmapping for our case, requirements of gmapping should be considered. Gmapping has some crucial requirements to fulfill; a horizontally-mounted, fixed LIDAR and something to provides odometry data. Unlike the Hector_slam algorithm, gmapping cannot tractable without odometry data. Since we don't have reliable odometry, another solution is applied. With combination gmapping and laser scan matcher, SLAM can be implied. Laser scan matcher process supplies the odometry data for gmapping.

Laser scan matcher allows to scan match between consecutive sensor_msgs/LaserScan messages and publish the estimated position of the laser as a geometry_msgs/Pose2D or a tf transform.

The package can be used without any odometry estimation supplied by other sensors. Therefore, this package can serve as a stand alone odometry estimator.

Alternatively, you can provide several types of odometry input to improve the registration speed and accuracy. The `laser_scan_matcher` can operate using `sensor_msgs/LaserScan` messages or `sensor_msgs/PointCloud2` messages. In our case, `LaserScan` messages are used.

In the classical frame-to-frame laser odometry, each laser scan is compared to the previous scan. The transformation between the two is aggregated over time to calculate the position of the robot in the fixed frame. Some noise in the scans is inevitable. Thus, even for a robot standing still, the incremental transformations might be non-zero. This could result in a slow drift of the pose of the robot. To alleviate this, laser scan matcher implement keyframe-based matching. The change in pose is calculated between the current laser scan and a "keyframe" scan. The keyframe scan is updated after the robot moves a certain distance. Thus, if the robot is standing still, the keyframe scan will not change, and the pose will remain more drift free.

After understanding requirements of gmapping, a proper launch file is written. The launch file of our gmapping implementation is given below.

```
<launch>
  #### connect to lms100 LIDAR #####
  <param name="/use_sim_time" value="false"/>
  <node pkg="lms1xx" name="lms1xx" type="LMS1xx_node">
    <param name="host" value="169.254.10.200" />
  </node>

  #### publish base_link -> laser transform #####
  <node pkg="tf" type="static_transform_publisher"
name="base_link_to_laser"
  args="0.0 0.0 0.0 0.0 0.0 0.0 /base_link /laser 40" />

  #### start rviz #####
  <node pkg="rviz" type="rviz" name="rviz"
  args="-d $(find laser_scan_matcher)/demo/demo_gmapping.rviz"/>
```

```

##### start the laser_scan_matcher #####
<node pkg="laser_scan_matcher" type="laser_scan_matcher_node"
  name="laser_scan_matcher_node" output="screen">
  <param name="fixed_frame" value = "odom"/>
  <param name="max_iterations" value="10"/>
</node>

##### start gmapping #####
<node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"
output="screen">
<roscpp command="load" file="$(find
bsk)/launch/gmapping_params.yaml" />
</node>
</launch>

```

Parameters of gmapping is written in another file which is gmapping_params.yaml. YAML is a lightweight markup language that supports all parameter types. It is a human-readable data serialization language that takes concepts from programming languages such as C and Python, and ideas from XML and the data format of electronic mail. Configuration can be saved in YAML files. For further detail and parameters of gmapping which is in gmapping_params.yaml file, Appendix-3 can be examined.

After running the launch file, mapping building process is started. While map building process is proceeding, the system is checked with rqt_graph. Graph of our gmapping implementation is shown in the Figure 4.21.

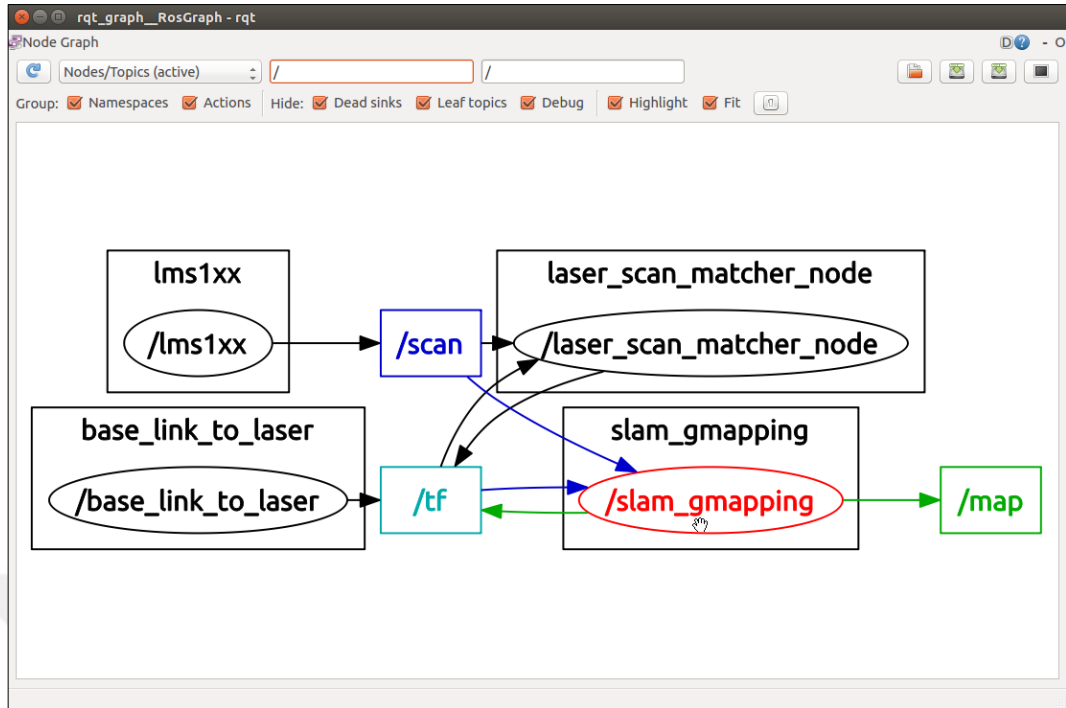


Figure 4.21 Graph of our gmapping implementation

4.2.3.3 Mapping Results with Gmapping

Map building initialized and the mobile robot is just at the beginning of the exploration the environment. Early stage of map building process is shown in Figure 4.22.

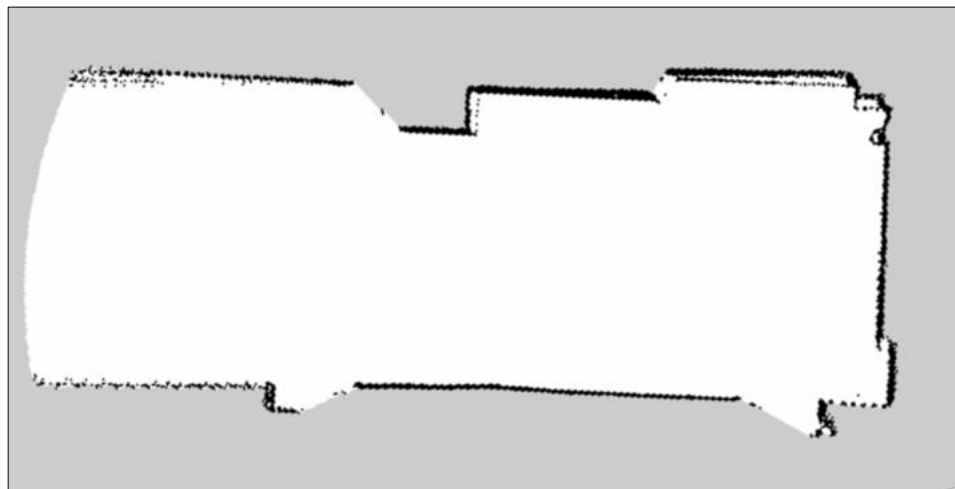


Figure 4.22 Early stage of map building process

As previously done in hector SLAM, map of Automatic Control Laboratory's corridor is built with gmapping algorithm this time. It is shown in Figure 4.23.

