POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

OBJECT DETECTION FOR UNMANNED GROUND VEHICLE BY USING LiDAR

Candidate
Nazanin Fasihihour

Supervisor
Prof. Marco Piras

Co-Supervisor
Dr. Di Pietra

October 2021

ACKNOWLEDGMENTS

This dissertation concludes from a difficult route, involving tears, sleepless nights, expenses, but also many gratification and happy moments. All of these moments together create an experience that has forged me as a person and as a professional. I would like to sincerely thank all the people who have supported and accompanied me on this path.

My first thoughts go to my parents who have supports me to follow my studies disdain the difficulties faced due to the distance that separates us and that has supported me in the darkest moments of this journey.

A special thanks go to Prof. Marco Piras who astutely guided and patiently helped me during this research and thanks to whom it was possible to complete even the last phases of the experimentation despite the difficulties imposed by the sanitary circumstances of this year.

Last but not least, I would like to thank all the friends who have been by my side and shared both the most difficult and the happiest moments of this path with me. Lastly, I dedicate this thesis work to my father who would surely be proud of the achieved goal if he were alive.

Nazanin Fasihihour

Torino 2021

TABLE OF CONTENTS

LIST OF FIGURES

ACRONYMS

ANNs	Artificial neural networks
ADAS	Advanced driving assistance system
ADS	Automated driving system
b	Bias
c	Classes number
C_{ki}	Measured value
$\tilde{C}_{ki}$	Estimated value
CW	Continuous-wave
CACC	Cooperative adaptive cruise control
DNN	Deep neural network
EKF	Extended kalman filter
F	Filter size
FMCW	Frequency modulation
GNSS	Global navigation satellite system
IoU	Intersection over union
L_{new}	Total loss
L_{prev}	Arbitrary loss function
LADAR	LAser detection and ranging
LiDAR	Light detection and ranging
MaaAS	Mobility-as-a-service
MAPE	Mean absolute percentage error
MAE	Mean absolute error
MSE	Mean squared error
NMSE	Normalized mean squared error
NMAE	Normalized mean absolute error
N	Number of input values
O	Output of the network
O_i	Model output for the i-th sample
P	Padding row/column size

R	Range of medium point and θits angle
RBPF	Rao-blackwellized particle filter
ReLU	Rectified linear unit
RoI	Regions of interest
RMSE	Root mean squared error
RADAR	RAdio detection and ranging
RPN	Region proposal network
RANSAC	Random sampling consensus
RBPF	Rao-Blackwellized particle filter
ROS	Robotic operating system
SSD	Single-shot multibox detector
SAW	Surface acoustic wave
S	Stride
SAE	Society of automotive engineers
TOF	Time of fly
T_i	Target output
TaaS	Transportation as a service
UGV	Unmanned ground vehicle
x	Input
YOLO	You Only Look Once
ω	Learnable weights
$\hat{O}$	Number of desired outputs
$\nabla J(\theta)$	Gradient of the loss function
θ	Parameter to be updated
η	Learning rate
γ	Momentum degree
α	Small value near zero in the function given by the next formula
y_i	Output probability of a particular class i in the output layer
λ	Regularization parameter represents

SUMMARY

Nowadays, increasing peace and improving the quality of life have been becoming important issues worldwide. To increase welfare in cities, the use of automatic cars and self-controllers is very important to reduce the accident rate and its related financial and human losses, which depends on increasing human knowledge and the use of powerful technologies such as LiDAR sensors. This type of sensor has high efficiency in identifying the objects detection and with the help of data that can be provided from this sensor, it is possible for engineers to prepare maps and after preparing various maps over time, it is possible to identify the objects location and their movement around a car. Therefore, in this thesis, the structure, use and performance of the LiDAR sensor have been identified for object detection and mapping. Then, to measure the accuracy of the LiDAR sensor, the sensor is installed on a ligament and then the position of various objects were measured. Then, the accuracy of the LiDAR sensor was evaluated by measuring the exact distance of objects from the sensor location using mean squared error (MSE), normalized mean squared error (NMSE), root mean squared error (RMSE), mean absolute error (MAE), normalized mean absolute error (NMAE) and mean absolute percentage error (MAPE). Thus, by programming at the top, the data collected by the sensor is prepared. Consequently, by collecting all the data, 2-D maps of the identified and measured objects have been prepared by using Gmapping in ROS from different places. Finally, three artificial neural networks (ANNs) were utilized, Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate, for object detection instead of using LiDAR sensor by identifying the angle and distance of objects, relative to the sensor location which helps to reduce costs and time-consuming. Therefore, the accuracy of different networks was assessed and discussed.

Keywords: artificial neural network, automation vehicle, LiDAR sensor, 2-D mapping, object detection.

INTRODUCTION

In the last several decades there has been a tremendous interest in self-driven vehicles in the field of robotics and the industrial sector more in general, and significant growth in the number of applications for which these vehicles are employed. The efforts of numerous cars for research and development in order to produce increasingly competitive, safe and accurate Autopilots have been a striking illustration of this phenomenon. This is also owing to the development of advanced onboard computers and extremely accurate and efficient sensors, as well as the emergence of remote controls which also seek to provide unparalleled connectivity with the car, particularly thanks to a new technology called network slicing. The robotics industry is currently focusing on delivering solutions that may encourage and strengthen human-machine collaboration, not just in militaries and space exploration as it used to be, in agriculture, as well as in others directly connected with the civic field. This year, this process has witnessed an increase in lifestyles, time-demanding and high car usage and traffic which are the main accident consequences like cost and human losses. This circumstance has more than ever underlined the necessity for autonomous instruments such as medical equipment and commodities to be transported, without risk of interaction with others or the requirement for anyone to depart. Today, self-driven vehicles, whether on land, in the air or elsewhere, are one of the most increasing realities. In many applications where environmental awareness is essential, such as driverless cars, autonomous robotics and enhanced/virtual reality, object detection plays a crucial role. Thus, environmental sensing, self-location, path planning and motion control should constitute the framework. The safe interplay of these autonomous systems with the environment and humans depends heavily on the capacity for the environment to detect, perceive and model. In this among, there have been many signs of progress in the field of 2D detection which help self-driving vehicles to detect objects located around, and the two main criteria of such 2D object detectors are their robustness and real-time inference. The 2D objects detection (2DOD) job is specified by a list of specific items to be identified, as a task in which the machine can recognize the objects within the scene and locate them in 2D space relative positions.

In this scenario, the objective of this thesis work is the realization of LiDAR sensor ability to detect objects and mapping in ROS and its adaptation to the complexities of the environment around a car and the position of various objects that may be unpredictably in the position of the car and pose a danger to the driver and other passengers. Learn how to connect to the computer and enable coding to get high accurate performance and object identification. LiDAR's sensor is the usefulness environmental sensor. LiDAR gives 2D and 3D information on the shape of an item and the relative distance from the sensor of the object. 2D object detection has the difficulty to obtain strong performance for real-life applications, as neither sensor alone can give sufficient information. The fusion of 2D data from LiDAR is one of the existing obstacles in multi-modal 2D objects identification.