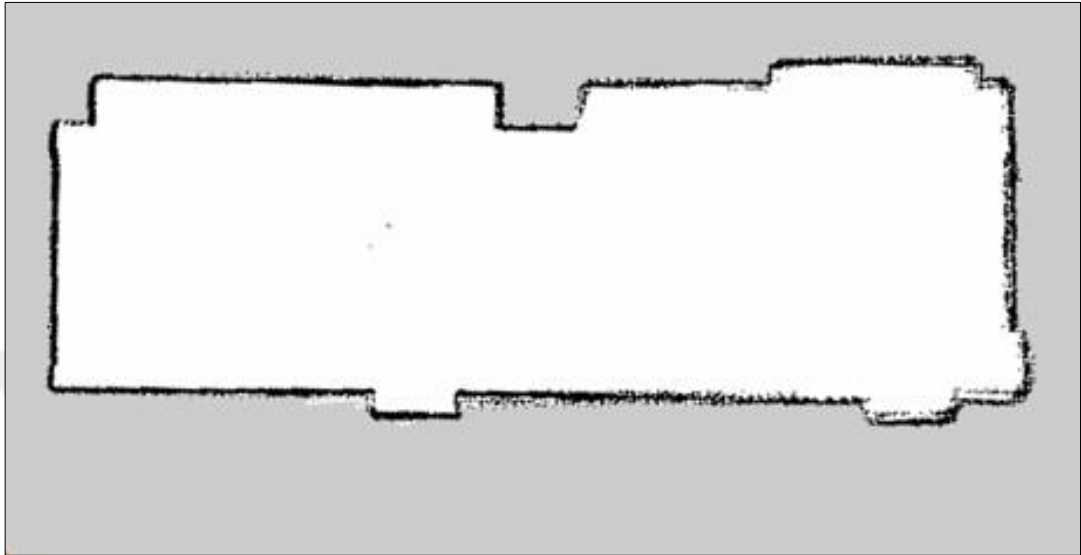


Figure 4.23 Built map of laboratory corridor

After the laboratory corridor, map of Automatic Control Laboratory is built. The map is successful but as seen in Figure 4.24, there is an irregularity that stands out. The shown axis is dislocated. Drifts like this are happened when mobile robot is turning. If it's not that obvious dislocation like this example, scan match algorithm of Gmapping can fix the minor dislocation problems while map is still building.



Figure 4.24 Built map of the Automatic Control Laboratory

Built map of the Automatic Control Laboratory and its corridor is shown in the Figure 4.25.

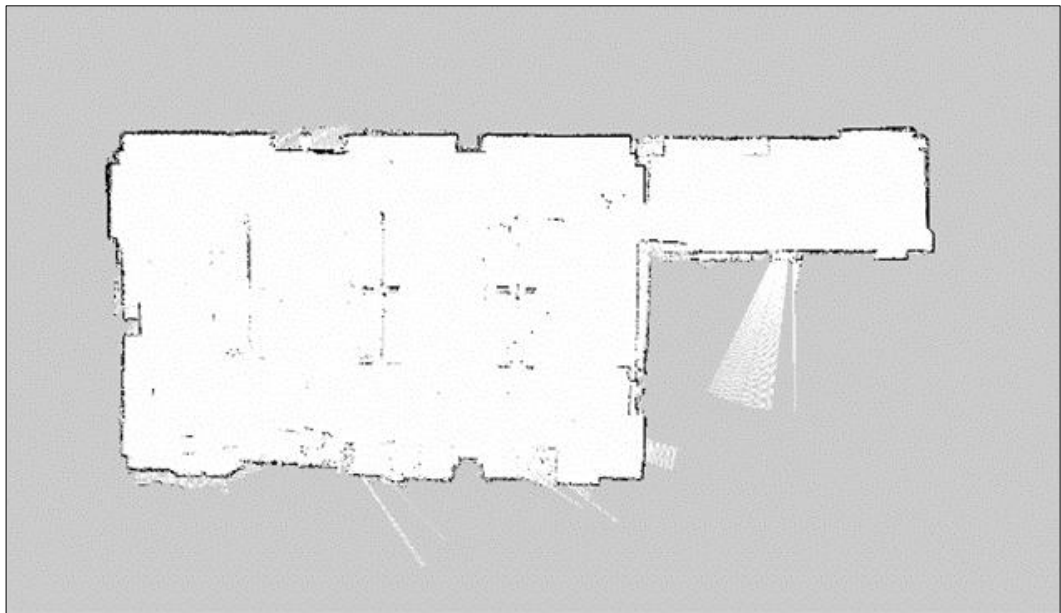


Figure 4.25 Built map of the Automatic Control Laboratory and its corridor

4.2.4 Comparison of Gmapping and Hector SLAM for Mapping

Both algorithms are well known and widely used. Advantages and disadvantages are always existed. The important point is that choosing most suitable one for the system. This choice depends on what our system and environment consists of. For instance, if the robotic system has not a reliable odometry data, hector slam should be considered, on contrary the robotic system has a reliable odometry data most well-known gmapping can be used. As seen in the Figure 4.26, both algorithms have sufficient results in modeling the environment for mobile robots.

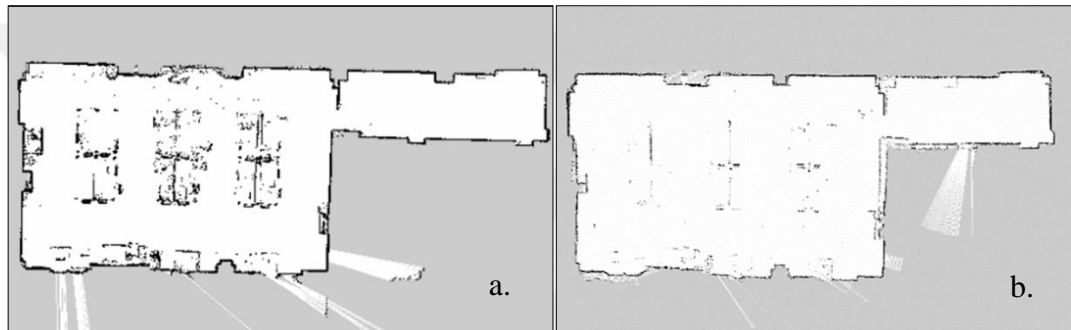


Figure 4.26 Comparison of hector slam and gmapping: a) hector slam b) gmapping

4.3 Object Detection & Costmaps

After getting reliable LIDAR data and building map successfully in previous sections, lastly we will focus on obstacle detection and costmaps in this part of the study.

For a successful navigation process in autonomous systems, path planning is a crucial process to accomplish. Path planning process includes obstacle avoidance task. Generally, both in indoor or outdoor environment, obstacles are existed. While mobile robots are fulfilling the given tasks, obstacles in the environment are required to avoid or overcome.

As mentioned before, object detection is the key part of the obstacle avoidance for creating tractable path planning. Object detection systems are able to detect most of the obstacles by looking for signs such as gradient of range.

In this study, for detection, range-based detection specifically LIDAR-based detection approach was implied. Detection algorithm for objects, employs LIDAR that mounted in mobile robot to evaluate the mobile robot's environment. For mobile robots, traversability is essential term to understand. This term indicates passable area of the mobile robot's surroundings. Traversability can be represented via costmaps that is computed. A representation is shown in Figure 4.27.

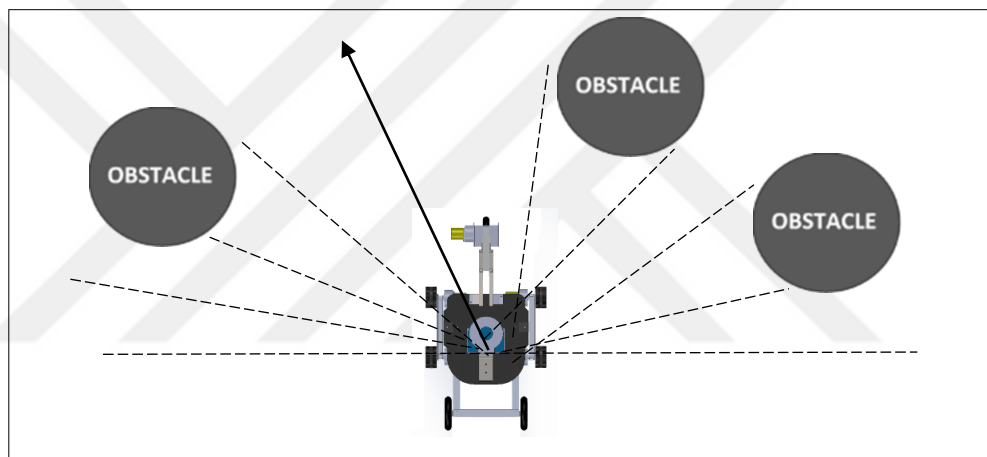


Figure 4.27 Representing mobile robot in an environment with obstacles

4.3.1 Theory of Costmap

Costmap is a two dimensional grid that represents the environment of the robot, where each cell has a cost for traveling through that area of the environment. In its simplest form, the costmap only has two values in it: a high value that represents cells which the robot cannot be in, and a low value for cells which the robot can be in. Since the high values will result in a collision between the robot and an obstacle, such values are often called lethal in the costmap. Throughout this dissertation, we will discuss how the values in the costmap are calculated and additional schemes for what the values can represent.