Conversely, there are a variety of motivations for this phenomenon. One of the many is that the technology sector is experiencing exponential growth, both in terms of investment and resources expended for research in the field and the ability of new instruments on the market to fulfil its aims more efficiently, safely and accurately, in particular in the field of sensors, and of all parts of the world. The emergence of artificial robotic programs, one of the aims of which is to ensure unparalleled vehicle-to-all communication through its new technologies, the so-called "network cutting," contributed to the growth in interest in the field, especially in recent years. One final major contribution must definitely be credited to the increased usage of artificial intelligence, namely autonomous (or autonomous) learning techniques such as neural networks, machine learning and deep learning. In the realm of object recognition, recent deep learning techniques, particularly for two-Data, such as camera pictures, have made important advances. Several new detection models for real-time activities such as self-sufficient driving have therefore been created. Therefore, developing highly accurate artificial neural networks could help to use the camera instead of high-tech LiDAR sensors. But to make sure about the used algorithms connecting the camera to the car, the different artificial neural networks should be adopted and validated with the object detection outcomes achieved with the LiDAR sensor.

According to what has been explained above, the importance of using intelligent control of vehicles in reducing fuel consumption, as well as casualties and damage from traffic accidents was identified. To develop the use of smart devices and equipment in the control of vehicles, various methods and assumptions have been made by researchers according to the level of control. Security has therefore always been a major concern in the transport industry because all parties involved in the supply chain (cost of recipient delays, vehicle repairs and possible medical costs of the driver's for the agency, expenses of damaged goods and other complications if safety requirements are not fulfilled and accidents occur, have been affected. As technology and markets evolve, safety standards for traffic have been strengthened throughout time to suit the market requirements (e.g. shorter lead times, lower excess inventory, better tracking) and trade unions (working conditions, workplace safety, among many more). But now, when cars and road infrastructure are typically dependable and the legislation and rules have come to maturity such that all passengers should be safe by road if they comply with legislation, drivers have become the major cause of road accidents.

Therefore, what is examined in this thesis is to answer the following questions:

- ✓ To what extent can the identification of different objects while driving be analyzed and evaluated?
- ✓ How to work with the sensor and the connection between the computer and the sensor to identify objects?
- ✓ How to use algorithms to identify objects and present a map?
- ✓ How accurate are the sensor and algorithm provided in identifying the position of objects and preparing a map of their position?

- ✓ Are there other solutions to reduce the cost and increase the use of self-drive, such as using an artificial neural network? And to what extent the provision of an artificial neural network can help predict the position of objects?
- ✓ How much trust is there in neural networks to connect a computer to a simple camera to detect the position of objects around a machine? and how accurate the neural networks are compared to the in-real time results of the LiDAR sensor

The current thesis in the field of object detection and tracking is largely based on visualization (images and mapping). Multispectral data were worded by visual sensors bids remarkable and characteristic features, hence enabling easy division of objects in crowded areas. Multi-objective detection and tracking have created these criteria and challenges have the most advanced technology in the field of detection and tracking. Though, it is difficult to obtain accurate 2D position data only with visual sensors. Similarly, optical sensors are exceedingly exposed to lighting conditions. Some attempts have been made to merge two and two-dimensional laser rangers and cameras using data integration techniques. The advantages of a laser scanner are accuracy in location measurement, straight measurement of three-dimensional situations, Infiltration of numerous obstacles and an active sense method that allows it to work daily and nightly. Because of these advantages, investigations in the field of detection and tracking with LiDAR has been carried out in this thesis. Thus, considering the number of objects and the relationship between the objects and continuous and discontinuous frames, mapping was carried out in the ROS to find the location of objects in the map. Finally, based on the obtained results using the LiDAR sensor, an Artificial Neural Network (ANN) algorithm was developed to predict the distance and location of objects based on the location of the sensor. This allows us to find a location of objects around the vehicle with no need for a costly and complex LiDAR sensor. The overall overview of investigations carried out in this thesis is presented in Figure i.

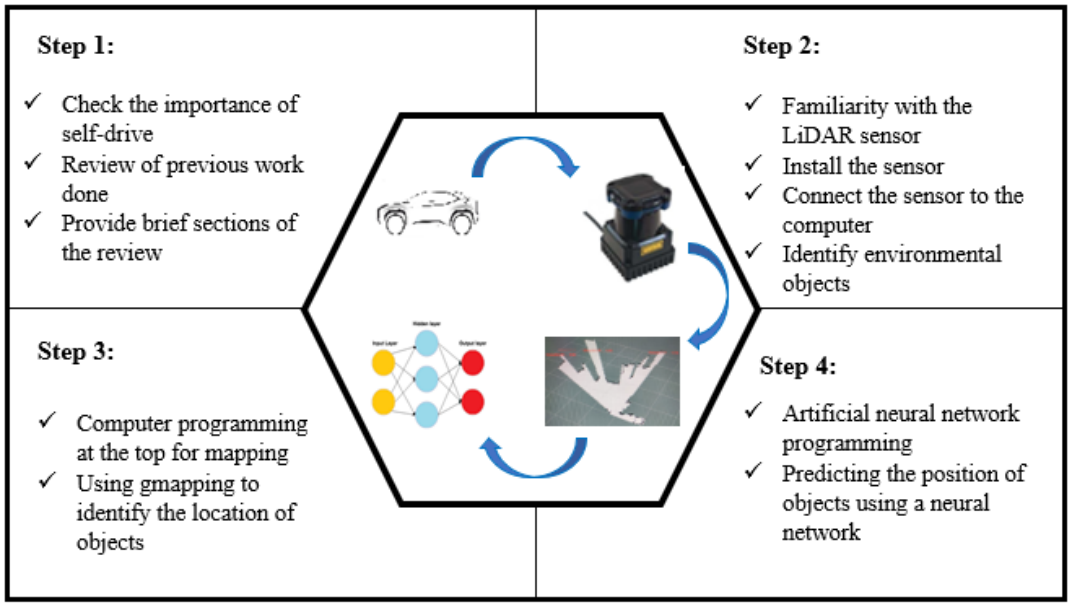


Figure i. General overview of carried out steps in the current thesis

Al in all, the current thesis is provided in the following chapters.

- ✓ *Chapter 1:* A literature review about object detection and using sensors and their algorithms have been carried out to highlight the importance of doing the current investigation;
- ✓ *Chapter 2:* In this section, the methodology and basic concepts are discussed about object detection. For this aim, the was for using sensor, LIDAR, its performance for data collection is discussed;
- ✓ *Chapter 3:* in these sections, the detection procedure and the tracking process for mapping are presented in the proposed framework;
- ✓ *Chapter 4:* in this section, the basic concept about the artificial neural network and the importance of using ANN is discussed using different algorithms;
- ✓ *Chapter 5:* in this section, the performed formulations and used algorithms in the current thesis are presented and discussed in detail;
- ✓ *Chapter 6:* in this section, the use of LiDAR sensors, the accuracy of the used sensor, and the obtained results are presented and discussed;
- ✓ *Chapter 7:* in this section, results were presented and discussed in the shape of a conclusion and presented the different recommendations to follow this study in future investigations by researchers.

CHAPTER 1

A BRIEF INTRODUCTION ABOUT UNMANNED GROUND VEHICLE

1.1. Introduction

Self-employed vehicles were established to reduce problems such as collisions, time to travel, pollution etc. they should be helpful nowadays. Many individuals go every day from home to work, school, fitness areas, parks, etc. Some people travel from one corner of the globe to another, whether it's from house to office or from locations across the world. Today, physically and symbolically, the globe is swiftly ranking. This movement is facilitated by recent inventions from hoverboards to supersonic jets. The increased population need for transport has also increased. Increased revenue per person typically indicates a better way of living that can be linked to better, pleasant and safer travel requirements. New models such as transportation as a service (TaaS) and mobility-as-a-service (MaaAS). The emphasis on individual or corporate car-sharing, have been promoted by the increase in demand for travel. Some instances of this may be seen in ride-sharing programs such as Uber and country-specific applications, such as Chauffeur Priv'e (in France). Low-cost flights, established largely a decade ago, draw most on the aviation market.

1.2. Various levels of autonomy

Object detection recognizes instances of semantic objects from other images or data, although object pursuing is demarcated as how one or more objects are tracked over a series of time phases. Object detection and tracking have been widely investigated for their wide claims in visual observation, traffic controlling, robotics, and self-driving. For instance, to provide safety and acceptable control for an autonomous vehicle, it must sense, track, and forecast the movement of moving objects nearby. The number of objects around a road (such as guardrails, pedestrians, animals) varies considerably which have different behaviour, reaction, velocity and size. A moving sensor can also make a big difference in the appearance of a fixed circumstantial due to changes in perspective. Additionally, obstruction can make objects vanish completely. All of these features make it very stimulating to sense and track moving substances in urban acts. The Society of Automotive Engineers (SAE) defines 6 levels of driving automation from Level 0 (fully manual) to Level 5 (fully autonomous), as described below and shown in Figure 1-1.

- **Level 0 (No Driving Automation):** The driver plays the most important role for all real-time events that are required to drive a car, like manual cars on the roads. At this level, there are certain automated mechanisms in place to assist the driver. Like, standard cruise control and emergency braking system;
- **Level 1 (Driver Assistance):** The driver is in charge of essential functions like steering and braking. But vehicles are equipped with some brand-new automated driving assisted systems, for instance, adaptive cruise control and lane control assist. These features have taken the control of the safety of the driver and car as well;
- **Level 2 (Partial Automation):** Advanced Driving Assistance System (ADAS) are the crucial aspect of level 2. Braking, acceleration, and steering are done automatically. Nevertheless, these vehicles are not fully automated, as the driver should pay full attention to the surroundings and remains in control;

- **Level 3 (Conditional Automation):** A level 3 car can drive individually employing an automated driving system (ADS) under certain conditions. They are capable of long-distance automatic driving; Cooperative Adaptive Cruise Control (CACC) is in the category of level3. CACC can produce dramatic increases in highway capacity, from 103% to 273%;
- **Level 4 (High Automation)**: an automated system can perform the driving task and monitor the driving environment without the need for a human to take control; nevertheless, the automated system can only operate in specific situations and under specific situations;
- **Level 5 (Full Automation):** an autonomous machine can do all driving activities in the same way that a human driver could.

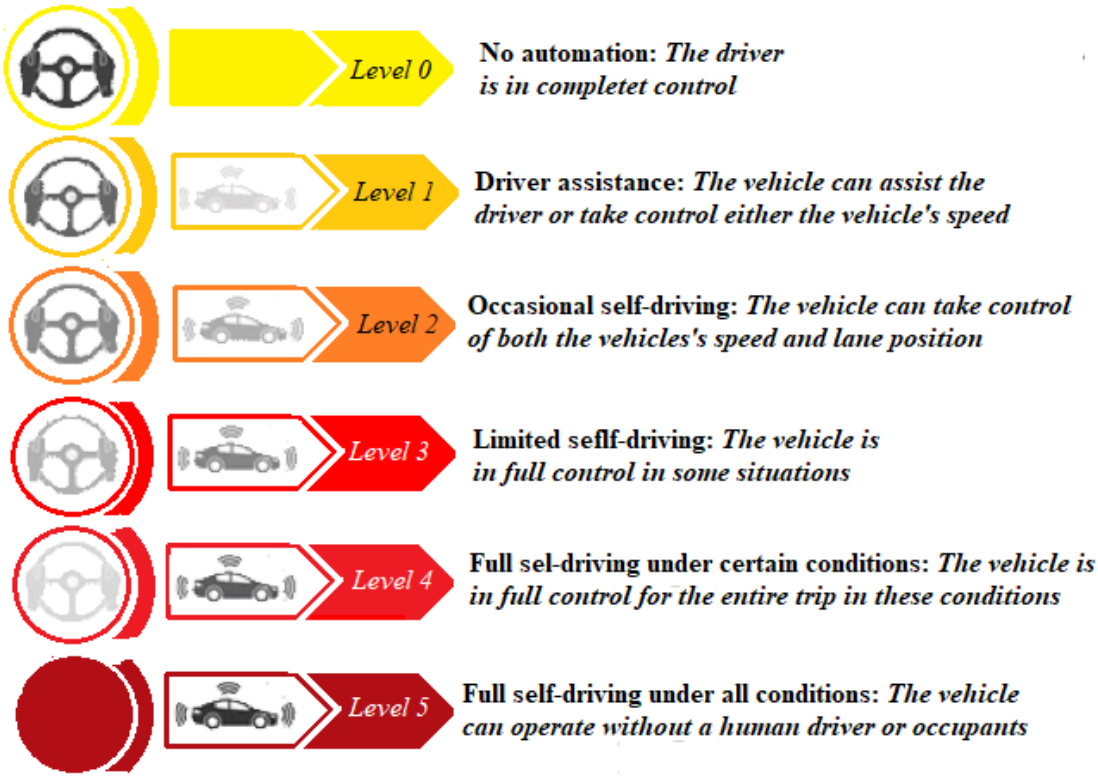


Figure 1-1. Different levels of driving automation (DA)

Sensors are an important element of the procedure in the background of autonomous driving. As already indicated, sensors can collect very required and valuable data in order to construct a model environment. For this aim, several kinds of sensors have been created and assessed and often integrated to provide a meaningful depiction of the surroundings. Two major groups of sensors can be grouped:

1- Active sensors with their tools that depend on their individual sources of contamination to detect the location, shape, distance of substances around the car and other data on the environment, by reflecting the propagating signal returned to the origin;
2- Passive sensors utilise the data already available in the environment to capture radiation, such as light, heat, vibration or other environmental events.

The primary family of sensors comprises radars, LiDAR and ultrasound sensors. These sensors are usually combined with passive, infrared or hyperspecific cameras, RGB cameras, GPS and inertial sensors in self-directed driving submissions to get more comprehensive environmental illustrations. Moreover, it is worth stating all those sensors that require touching the environment they feel. Thermometers, tactile sensors, strain gauges and bumper sensors are these devices. This cannot be complicated when it comes to a thorough description of the environment, as a multifaceted illustration does not lead to a well outcome. The environment in which the car operates relies on this. Of course, as the application is more flexible and broad, the more the image can adjust for new occurrences.

The presentation would, for example, be fairly basic if the intention was to run a combination gatherer on a straw arena. For any position correction only two sensors, a GPS and an IMU may be required. In such a scenario, the sophisticated and specific representation of this field is not essential to produce super accurate and costly sensors. In contrast, it would be necessary to employ different sensors, thereby creating a more complicated representation in a multiplicity of settings, where there are numerous dynamic barriers, such as people or other moving cars. The most frequent forms of illustrations nowadays are:

- Metric maps of topology 1.2 combining a 2D metric and topological frame. In the past objects are recognized by accurate coordinates, but are extremely susceptible to noise, whereas in the last just locations are regarded as graphs and their connections;
- Complete metric maps, creating a fixed coordinate system for the complete operational environment. As highlighted by [11], the tendency in recent years towards this kind of illustration is attributable to many causes, including:
  - ✓ More precise localization methods further enhance GPS integrity, accuracy and accessibility owing to technologies such as the GNSS (Global Satellite Navigation System).
  - ✓ More and more influential computers that can store and procedure more and more data in fewer time. So these computers can interrelate with ever wider and more comprehensive maps;
  - ✓ More precise sensors like stereo and LiDAR vision;
  - ✓ Recent advances in procedure planning using these map categories.