Costmap includes information about objects/obstacles. It consists of grid cells where each grid cell has a value $f(x, y)$. Values above some predefined threshold are designated as *lethal* values, representing states that must be avoided like those resulting in collisions (Lu, 2014).

Focusing on the frequently employed two dimensional Gaussian distribution, defined as

$$f(x, y) = A \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \quad (4.36)$$

The cost in each cell under the Gaussian depends on both the amplitude, A , and the variance, σ .

There are at least three different pieces of knowledge that the robot needs to be able to store in a cell:

1. No knowledge corresponding to unknown
2. Obstacle corresponding to lethal cell
3. No obstacle corresponding to free space

These cell types are shown in Figure 4.28. Gray cells are representing unknown spaces, black cells are representing lethal spaces and white cells are representing free space.

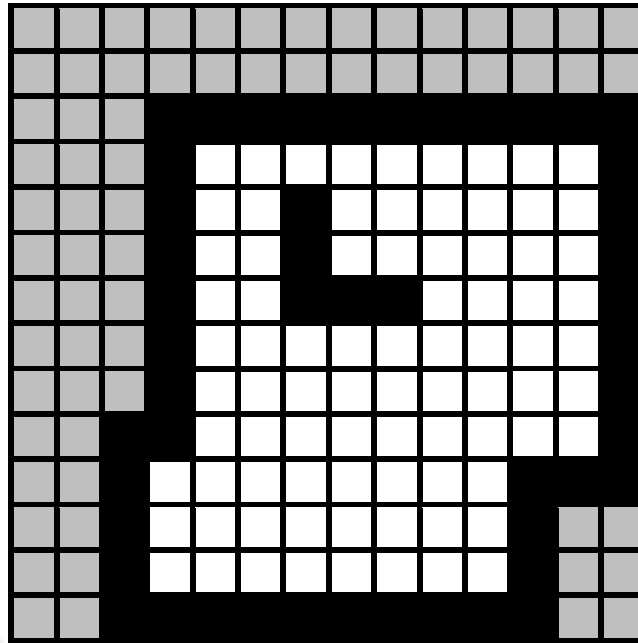


Figure 4.28 Cell types

In general, the planning algorithms need to avoid collisions between the robot and the lethal cells, primarily by staying in free space.

Layered Costmaps: Instead of storing obstacles data directly in the grid, the layered costmap maintains an ordered list of layers. These layers track the data related to a specific functionality. The data for each of the layers is then assembled into the master costmap in two distinct passes.

In the first pass, the *updateBounds* method, every individual layer is polled to conclude how much of the costmap it needs to update. The layers are iterated over, in order, providing each layer with the bounding box that the previous layers need to update. Every individual layer can expand the bounding box as necessary. This first pass results in a bounding box that determines how much of the master costmap needs to be updated.

During the second pass, the *updateValues* method is called, during which each consecutive layer will update the values within the master costmap's area of

bounding box. Figure 4.29 shows the update algorithm using a set of layers that replicate the behavior of a basic monolithic costmap (Lu, 2014).

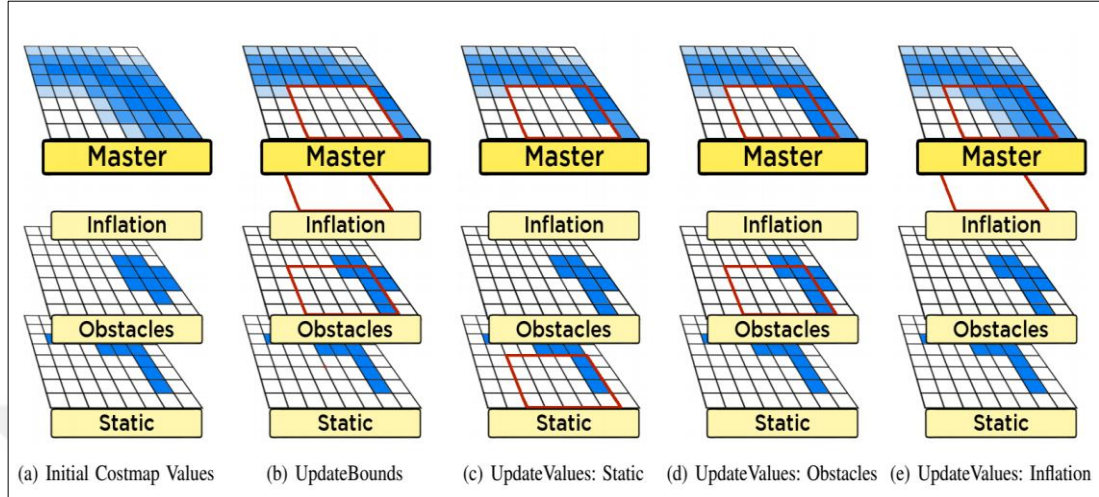


Figure 4.29 Update algorithm Costmap layers (Lu, 2014)

The cost function is computed as follows for all cells in the costmap further than the inscribed radius distance and closer than the inflation radius distance away from an actual obstacle:

$$e^{-csf*(X-R_{inscribed})} * (\text{costmap_2d}::\text{INSCRIBED_INFLATED_OBSTACLE} - 1) \quad (4.37)$$

where `costmap_2d::INSCRIBED_INFLATED_OBSTACLE` is currently 254, X is distance from obstacle (mm), $R_{inscribed}$ is the inscribed radius of the mobile robot (mm), csf is the cost scaling factor. It is shown in Figure 4.30.

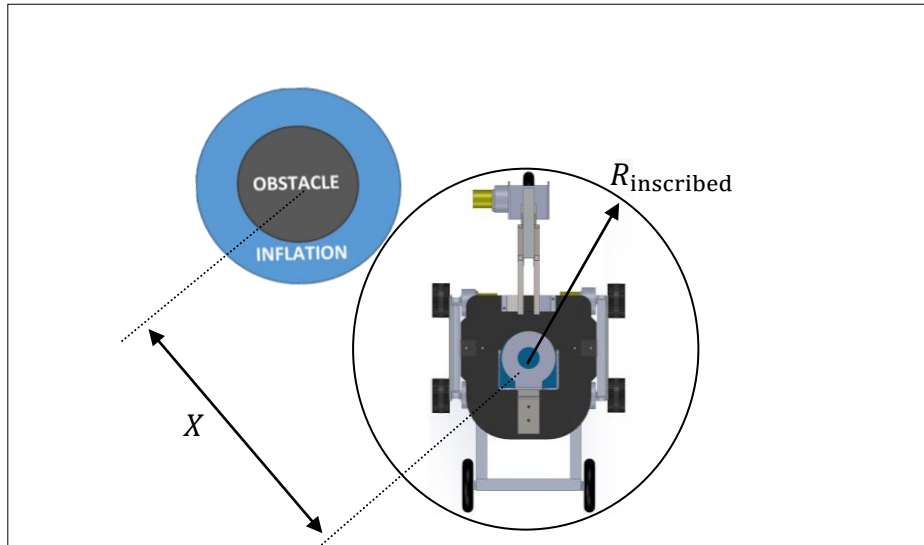


Figure 4.30 Mobile robot and obstacle interaction

4.3.2 Implementation of Costmap_2d and Experimental Work

Costmap_2d package supplies an implementation of a 2D costmap that takes in sensor data from LIDAR and creates a 2D occupancy grid of the data and inflates costs in a 2D costmap based on the occupancy grid and adjustable inflation radius. Costmap_2d also provides support for map_server based initialization of a costmap, rolling window based costmaps, and parameter based subscription to and configuration of sensor topics.