Maps may be produced in real-time either using the car sensors or fully supplied with the algorithm if received in advance. We can keep a map updated, static or dynamic while the robot is moving. Hybrid versions including the examples above can be utilized for best outcomes, as with most engineering solutions. For instance, two illustrations can be utilized simultaneously, one globally and one locally. A local illustration with extremely detailed data shows what the robot is around, while the global illustration comprises all the data most important to the "perceived" robot in the limited surround. The difficulty of each individual depiction leads to this kind of planning. Locally, the robot is in danger of showing myopia while overloading its memory and extending its

calculation algorithm times worldwide, particularly on extremely big maps with a wealth of information.

1.3. Unmanned ground vehicle

The unmanned ground vehicle (UGV) to be employed in both simulations and field testing is an important and required factor to pick from. Firstly, there will be a definition and an overview of the UGV scene. Without the physical presence of humans, the word UGV defends a land vehicle. This sort of vehicle is utilized in various applications and fields as well as in three principal forms of vehicle control as outlined in [13]:

1- manual, which generally sends direct commands through a remote radio control to car actuators;
2- Monitoring control, when a human supervision officer remotely delivers extremely general orders to the vehicle and can understand these directions properly and divide them into simpler, more granular measures. The supervisor may simultaneously verify that the vehicle does the duty properly. This is the intermediary phase between manual and completely automated control;
3- Automatic, wherein the car may interrelate with the surrounding environment and decide based on the allocated job independently. The most conventional parts of use are those in which a UGV has to be employed for risky activities such as disrupting the bomb and treating hazardous material, or when the site is not available to a man or even carry large things better. As a result, the fields of interest might include military, interplanetary or even civilian exploration, such as agriculture, production, mining, and disaster relief.

Different UGV is available to find the route based on the project using for, environment and ground surface conditions. These vehicles move with different velocities and work with a tiny off-road radio-controlled car capable of achieving speed on flat terrain. Figure 1-2 shows various wheelbases and an axle path as examples, even if the vehicle goes up steeply, and also has a wide centre surface, which may be utilized for installing several navigational components, shown in Figure 1-3. The number of axles, the performance and the rotation of the UGV wheels is very important. Because navigation, especially in difficult terrain conditions, provides the UGV with the ability to rotate properly and move effectively. This has a significant effect on increasing the accuracy of routing. Therefore, the second element is the combination of the 4-wheel and 4-performance drive suspensions shocks that provide outstanding grip and stability even on difficult terrain. The turning radius of vehicles is also fascinating, thus it can navigate even in space. The minimum rotational radius must be determined as follows with respect to the maximum angle of rotation.

$$Radius = \frac{wheelbase}{\tan\theta} \tag{1-2}$$

Figure 1-2. different kinds of UGV

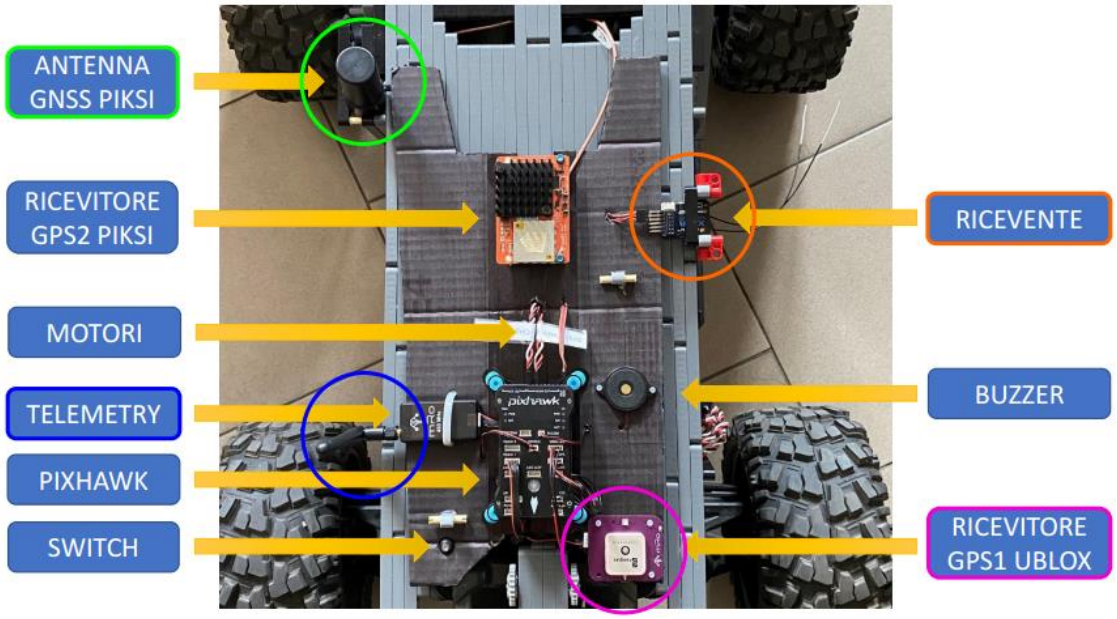


Figure 1-3. View from above the internal anatomy of UGV components.

Initially, the car depicted here was not ready to attach sensors and thus several modifications were performed. The first was the dismantling of the upper body and its support framework to achieve a naked design. Those two additional smaller plates were placed on the front and back axis, respectively, to be utilized as a basis for the installation of sensors and the flight controller, as illustrated in Figure 1-4. This setup, although not final, allowed us to have a comfortable supporting surface so that the sensors were more freely adjusted and a certain space was provided for the future installation of additional sensors to enhance the truck. The top plate attached to the back axle is a noteworthy feature since it gives room to fit the antenna of the GNSS module which is detailed in this chapter's hardware section. The antenna is put in an elevated location that promotes the reception of satellite signals. The position in which the points of way are determined in the algorithm corresponds perfectly. Therefore, it is recommended that a custom-

made fiberglass board be used as a basis for the installation of the sensors and the flight controller should be built on both the front and back axles, with two additional smaller boards attached to the board (Figure 1-4). Although not definitive, this configuration allows us to have a comfortable supporting surface in order to configure the sensors in a more free manner and to install other sense sensors in the future to develop the car. A particularly interesting element is the top plate installed on the rear axle since the antenna of the GNSS module is mounted in some room and will be detailed in-depth in the hardware portion. This antenna is positioned in a higher place that favours the receiving of signals and perfectly coincides with the position where the paths of the algorithm are computed. Then a number of holes were drawn on the ground plate to allow the power, steering and throttle control cables and on the rear top plate to pass through to fasten the antenna of the GNSS module.

Figure 1-4. Vehicle with its tradition made fiberglass support plate.

To move the vehicle, some hardware components that are compatible with the needs of the project (information/instructions provided by the algorithm) act as an intermediary between the software part and the vehicle actuators. It is intended for the assembly of hardware components, their installation and adjustment of their basic parameters. So only the main aspects will be reported. The same software guide suggests different controllers, regarding software choices known as Ardupilot's Mission Planner as a ground control station. Other accessories needed for navigation یue to plug & play compatibility, some were recommended by the guide. In this thesis, the sample used in the UGV networks plan form is shown in Figure 1-5.

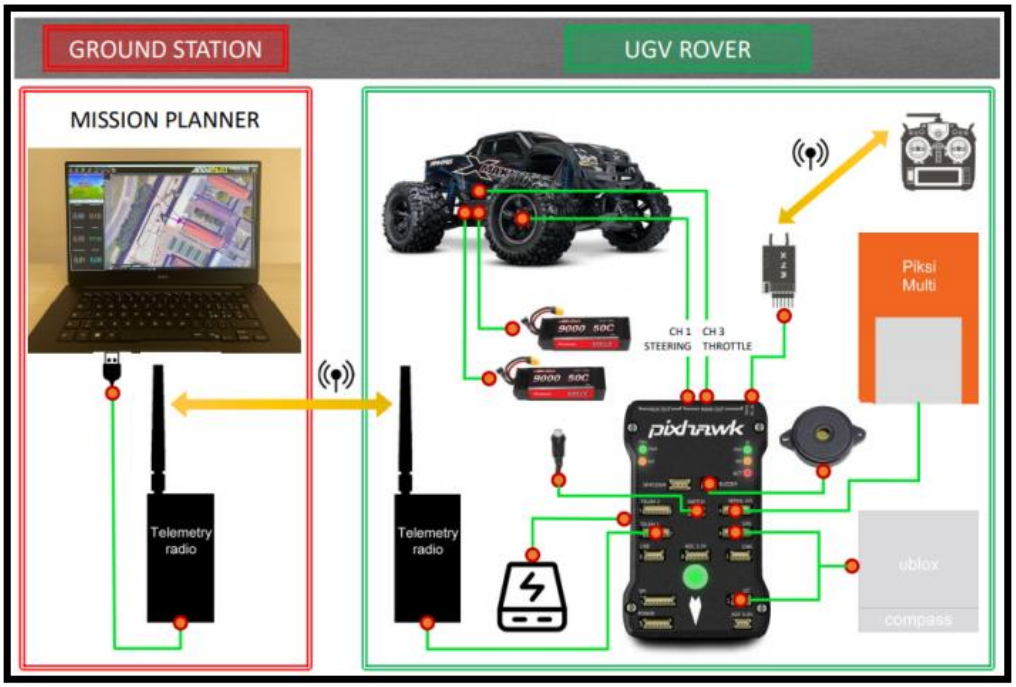


Figure 1-5. UGV networks plan [2].

1.4. Sensor application for UGV

Common systems comprise RADAR (RAdio Detection And Ranging), LiDAR (Light Detection And Ranging), and LADAR (LAser Detection And Ranging). Although their names are diverse, they all transmit and receive electromagnetic energy while operating in diverse frequency bands [50]. A LiDAR is an active mapping system that actions the distance to the goal by revealing the aim with a pulsed laser beam and recording reflected pulses. One of the most common is called LiDAR pulse or LiDAR linear mode, which releases a very short but penetrating pulse of Time of Fly (TOF) from the moment the pulse is emitted until the signal is reflected by the sensor Received records [51]. The range can be calculated by the speed of the pulsed laser beam, which is identified. Normal linear mode LiDAR releases Gaussian laser pulses. Reflected pulse waveforms are utilized to detect various obstacles. If the width of the laser pulse is also wide, several waveforms are mixed as one. So, the resolution of the linear range in LiDAR mode is restricted with the width of the laser pulse. Though LiDAR linear mode is presently the most developed technique of measuring light [1], LiDAR continuous wave (CW) is attracting increasing attention. LiDAR CW does not release a pulse but releases a continuous beam of laser radiation. By measuring the phase variance between the conveyed and received beams, the range can be calculated. Because the wavelength of a sine wave period can be very short, LiDAR CW can attain more accurate measurement outcomes. Based on incessant wave expertise, the LiDAR Continuous Wave Frequency Modulation (FMCW) was conceived. FMCW knowledge is best designated as "modern

optical radar simulation in performance", so it is occasionally abbreviated as "LADAR" [30]. When scanning, the system continuously illuminates a goal. After getting the reproduced continuous wave, the system splits the reproduced signal into the measured signal and the carrier signals. Compared to the unique reference signal, the variance in phase angle between the reference signal and the reproduced signal is found, therefore defining the frequency of the development note frequency [30]. To rise the resolution of the range, the signal system conveys a constant frequency of waves. The conveyed signal goes up and down sequentially after inflexion, respectively. Later the band resolution in FMCW LiDAR is measured by the maximum frequency and therefore the straight wavelength [30]. The resolution of the FMCW LiDAR is restricted by the controlled signal bandwidth. In summary, FMCW LiDAR has numerous main advantages over pulsed LiDAR [30, 52]:

- ✓ Collect Doppler images by changing Doppler;
- ✓ Measure more range with less power;
- ✓ More resolution.

Although the FMCW LiDAR outdoes the LiDAR linear mode in some respects, it cannot substitute the LiDAR linear mode due to its performance. For instance, FMCW LiDAR needs more time to achieve scanning [30]. The FMCW LiDAR processing unit is also higher and surges the size, weight and power required by the system [30]. We have to choose which mode to employ based on the precise goals of the program. For instance, for automatic driving, moving object recognition and speed measurement utilizing frequency variations with CW are faster and less prone to fault than numerous pulse jumps measurements with LiDAR pulse scans due to the Doppler impact. When scanning a moving goal, the amplitude and the Doppler impact both change the frequency, resulting in uncertainty in the measurement between range and velocity. The uncertainty is detached by a sequence of crickets moving up and down, where the Surface Acoustic Wave (SAW) bandages are just one of them [53-54].

1.5. Sensor modalities and data representations for UGV

The representation of data refers to the format in which the sensor data is kept. For effective storage and recovery, certain data have various representations. In terms of packing, algorithm easy analysis and deference speed, each data format has its own advantages and disadvantages.

1.5.1. Camera

A camera delivers visual information about the environment by collecting the incident light into pictures or image sequences (video). Compared to other LiDAR, RADAR and stereo cameras it is widely available and reasonably cheap. It is available. It offers rich texture and color information in real-time (15Hz - 120Hz), however, it does not provide detailed information about the vehicle. An image picture frame can be shown in the form of a dense 2D array with a matching pixel intensity in each element(pixel) of the array. The data of the pixels may be represented as

RGB, HSV, or greyscale. Inferior 3D geometry from 2D camera data is hard because of a lack of depth information.

1.5.2. LiDAR

Multi-object tracking is classified based on detection-based tracking and non-detectable tracking based on object detection before initial tracking. Detection-based tracking techniques need the detection of objects before tracking, while non-detectable tracking techniques need several moving objects as input. The detection-based tracking technique is more general because it does not need preceding data about the number of objects. Model-based or non-model-based techniques can be utilized to distinguish an object from point clouds. Model-based methods recognize objects according to the previous model data. Such methods are favoured when the intended objectives have been identified and can therefore be pre-modelled. For instance, some procedures use foot signatures to locate pedestrians. Adopts multiple hypothesis tracking with adaptive occlusion probabilities to track 2D leg signatures in laser range data utilizes a supervised learning method with AdaBoost to train a classifier on features of legs in 2D range data to detect people. The detected individual persons are tracked with multiple hypotheses tracking as well focuses on tracking vehicles, in which the object's geometry is approximated with a rectangular shape of width and length. Besides, a vehicle dynamic scheme is presented which adopts the velocity evolves via the addition of random bounded noise and the pose changes by linear gesticulation. In place of using discriminative detectors, developing a generative model with the capability to detect and track a large number of object classes of varying sizes and shapes.