The costmap automatically subscribes to sensors topics and updates itself accordingly. Each sensor is used to either mark, clear or both. *Marking* means inserting obstacle information into the costmap and *clearing* means remove obstacle information from the costmap. The clearing operation, performs raytracing through the grid from the origin of the sensor outwards for each observation reported.

For implementing costmap_2d which is a layered costmap as mentioned before, a few requirements should be met. These requirements are range sensor data and the static map, which correspond to our case as LIDAR data and the built or building map from hector_slam or gmapping.

For employing `costmap_2d` object, there are a few ways. One of them is providing built map via `map_server` and after localization with *amcl* (Adaptive Monte Carlo Localization) and seen obstacles in `costmap_2d`. For being able to employ the built map as static layer, `map_server` tool is needed. `Map_server` is a ROS node that reads a map from disk and offers it via a ROS service. The current implementation of the `map_server` converts color values in the map image data into ternary occupancy values: free (0), occupied (100), and unknown (-1). Future versions of this tool may use the values between 0 and 100 to communicate finer gradations of occupancy.

The second one is building map of the environment via `hector_slam`, `gmapping` or etc. and detection obstacles and representing them in `costmap` simultaneously.

For our case, the implementation in which map building process and the obstacles detection are proceeding at the same time method is chosen. After managing to successfully building maps with `hector_slam` and `gmapping` previous sections, one of the `costmap` requirements is supplied. The other requirement is sensor data that we already have and be able use in mapping process.

A new launch file is written. This launch file includes `gmapping` launch file that used in the previous section and the `costmap` node. After running the launch file map building initialized. While robot is exploring the environment, map building and object detection is proceeding and `costmap` is being created.

```
<launch>
<!-- Run the implementation of gmapping launch -->
<include file="$(find bsk)/launch/gmapping_lms.launch"/>

<!-- Run the costmap node -->
<node name="costmap_node" pkg="costmap_2d" type="costmap_2d_node"
>
<rosparam file="$(find bsk)/launch/params.yaml" command="load"
ns="costmap" />
</node>
</launch>
```

The params.yaml file includes the parameters that we arranged. One of the important points is that static map parameter should be set true for our case. This parameter indicates external map which is supplied other than navigation stack in which costmap exists. In our case gmapping is supplying the external map. Footprint is a projection of the mobile robot which is used.

```
global_frame: /map
robot_base_frame: base_link
update_frequency: 5.0
publish_frequency: 1.0

static_map: true
rolling_window: false
width: 10.0
height: 10.0
resolution: 0.02

transform_tolerance: 0.3
obstacle_range: 2.5
max_obstacle_height: 2.0
raytrace_range: 3
footprint: [[-0.20, -0.19], [-0.20, 0.19], [0.25, 0.19], [0.4,
0.0], [0.25, -0.19]]
footprint_padding: 0.01
inflation_radius: 0.2
cost_scaling_factor: 10.0
lethal_cost_threshold: 100
observation_sources: base_scan
base_scan: {data_type: LaserScan, sensor_frame: /laser, topic:
/scan, expected_update_rate: 0.4,
observation_persistence: 0.0, marking: true, clearing: true,
max_obstacle_height: 0.5, min_obstacle_height: 0}
```

After running the launch file, while process is proceeding, the system is checked with `rqt_graph`. Graph of our costmap implementation is shown in the Figure 4.31 below.

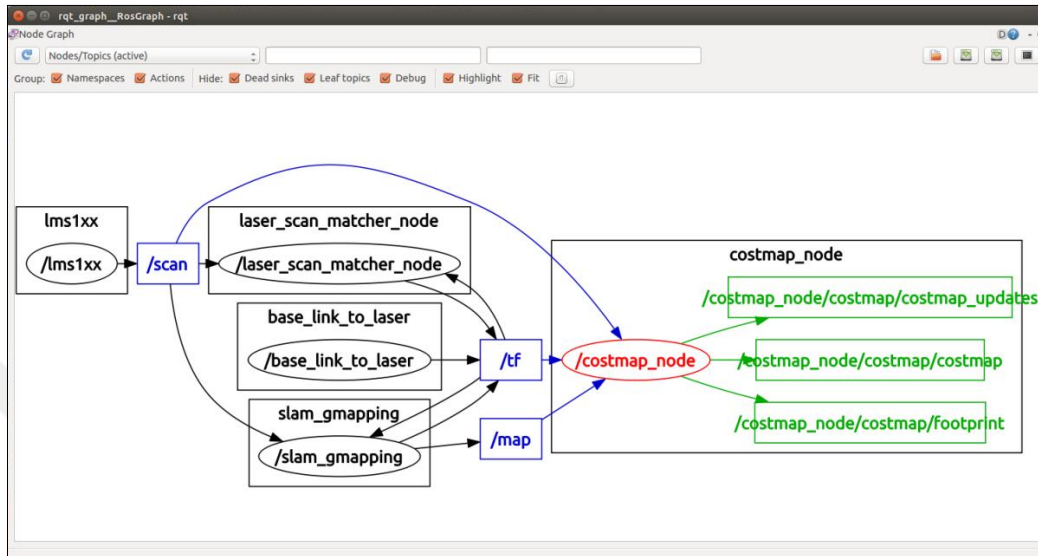


Figure 4.31 Graph of our costmap implementation

Transformation tree (tf tree) is shown in Figure 4.32 below.

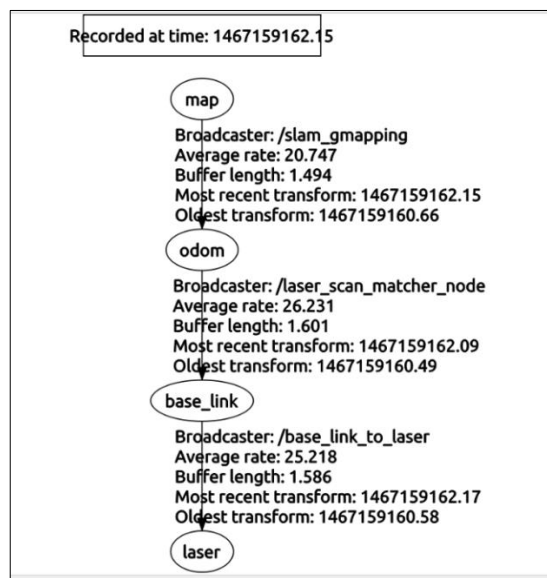


Figure 4.32 tf tree of our implementation

Static map layer represents a largely unchanging portion of the costmap, like those generated by SLAM. Figure 4.33 shows the map built with `hector_slam`.

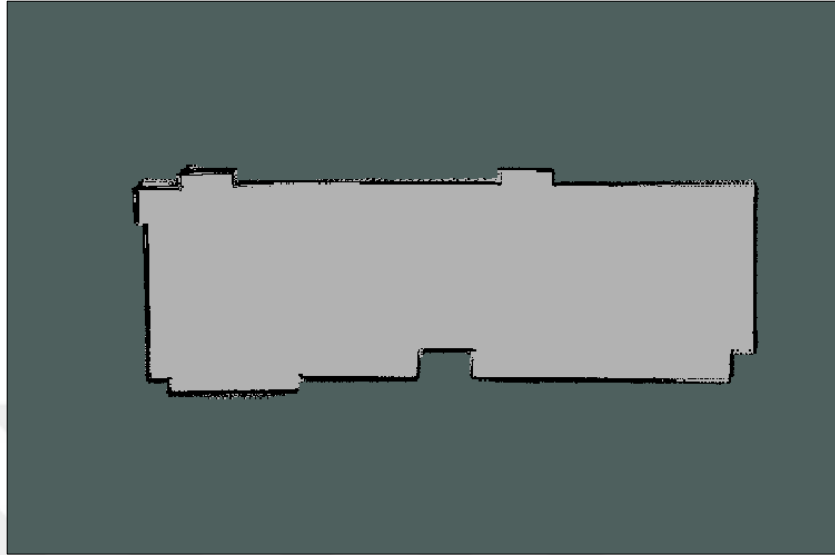


Figure 4.33 Static layer

Obstacle layer tracks the obstacles as read by the sensor data. The `ObstacleCostmapPlugin` marks and raytraces obstacles in two dimensions. Sensor data is shown in blue and the footprint of the mobile robot is shown in red in Figure 4.34.