1.5.2.1. Sensor data collection procedure

For a machine to read real-world truth, there must be data to process. Real-world data collection is usually done with sensors, and the type of sensor determines the characteristics of the data generated. Providing data by sensor consists of a set of coordinates, each return typically containing characteristic values to which the return is associated or to the pulse from which the return is generated, as below:

- **Pulse density:** this is a straight function of the footprint distance on the assumption flat the surface as pulse density = 1 / (footprint distance^2). This is the most constant measurement of the two-dimensional determination of the sensor dataset also used in the current thesis;
- **Return density:** this is the public term utilized to define data sets and is frequently disordered with pulse density. This average number of returns in the available data set per unit area is typically 1 m^2. using the exemption of single-return schemes, the return density is measured by the stipulations and procedure of a sensor and by the scanned goal. Assuming that all other characteristics remain constant, the density of return created by a system capable of returning four square meters at the position of an object is greater than the density created by adjacent pastures, as shown in Figure 1-6. Since in the last circumstance, approximately all returned energy is in a quantum. As of this scene-dependent difference, users have to set the lowest pulse density for an assumed capture instead of a return density;

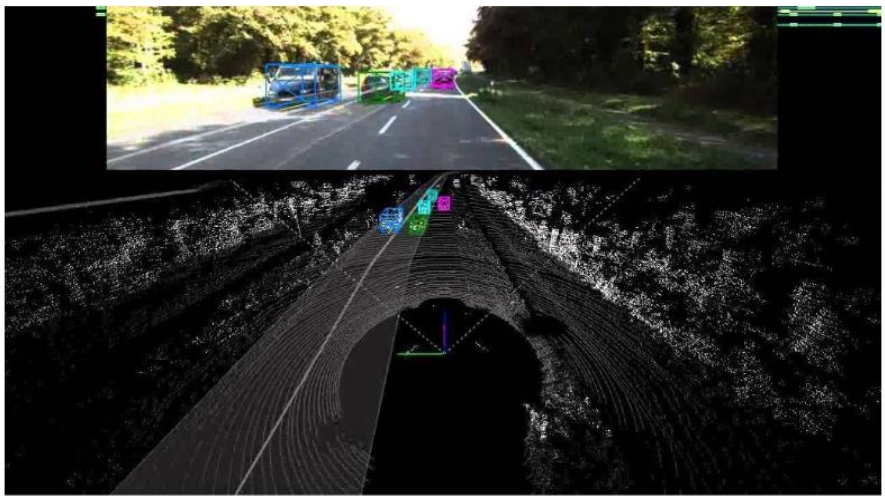


Figure 1-6. False-colour digital inflight photo and corresponding grey-scale raster of LIDAR

- **Return intensity**: this is a feature that defines the scattering power of the beam at the desired return. It depends on the characteristics of the goal reflection and can therefore possibly be utilized to discriminate against the goal. Due to the dependence on the effects of two-way reflection spreading, the distance to the sensor device, the whole number of returns specified in the parent beam, the returned rank), its application to object classification is reduced in the main beam and the receiver gain factor. The latter term defines the scalability of the receiver's compassion to avoid hardware damage if an extremely large amount of scattered energy is received, which may happen with high reflective goals. Such a decrease in sensor sensitivity is virtually prompt. Reverse scaling, a rise in sensitivity in the attendance of a persistent scattering of weak energy, typically takes a few seconds. The attendance of a single, high-reflection, scanned goal in a straight line can lead to a significant difference in the average intensity of the return at the overlap of two head-to-head moving lines;
- **Return number:** this denotes the efficiency rank between those produced by a beam. This only makes sense for schemes that provide multiple efficiencies per beam. The number of returns should not be disordered with the number of returns, a beam property**.**

Features that a return receives from its original beam comprise the scan angle, which is typically documented in degrees. The scan end line is a binary property that designates whether the main beam has noticeable the edge of a scan line, and items that are occasionally determined at the data processing stage, such as freight line indicators or classification systems, show the exact pulse propagation time. Accurate enough to save GPS time, this feature can be used as a unique identifier for a pulse.

LiDAR sensor used in this thesis is developing remote sensing expertise with auspicious possible to help map, monitor and evaluate different objects and resources. Compared to analogue or digital optical remote sensing, LiDAR proposals palpable benefits, counting almost complete recording of spatially spread information and the capability to enter the perpendicular profile of an area covering and control its structure. LiDAR has been utilized in numerous fragments of the

world to effectively measure the elevation and size of single or base trees, to guess canopy closure, capacity, wildlife habitat assessment; and to determine the sensitivity of standing to fre. The LiDAR sensor, which functions from the first stage, includes a set of tools: a laser device. The laser sensor submitted light pulses to regulate the range up to a distant aim. The distance to the goal is resolute by precisely measuring the time lag among the pulse propagation and the detection of the reflected signal. In topographic and mapping submissions, the pulse wavelength in the near-infrared portion of the spectrum is typically between 1035 and 1060 nm. With the LiDAR wave shape, the energy returned to the sensor is documented as a continuous signal. Using LiDAR with discrete and small steps, the reflected energy is measured at amplitude intervals and recorded at exactly the indicated points in time and space. Common substitutes to the time "point" comprise "return" and "echo". The energy variety related to each reappearance is known as intensity.

The main operational stipulations of the LiDAR sensor followed in the current thesis are described below:

- **Scanning frequency:** this is the number of pulses submitted by the sensor in one second. Older devices submitted several thousand pulses per second while up-to-date devices support frequencies by 167 kHz. Occasionally they can function at frequencies underneath the highest, usually 100 kHz, but rarely at short frequencies, such as 10 kHz. The sensed frequency is straight associated with the achieved discrete efficiency density;
- **Scanning pattern:** this is a two-dimensional plan of pulse efficiencies expected from a surface and depends on the instrument utilized to straight the pulses in a straight line. In these outlines, the pulse is guided by a wavering glass along the scan surface, and the output is incessantly generated in the directions of the scan. Although this outline is planned to keep the distance between the returns, in practice the pulse density is not uniform and due to the slowing down of the glass, the returns "accumulate" at the end of the swat. The non-uniform efficiency gap can be reduced to some extent, but not eradicated using galvanometers;
- **Beam divergence**: Contrasting an actual laser device, the path of the photons in the beam submitted by a LiDAR tool diverges somewhat from the emission line (axis) and creates a narrow cone instead of the archetypal thin cylinder of real laser schemes. The stretch beam divergence denotes a surge in beam diameter that happens as the distance between the laser device and the plane increases the axis of the beam growths. Typical beam deviation sceneries are between 0.1 and 1.0 mm, as illustrated in Figure 1-7. Since the entire volume of pulse energy leftovers continuous nevertheless of the beam divergence, at larger beam divergences, the pulse energy is spread through a greater zone, reducing the signal-to-noise fraction.