Figure 4.34 Obstacle layer

Inflation layer is an optimization that adds new values around lethal obstacles. It means it inflates the obstacles in order to make the costmap represent the configuration space of the robot. The inflation layer exists for keeping the robot out of range of obstacles. This is shown in Figure 4.35.

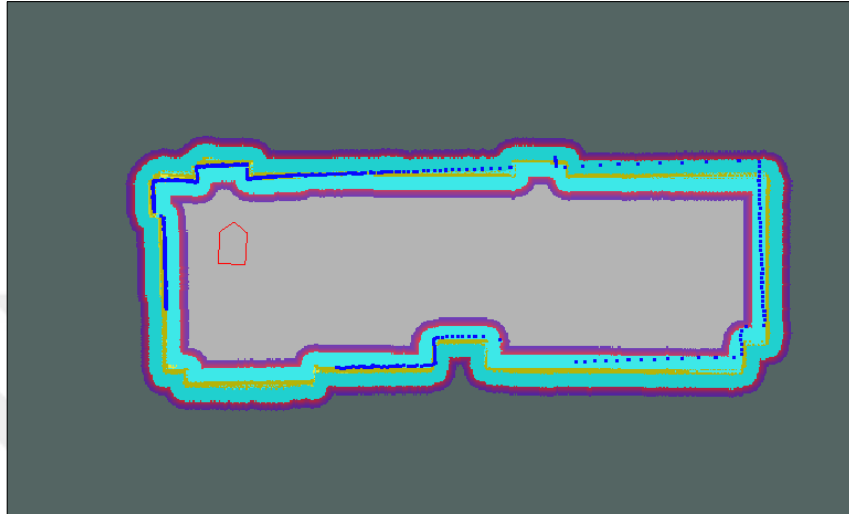


Figure 4.35 Inflation layer also it can be called master layer

While mobil robot is exploring between tables, the object in same the plane with where LIDAR's measurement is done, can be seen. In the experiment below, mobile robot cannot see tables but can see the table legs and chair legs. These objects are represented in the costmap. This can be seen in Figure 4.36.



Figure 4.36 Costmap while robot is exploring between tables in the laboratory

When dynamic objects or a person is coming into mobile robot's sight, object detection is done and represented in the costmap. Whenever the object or the person is disappeared from the sight of mobile robot, costmap clearing function does its job and update the costmap its current stage. This experiment can be seen in Figure 4.37.

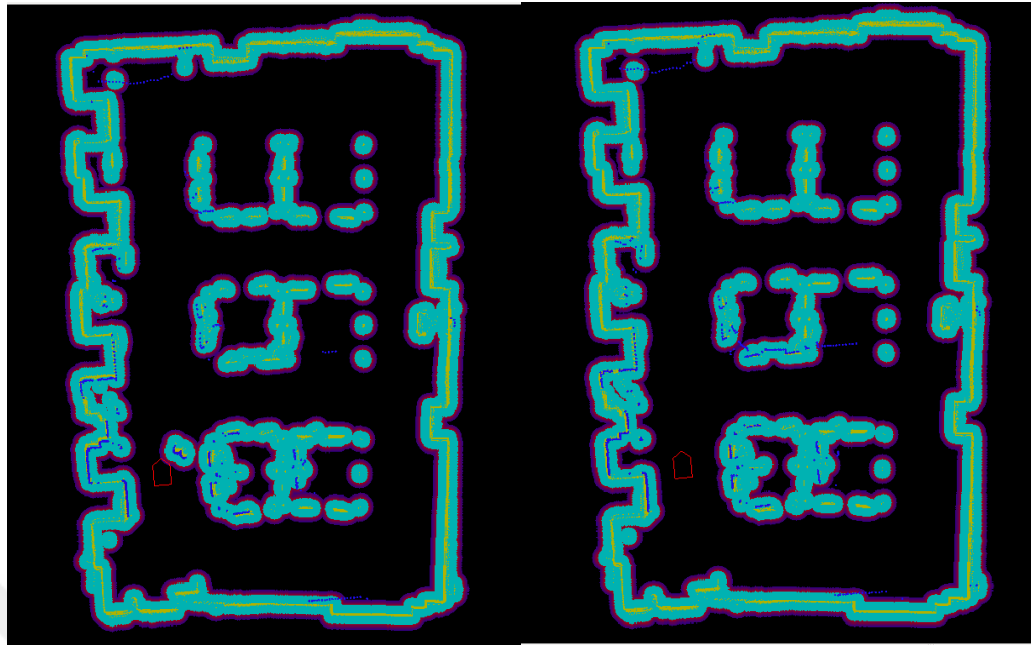


Figure 4.37 Dynamic objects and clearing function of costmap

Costmap of Automatic Control Laboratory is shown in Figure 4.38. LIDAR data can be seen as blue.



Figure 4.38 Costmap of Automatic Control Laboratory

For mobile robots, one of the important things to watch out while exploring an indoor environment is a doorway. The inflation function of costmaps shows its necessity clearly in this experiment. Inflation prevents mobile robot from possible collisions and interactions with objects. This experiment is shown in Figure 4.39 and Figure 4.40.



Figure 4.39 Mobile robot while passing the doorway

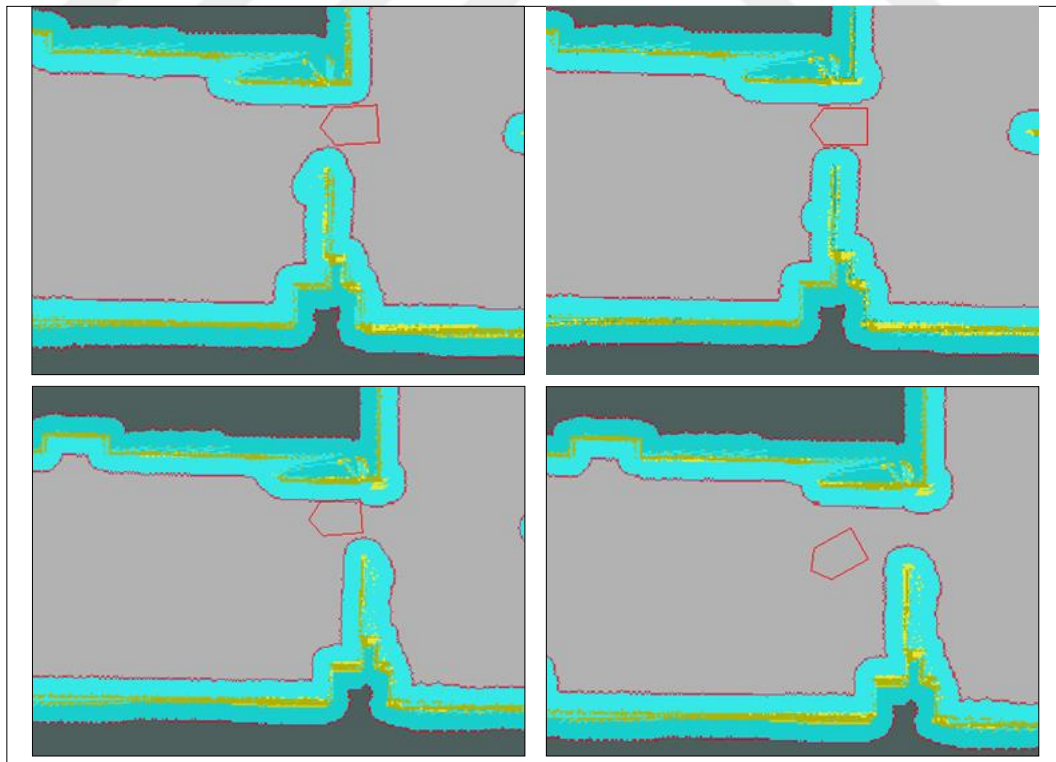


Figure 4.40 Costmap while mobile robot is passing the doorway

For detecting an object in the environment, it is not enough to have knowledge only the properties of the LIDAR, but also knowledge of the system based on the distance and orientation of the object relative to the mobile robot. Object detection with LIDAR is shown in Figure 4.41.

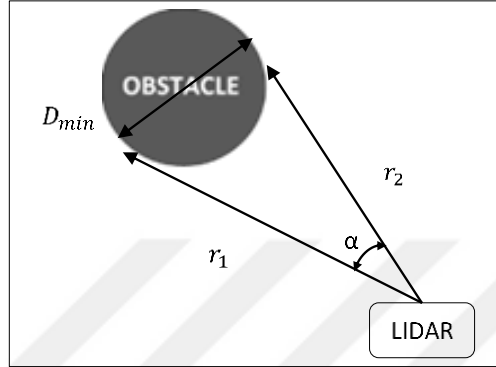


Figure 4.41 Minimum detectable object

D_{min} is the detectable minimum diameter of the obstacle (mm), r_1 and r_2 are range values (mm) of the two consecutive LIDAR beams, α is the angle between two consecutive LIDAR beams. With the law of cosines, D_{min} can be calculated as following equation,

$$D_{min} = \sqrt{r_1^2 + r_2^2 - 2r_1r_2 \cos \alpha} \quad (4.38)$$

As indicated in Table 3.2 in previous chapter, LIDAR with 50 Hz, has an angular resolution 0.5° . So that, for a known two consecutive range data, detectable minimum object can be calculated. For instance, in a scenario where r_1 is 2003 [mm] and r_2 is 1997 [mm], diameter of the minimum detectable object can be estimated as 18.45 [mm].

In the Figure 4.42, the object detection experiment is shown. The target object is the leg of the table which has a 40 [mm] diameter. Distance between the LIDAR mounted on the mobile robot and the target object is R' , 4300 [mm], by using the Eq.

(4.38) diameter of the minimum detectable object is calculated as 37.8 [mm]. As seen in Figure 4.42, even the sensor is stationary an object which has a diameter greater than the minimum value mentioned above can be detected definitely. However, since the robot, so the LIDAR is mobile, objects which have diameters smaller than the calculated minimum diameter can be also detected.

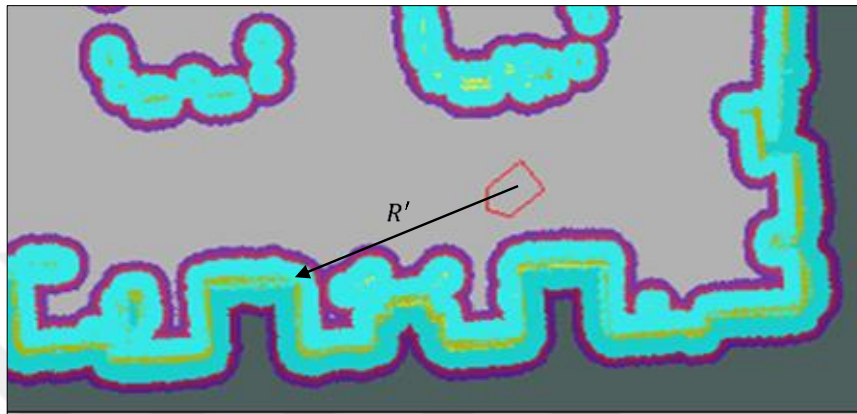


Figure 4.42 Minimum detectable object in costmap

After obstacle detection is achieved and represented in costmaps, precise obstacle avoidance, path planning steps of autonomous navigation can be provided.

CHAPTER FIVE

CONCLUSION

In this research, one of the most fundamental problems for mobile robots, achieving reliable sensor data in operation of mobile robots in unknown environments is discussed. The characteristics of a LIDAR sensor are examined.

Distance traveled by the mobile robot can be sensed by encoders, however this information is not reliable for positioning the robot since wheels may slip even on indoor floor surfaces so that mobile robot may travel less than the distance encoders or other internal sensors indicate (e.g. internal odometry error). For eliminating cumulative errors like those, different types of environmental reference sensors might be used depending on the working environment. Even this method reduces the errors, for improving the data reliability, filters, mean value calculations and combining different sensor data of the same moment might be implemented to the feedback value of the control. Errors occurred because of resolution of sensors (encoders) are analyzed. Robustness of the control depends on the reliability of the sensor measurement. Measurement quality is examined with raw LIDAR data and with filters.

2D map building of the unknown indoor environment via LIDAR data is discussed and map building is achieved using simultaneous localization and mapping (SLAM) algorithms such as Gmapping and Hector SLAM. Built maps are represented in 2D occupancy grid maps. Object detection is studied in order to perform tasks adequately in an environment with obstacles. It is provided with LIDAR-based detection and represented in costmaps.

As a future work, sensor fusion of various sensors such as Internal Measurement Unit (IMU), Kinect and camera, may be applied for more robust automation systems. For the control of the mobile robot and further automation process may be studied and accomplished. Also with hardware modifications of our system, 3D maps of the unknown environments might be built.

REFERENCES

- Alexander, D. (2015). *How to minimize the systematic and the statistical error of SICK LMS sensors*. Retrieved May 11, 2016, from <http://sickusablog.com/how-to-minimize-the-systematic-and-statistical-error-of-sick-lms-devices>.
- Baltsavias, E.P. (1999). Airborne laser scanning: Basic relations and formulas. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54, 199-214.
- Causo, A., Takemura, K., Takamatsu, J., Ogasawara, T., Ueda, E., & Matsumoto, Y. (2010). Predictive tracking in vision-based hand pose estimation using Unscented Kalman Filter and multi-viewpoint cameras. D. Chugo, (Ed.), *Human-Robot Interaction*. (155-170). Croatia: Intech.
- Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo Localization for mobile robots. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 2, 1322-1328). IEEE.
- Doucet, A., De Freitas, N., Murphy, K., & Russell, S. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* (pp. 176-183). Morgan Kaufmann Publishers Inc.
- Doucet, A. (1998). *On sequential simulation-based methods for Bayesian filtering*. Technical report, Signal Processing Group, Department of Engineering, University of Cambridge.
- Doucet, A., De Freitas, N., & Gordon, N. (2001). *Sequential Monte Carlo methods in practice*. Springer Verlag.

- Filliat, D., & Meyer, J.A. (2003). Map-based navigation in mobile robots – I. A review of localization strategies. *Journal of Cognitive System Research*, 4 (4), 243-282.
- Foote, T. (2013). tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on* (pp. 1-6). IEEE.
- Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1), 34-46.
- Grisetti, G., Stachniss, C., & Burgard, W. (2005). Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (pp. 2432-2437). IEEE.
- Habib, M.K. (2007). Robot mapping and navigation by fusing sensory information. In S. Kolski, (Ed.), *Mobile robots: Perception & navigation* (571-594). Germany: Plv/ARS.
- Hopkinson, C., Chasmer, L., Young-Pow, C., & Treitz, P. (2004). Assessing forest metrics with a ground-based scanning lidar. *Canadian Journal of Forest Research*, 34 (3), 573-583.
- Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman Filter to nonlinear systems. In *AeroSense'97* (pp. 182-193). International Society for Optics and Photonics.
- Kandrot, S.M. (2013). *Coastal monitoring: A new approach*. Retrieved March 12, 2016, from <http://research.ucc.ie/journals/chimera/2013/00/kandrot/09/en>.

- Kao, D., Kramer, M., Luo, A., Dungan, J., & Pang, A. (2005). Visualizing distributions from multi-return LIDAR data to understand forest structure. *The Cartographic Journal, Special Issue on GeoVisualization*, 42 (1), 35-47.
- Kim, J., & Lee, J. (2013). Real-time estimation of maximum friction and optimal slip ratio based on material identification for a mobile robot on rough terrain. In *Control, Automation and Systems (ICCAS), 13th International Conference on* (pp.1708-1713). IEEE.
- Kohlbrecher, S., Oskar von, S., Meyer, J., & Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics* (pp. 155-160). IEEE.
- Lu, D. V., Hershberger, D., & Smart, W. D. (2014). Layered costmaps for context-sensitive navigation. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 709-715). IEEE.
- Lu, D. V. (2014). *Contextualized robot navigation*. Ph.D Thesis, Washington University in Saint Louis, Missouri.
- Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision (darpa), in *DARPA Image Understanding Workshop*, (121–130).
- Marcoe, K. (2007). *Lidar ~ An introduction and overview*. Retrieved April 30, 2016, from http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Marcoe_LiDAR.pdf.
- Maybeck, P. S. (1982). *Stochastic models, estimation, and control* (Vol. 3). Academic Press.