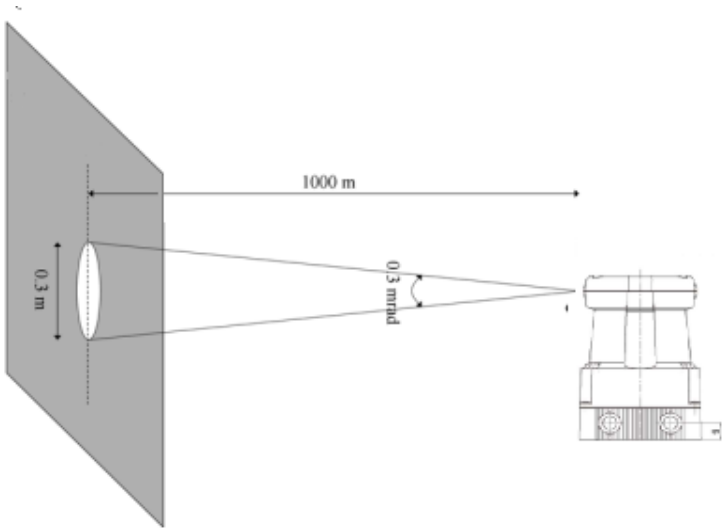


Figure 1-7. Image of LIDAR beam divergence is drawn at horizontal and vertical distances at different scales

- **Scanning angle:** This is the angle at which the axis of the beam moves away from the "focal" plane of the LiDAR device. The combination of scan angle and altitude determines the scan level followed in the current thesis is shown in Figure 1-8.

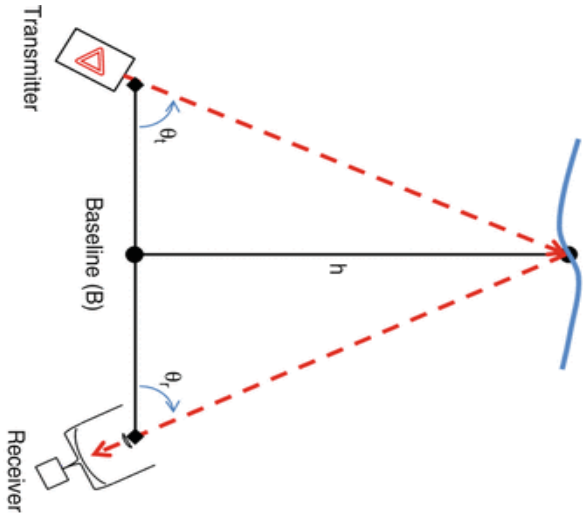


Figure 1-8. Image of LIDAR information scanning feature. The aircraft is assumed to run parallel to the ground and there is also a chainsaw scanning pattern

- **Footprint diameter:** this diameter is a beam that is cut by a plane vertical to the axis of the beam at a distance from the tool to the object. Therefore, it is a function of the divergence of the beam and the distance from the goal. The pulse energy spreading is not uniform in the amount of footprint. It declines outward from the centre and can be approached by a two-dimensional Gaussian spreading.

- **Pulse length:** this pulse time is in nanoseconds. Together with the discretization surroundings, it determines the determination of the pulse series in several arrival schemes or at least the distance between sequential returns of a single pulse.
- **Number of returns:** this is the highest number of different returns that can be removed from a single beam. Some schemes can detect the first or last arrival. Most up-to-date schemes can detect multiple efficiencies from a single beam.
- **Footprint spacing:** this is the nominal distance between the centres of successive beams along and between the scan lines, which together with the beam deviation regulates the three-dimensional determination of the LiDAR information. Footprint distance is an occupation of scanning frequency, altitude and speed.
- **Discretization settings:** These are integral features in the scattered energy processing of a pulse to recognize different efficiencies. They are arrangement and proprietary and are occasionally denoted as digitization settings. They control the smallest amplitude of energy vital to generate a return and, along with the pulse length, regulate the minimum distance between consecutive earnings of a pulse. Up-to-date devices can procedure the energy scatter of a beam and detect up to six earnings, but most only support up to four. Ideal settings for mapping.

CHAPTER 2

OBJECT DETECTION IN UGV NAVIGATION

2.1. Introduction

This chapter presents the theoretical features and basic concepts of using the sensors that are the basis of this dissertation. This chapter begins by describing the benefits of sensors, how to use them, and a used methodology in this thesis. In general, this chapter explains the principles of object recognition evaluation methods.

2.2. Object Detection

Scientists investigate different phenomena by analyzing, forecasting, and identifying patterns in the world. Fundamentally, to comprehend the environment, computer images could play an effective role in controlling the possible changes. Therefore, object detection is a way that is thoroughly associated with computer visualization. This denotes the classification and localization of objects of interest in an environment. The localization of items in the framework denotes the production of constraint boxes that effort to cover each of the items. Perfectly, a limiting box must be as insignificant as possible while holding the whole object. In addition, each limited box produced has a ticket that denotes the arrangement of objects. The object recognition problem is partially a deterioration problem in terms of locating and sizing the range box and is somewhat of an arrangement problem in terms of labelling each of the finite produced boxes. In this among, deep learning methods are suitable for object recognition work [38], [27], [39], where a deep neural network learns to excerpt structures from the input data and, based on the features, restricted boxes labelled with the projected label. The overall flowchart of the proposed detection-based tracking technique is illustrated shown in Figure 2-1.

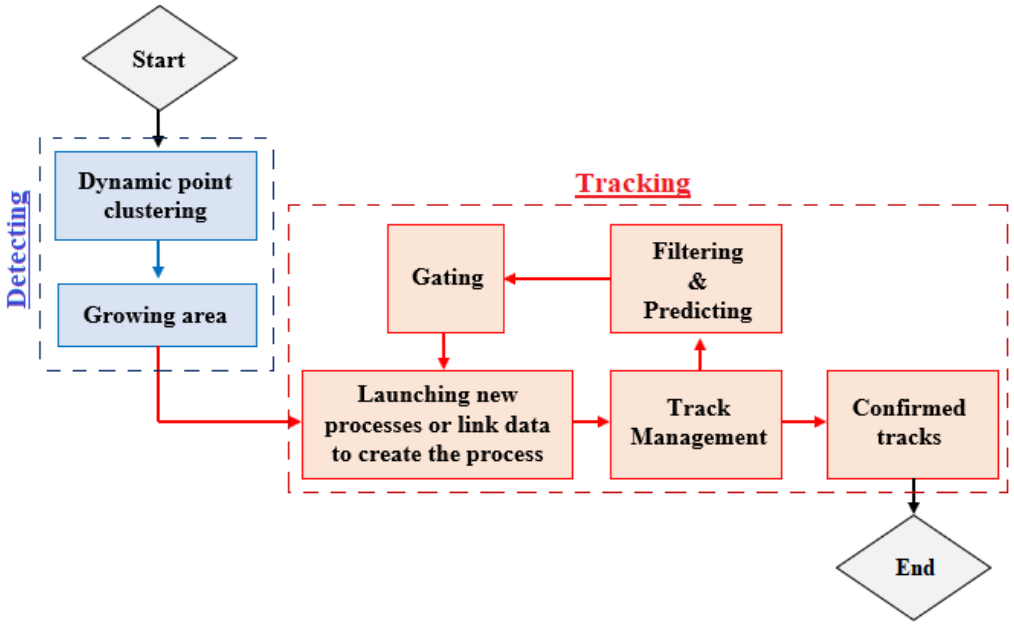


Figure 2-1. Overall flowchart of the proposed detection-based tracking technique

The class of deep learning object recognition techniques can be divided into five parts: feature extraction, regional proposals, classification, regression, and prunin, as described below in detail:

2.2.1. Feature Extraction

One of the well-known kinds of data for object recognition programs is images. Understanding a computer is a challenging task to measure the values of an image, which for the computer is only a matrix of variables [12]. In particular, image data varies dependent on varying parameters such as lighting circumstances. The solution is to acquire the general structures from real data and try to find some images which split the object of interest from the background and other objects. Structures inside an image are usually signified by pixel chunks. Traditional approaches such as Histogram of Oriented Gradient [9] and Scale Invariant Feature Transform [28-29] by creating the colour of the pixels in the patches, create structures for the furthermost variations known as the directional gradient. In the case of in-depth learning, structures are usually learned using convolutional neural networks. These kinds of grids can weigh in on specific features in patches and represent more abstract features. Networks are trained by showing samples of their hypothetical inputs and outputs. Finding the features in the input is vital for making the anticipated decision. By providing sufficient examples, preferably, the network can distinguish one class from another in general with the use of educated structures.