- McGee, L. A., & Schmidt, S. F. (1985). *Discovery of the Kalman filter as a practical tool for aerospace and industry*. NASA Technical Memo 86847.
- Meeussen, W. (2010). *REP 105: Coordinate frames for mobile platforms*. Retrieved March 12, 2016, from <http://www.ros.org/reps/rep-0105.html>.
- Nayegandhi, A. (2007). *Lidar technology overview*. Retrieved April 30, 2016, from http://lidar.cr.usgs.gov/downloadfile2.php?file=Nayegandhi_Lidar_Technology_Overview.pdf.
- Raibert, M., Blankespoor, K., Nelson, G., Playter, R., & theBigDog Team (2008). Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress, 17(1)*, 10822-10825. Proceedings Seoul, Korea.
- Riley, P., & Crowe, P. (2006). Airborne and terrestrial laser scanning – Applications for Illwarra Coal. *Coal Operators' Conference*, 266-275.
- ROS concepts*. (n.d). Retrieved April 12, 2016, from <http://wiki.ros.org/ROS/Concepts>.
- Rudan, J., Tuza, Z., & Szederkenyi, G. (2010). Using LMS-100 laser rangefinder for indoor metric map building. In 2010 *IEEE International Symposium on Industrial Electronics* (pp. 525-530). IEEE.
- Salmond, D., & Gordon, N. (2005). *An introduction to particle filters*. Retrieved September 14, 2015, from <http://dip.sun.ac.za/~herbst/MachineLearning/ExtraNotes/ParticleFilters.pdf>.
- Schmidt, S. F. (1981). The Kalman filter - Its recognition and development for aerospace applications. *Journal of Guidance, Control, and Dynamics*, 4(1), 4-7.

- SICK AG (2010). *Laser measurement systems of the LMS1xx product family*. Retrieved August 15, 2015, from <http://www.sick.com>.
- Siegwart, R., & Nourbakhsh, I. R. (2004). *Introduction to autonomous mobile robots*. Cambridge, MA: The MIT press.
- Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.
- Thrun, S. (2002). *Robotic mapping: A survey*. Retrieved May 1, 2016, from <http://reports-archive.adm.cs.cmu.edu/anon/2002/abstracts/02-111.html>.
- Thrun, S. (2002). Particle filter in robotics. In *Proceedings of Uncertainty in AI*.
- Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Dolan, J., et al. (2007). *Tartan racing: A multi-modal approach to the DARPA urban challenge*. Retrieved March 19, 2016, from http://archive.darpa.mil/grandchallenge/TechPapers/Tartan_Racing.pdf.
- Welch, G., & Bishop, G. (1995). *An introduction to the Kalman filter*. Technical Report. University of North Carolina at Chapel Hill Chapel Hill, NC, USA.
- Yıldız, B., & Gören, A. (2012). Engebeli arazide ilerleyebilen gezgin robot tasarımı ve imalatı. *Endüstri Otomasyon Dergisi*, (189), 22-27.

APPENDICES

APPENDIX-1: Technical specifications of LMS100 LIDAR

9 Technical specifications

9.1 Data sheet LMS laser measurement sensor

	Minimum	Typical	Maximum
Functional data			
Scan angle			270°
Scanning frequency	25 Hz		50 Hz
Remission	10%		Several 1,000% ¹⁾ (reflectors)
Angular resolution			
With 25 Hz	0.25°		0.5°
With 50 Hz		0.5°	
Measuring error 1. Reflected pulse ²⁾			
Systematic error		±30 mm (1.18 in)	±50 mm (1.97 in)
Temperature drift	0 mm/°C (0 in/°F)		0.32 mm/°C (0.01 in/°F)
Statistical error (1 σ)		12 mm (0.47 in)	20 mm (0.79 in)
Immunity to external light		40 klx	
Evenness of the scan field (25 Hz)			
Cone		±0.5°	±1°
Inclination		±1°	±2°
Distance from mirror axis of rotation (zero point on the X and Y axis) to the rear of the device		55 mm (2.17 in)	
Distance between centre of the scan plane and the bottom edge of the housing		116 mm (4.57 in)	
Distance measuring range			
LMS100/LMS111/LMS12x/LMS13x/LMS173/LMS182/LMC12x/LMC13x	0.5 m (1.64 ft)		20 m (65.62 ft)
LMS151	0.5 m (1.64 ft)		50 m (164.04 ft)
Hardware blanking window	0 m (0 ft)		15 m (49.21 ft)
Step width		1 m (3.28 ft)	
Power-up delay			60 s
Of a configured device		15 s	
Configurable restart after	2 s		60 s
General data			
Laser protection class	Laser class 1 according IEC 60825-1 (2007-3)		
Enclosure rating	As per EN 60529 (1991-10); A1 (2002-02)		
LMS100/LMS12x/LMS173/LMC12x	IP 65		
LMS111/LMS13x/LMS151/LMS182/LMC13x	IP 67		
Protection class	III as per EN 50178 (1997-10)		
EMC test	As per EN 61000-6-2 (2005-08), EN 61000-6-3 (2007-03)		
Electrical safety	As per EN 50178 (1997-10)		

Tab. 25: Data sheet LMS

	Minimum	Typical	Maximum
Operating temperature range			
LMS100/LMS12x/LMS173/LMC12x	0 °C (32 °F)		+50 °C (+122 °F)
LMS111/LMS13x/LMS151/LMS182/LMC13x	-30 °C (-22 °F)		+50 °C (+122 °F)
LMC12x with mounting kit 1	-30 °C (-22 °F)		+45 °C (+113 °F)
LMC12x with mounting kit 2	-30 °C (-22 °F)		+50 °C (+122 °F)
Storage temperature range	-30 °C (-22 °F)		+70 °C (+158 °F) max. 24 h
Humidity (taking into account the operating temperature range)	DIN EN 60068-2-61, Procedure 1		
Vibration resistance	As per EN 60068-2-6 (1995-04)		
Frequency range	10 Hz		150 Hz
Amplitude	5 g RMS		
Shock resistance	As per EN 60068-2-27 (1993-03), EN 60068-2-29 (1993-04)		
Single shock	15 g, 11 ms		
Continuous shock	10 g, 16 ms		
Sender	Pulsed laser diode		
Wave length	895 nm	905 nm	915 nm
Divergence of the collimated beam (solid angle)		15 mrad	
Light spot size at the optics cover		8 mm (0.31 in)	
Light spot size at 18 m (59.06 ft) scanning range		300 mm (11.81 in)	
Housing			
Material	GD-ALSi12 3.2582.05		
Color LMS100	RAL 5012 (blue)		
Color LMS122/LMS132/LMS182	RAL 9005 (black)		
Color LMS123/LMS133/LMS173	RAL 9003 (white)		
Color LMS111/LMS121/LMS131/LMS151	RAL 7032 (gray)		
Alloy	Excellent weather resistance as per DIN EN 106:1988, plate 3		
Optics cover			
Material	Polycarbonate		
Surface finish	Outside with scratch-resistant coating		
System plug (LMS100/LMS12x/LMS173)			
Material	GD-ALSi12 3.2582.05		
Color	RAL 9005 (black)		

Tab. 25: Data sheet LMS

	Minimum	Typical	Maximum
Cable entries (LMS111/LMS151)			
Material	Stainless steel/plastic		
Dimensions ³⁾			
Height LMS100/LMS122/LMS123/LMS173/ LMC12x			152 mm (5.98 in)
Height LMS111/LMS13x/LMS151/LMS182/LMC13x			162 mm (6.38 in)
Width			102 mm (4.02 in)
Depth			106 mm (4.17 in)
Total weight (without connection cables)		1.1 kg (2.43 lb)	
Electrical data			
Supply voltage LMS100/LMS111/LMS151 SELV or PELV as per IEC 60364-4-41 (2005-12)	10.8 V	24 V	30 V
Supply voltage LMS12x/LMC12x/LMS13x/LMC13x/ LMS173/LMS182 SELV or PELV as per IEC 60364-4-41 (2005-12)	9 V	24 V	30 V
Permissible residual ripple			±5%
Supply voltage for the LMS111/LMS151/LMS182/ LMS13x/LMC13x heating	19.2 V	24 V	28.8 V
Switch on current			2 A
Operating current at 24 V without output load		0.35 A	0.5 A
Operating current with max. output load		0.65 A	0.8 A
Operating current with max. heating performance		2.3 A	2.5 A
Power consumption without output load		8.4 W	12 W
Power consumption with max. output load		16 W	20 W
Power consumption with max. heating performance		55 W	60 W
Electrical connection LMS111/LMS13x/LMC13x/LMS151	M12 round plug connector		
Electrical connection LMS100/LMS12x/LMC12x	Removable system plug with screw terminal block		
Technical specifications, screw terminals			
Cross-section of rigid cores (American Wire Gauge – AWG)	0.14 mm ² (approx. 26 AWG)		1.5 mm ² (approx. 16 AWG)
Cross-section of flexible cores (American Wire Gauge – AWG)	0.14 mm ² (approx. 26 AWG)		1.0 mm ² (approx. 18 AWG)
Insulation stripping length for the cores		5 mm (0.20 in)	
Screw tightening torque	0.22 Nm		0.3 Nm

Tab. 25: Data sheet LMS

	Minimum	Typical	Maximum
Cable length for device power supply at 24 V			
With 1 mm ² wire cross-section (approx. 18 AWG)			220 m (721.78 ft)
With 0.5 mm ² wire cross-section (approx. 22 AWG)			110 m (360.89 ft)
With 0.25 mm ² wire cross-section (approx. 24 AWG)			50 m (164.04 ft)
Cable length for heating at 24 V			
With 1 mm ² wire cross-section (approx. 18 AWG)			45 m (147.64 ft)
With 0.5 mm ² wire cross-section (approx. 22 AWG)			20 m (65.62 ft)
With 0.25 mm ² wire cross-section (approx. 24 AWG)			10 m (32.80 ft)
Cable length for device power supply at 12 V			
With 1 mm ² wire cross-section (approx. 18 AWG)			20 m (65.62 ft)
With 0.5 mm ² wire cross-section (approx. 22 AWG)			10 m (32.80 ft)
With 0.25 mm ² wire cross-section (approx. 24 AWG)			5 m (16.40 ft)
Switching inputs			
Number	2		
Input resistance on HIGH		2 kΩ	
Voltage for HIGH	11 V	24 V	30 V
Voltage for LOW		0 V	5 V
Input capacity		15 nF	
Static input current	6 mA		15 mA
Dynamic control inputs			
Number	2		
Input resistance on HIGH		2 kΩ	
Voltage for HIGH	11 V	24 V	30 V
Voltage for LOW	-3 V	0 V	5 V
Input capacity		1 nF	
Static input current	6 mA		15 mA
Duty cycle (Ti/T)		0.5	
Input frequency			100 kHz
Current load per incremental encoder		50 mA	100 mA
Velocity range that can be sampled			
Forward	From +100 mm/s to +20,000 mm/s (+3.94 in/s to +788 in/s)		
Backward	From -10 mm/s to -20,000 mm/s (-0.39 in/s to -788 in/s)		
Incremental encoders that can be evaluated			
Type	Two-channel rotary encoder with 90° phase offset		
Enclosure rating	IP 54		
Supply voltage	V _S - 3 V		V _S

Tab. 25: Data sheet LMS

	Minimum	Typical	Maximum
Outputs required on the incremental encoders	Push/pull		
Pulse frequency			100 kHz
Number of pulses per cm	50		
Cable length (screened)			10 m (32.80 ft)
Digital switching outputs LMS100/LMS111/LMS151			
Number	3		
Voltage drop load		2 V	
Maximum switching current			140 mA
Current limiting (after 5 ms at 25 °C (77 °F))	100 mA		200 mA
Power-up delay	Negligible		
Switch off time		0.8 ms	2 ms
Relay switching outputs (semiconductor relay/ MOSFE/) LMS12x/LMC12x/LMS13x/LMC13x/ LMS173/LMS182			
Number	2		
Switching voltage			DC/AC 40 V
Switching current			0.5 A
Contact resistance		0.34 Ω	0.7 V
Output capacitance			220 pF
Dielectric strength of input/output			1500 VAC
Power-up delay		1.3 ms	0.1 ms
Switch off time		0.1 ms	0.5 ms
Serial auxiliary interface			
Communication protocol	RS-232 (proprietary)		
Transmission speed	9,600 Baud	57.6 kBd	115.2 kBd
Serial host interface			
Communication protocol	RS-232 (proprietary)		
Transmission speed (selectable)	9,600 Baud	57.6 kBd	115.2 kBd
Cable length at 38,400 kBd and wire cross- section 0.25 mm ² (approx. 24 AWG)			15 m (49.21 ft)
Galvanic de-coupling	Yes		
Wire cross-section of the connecting cable			0.25 mm ² (approx. 24 AWG)
Ethernet	10/100 MBit/s		
CAN	20 kBit/s, 500 kBit/s, 1 MBit/s		

Tab. 25: Data sheet LMS

- 1) Corresponds to Diamond Grade 3000X™ (approx. 1,250 cd/lx × m²).
- 2) The time after the first reflected pulse from which measurement can be performed with full accuracy is dependent on the target that reflected the first reflected pulse.
- 3) Without fixing screws and projection of cable glands with system plug mounted.

APPENDIX-2: Hector_mapping_default.launch File

```
<launch>

<arg name="tf_map_scanmatch_transform_frame_name"
default="scanmatcher_frame"/>
<arg name="base_frame" default="base_footprint"/>
<arg name="odom_frame" default="nav"/>
<arg name="pub_map_odom_transform" default="true"/>
<arg name="scan_subscriber_queue_size" default="5"/>
<arg name="scan_topic" default="scan"/>
<arg name="map_size" default="2048"/>

<node pkg="hector_mapping" type="hector_mapping"
name="hector_mapping" output="screen">

<!-- Frame names -->
<param name="map_frame" value="map" />
<param name="base_frame" value="base_link" />
<param name="odom_frame" value="base_link" />

<!-- Tf use -->
<param name="use_tf_scan_transformation" value="true"/>
<param name="use_tf_pose_start_estimate" value="false"/>
<param name="pub_map_odom_transform" value="$(arg
pub_map_odom_transform)"/>

<!-- Map size / start point -->
<param name="map_resolution" value="0.0250"/>
<param name="map_size" value="$(arg map_size)"/>
<param name="map_start_x" value="0.5"/>
<param name="map_start_y" value="0.5" />
<param name="map_multi_res_levels" value="2" />

<!-- Map update parameters -->
<param name="update_factor_free" value="0.4"/>
<param name="update_factor_occupied" value="0.9" />
```

```

<param name="map_update_distance_thresh" value="0.4"/>
<param name="map_update_angle_thresh" value="0.06" />
<param name="laser_z_min_value" value = "-1.0" />
<param name="laser_z_max_value" value = "1.0" />

<!-- Advertising config -->
<param name="advertise_map_service" value="true"/>

<param name="scan_subscriber_queue_size" value="$(arg
scan_subscriber_queue_size)"/>
<param name="scan_topic" value="$(arg scan_topic)"/>

<param name="tf_map_scanmatch_transform_frame_name" value="$(arg
tf_map_scanmatch_transform_frame_name)" />
</node>

<!--<node pkg="tf" type="static_transform_publisher"
name="map_nav_broadcaster" args="0 0 0 0 0 0 map nav 100"/>-->

</launch>

```

APPENDIX-3: gmapping_params.yaml File

```
map_udpate_interval: 1.0
maxUrange: 5.0
sigma: 0.1
kernelSize: 1
lstep: 0.15
astep: 0.15
iterations: 1
lsigma: 0.1
ogain: 3.0
lskip: 1
srr: 0.1
srt: 0.2
str: 0.1
stt: 0.2
linearUpdate: 1.0
angularUpdate: 0.5
temporalUpdate: 0.4
resampleThreshold: 0.5
particles: 10
xmin:-5.0
ymin: -5.0
xmax: 5.0
ymax: 5.0
delta: 0.02
llsamplerange: 0.01
llsamplestep: 0.05
lasamplerange: 0.05
lasamplestep: 0.05
```