2.2.2. Region Proposals

One problem with object recognition is that the number of things in a location can diverge, meaning that the goal output can differ in terms of size. There are some ways to resolve this problem. One of the more characteristic approaches is named a descending window, which includes, as the name implies, dragging a window onto the input image and removing slighter units, in which each unit is accomplished as an area suggestion. This technique controls objects of various sizes and scales by grading the input image and then re-running the slider. Proposals in the created area can be taken as finite unsystematic boxes, which basically enclose boxes for the model rather than whether they comprise substances.

2.2.3. Classification

Each of these suggestions in the above area (constraint boxes) is subject to the organization to determine if the constraint box contains an item. Classification is completed either over a distinct organization scheme or as an essential portion of the area suggestion model. General arrangement for each class is likely to exist in the projected range boxes. Classically, an extra class is added with reference to the background to remove the restricted boxes with no items in them.

2.2.4. Regression

Additional regression is usually used because the generated area bids are not expected to enclose items as much as possible. The theory is performed as a classification or individually or

as a more integrated part of area propositions. The purpose of regression is to eventually tauten the proposed ranges, with compensated values, for better storage of objects.

2.2.5. Pruning

Each of the limiting boxes produced by the model is usually pruned. The most commonly used way to do this is called Non-Maximum Suppression. The non-maximum suppression includes the removal of all constraint boxes whose output is likely to be classified below a specified threshold and removes forecasts that the method does not believe are items. Restricted boxes that point to the same object are similarly eliminated. This is done by scheming the overlay between the boxes, and if the overlay is greater than the specified threshold, only the range box with the highest arrangement inevitability is maintained.

2.3. Intersection over Union (IoU)

The most common metric utilized in object recognition is the intersection over the union. It is utilized to measure the intersection between two figures and is therefore usually used when corresponding proposed area frames with basic facts because the overlap is approximately that is taken into account in estimates. It does this by separating the overlap part between the boxes by the total zone of the boxes. This purpose could be defined, as Equation 2-1 and Figure 2-2.

$$\text{Intersection over Union } (b_1 \cap b_2) = \frac{b_1 \cap b_2}{b_1 \cup b_2} \tag{2-1}$$

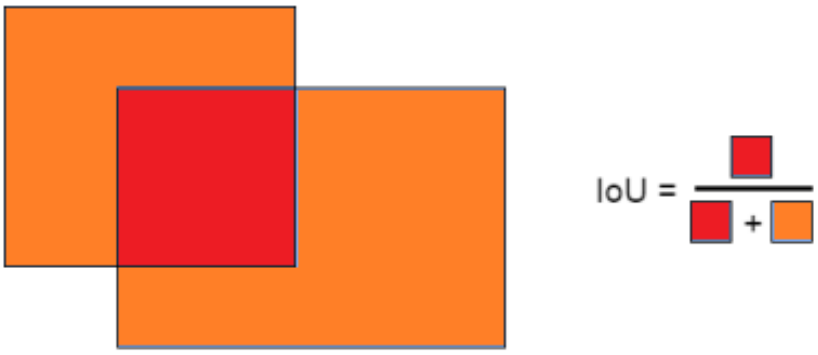


Figure 2-2. Graphic illustration of the overlapping on the union between two rectangles in 2D

Another point is that the boxes do not require an axis alignment. Though, if the alignment between the boxes is different, the actual intersection over union calculation becomes much more complicated. The reason it becomes difficult is that the consistent forms of connections and junctions usually turn into asymmetrical forms with complex parts.

2.4. Hard Negative Mining

Object recognition is not just the task of discovering objects in stated modules, but also the job of finding them. In general, the volume of background in input is much higher than the required value, which means that a lot of background arrangement is done. Any constraint box that is

confidential as a background is named a negative prediction. Neural network training is done by feeding information about the hypothetical results. Therefore, to teach the model of correct object recognition, not only optimistic predictions but also background samples are required. Categorizing whether a restrictive box has a background can be unpredictably difficult, particularly for any item class that is not involved as a precedent. For example, if the model does not follow the tram, all trams are measured background. Negative extraction is a common method for emphasizing hard forecasts. This is performed through training when the technique provides a positive forecast in the default context. Consequently, the programmer put it into the training procedure with difficulty, and thus programmer get a network that works better in properly organizing the background and thus in properly distinguishing between items and the theoretical context.

2.5. 2D Object Detection

Methods for deep learning to detect 2D objects can be classified into two categories:

Detectors with a single stage and two stages [42]. During the first stage of the two-stage method, features are extracted and Regions of Interest (RoI) are proposed. Afterwards, probability scores and final bounding boxes are calculated for each RoI. Single-stage methods, however, usually do not generate ROI. Rather, classification regions and regression regions are selected deterministically in advance. Faster Region-based Convolutional Neural Network (Faster R-CNN) is among the highest-performing SOTA two-stage detectors. Overall, two-stage methods are more accurate compared with one-stage methods. Nevertheless, the training and the actual inference processes tend to be slow. Regarding one-stage methods, SOTA has become popularly known as the "You Only Look Once" method (YOLO) and the single-shot multibox detector (SSD). Compared to the two-stage method, these methods are less accurate but faster. Considering how important speed is for AD, these types of methods are particularly suitable for the thesis. As this section begins, it will explain the fundamentals of Faster R-CNN, followed by a discussion of YOLO and SSD, two-stage detectors.

2.5.1. Faster Region-based Convolutional Neural Network (Faster R-CNN)

The previous section described how region proposals were originally generated using the sliding-window method, wherein each boundary placement was regarded as an individual input image for the classifier. The R-CNN paper by Grigorich et al. [15] introduced a more practical approach. A method was developed using a selective search algorithm [44] to remove region proposals without effective objects within them. In simplest terms, the selective search algorithm involves grouping similar portions of the input based on colour segmentation. The same authors have improved the algorithm further in two iterations, first by releasing Fast R-CNN [14]. Comparing the original release to R-CNN, the performance has been significantly improved. In order to speed up the process, features were extracted from the input once rather than extracting features from each region proposal independently. A process called ROI pooling is used to obtain equal-sized feature maps for each region proposal. Each region proposal is turned into a large

feature map by using ROI pooling, followed by maximum-pooling (varying in size and stride depending on region size), therefore resulting in feature maps of set sizes for all region proposals. According to the latest version of the model, Faster R-CNN [39], Gorchich et al. suggest Region Proposal Network (RPN) as a means of creating region proposals for the network under training. This RPN obviated the computational bottleneck of the selective search method used in previous iterations, producing the complete (simplified) network shown in Figure 2-3. An RPN extracts a feature map from an initial CNN by applying the sliding-window method to it.

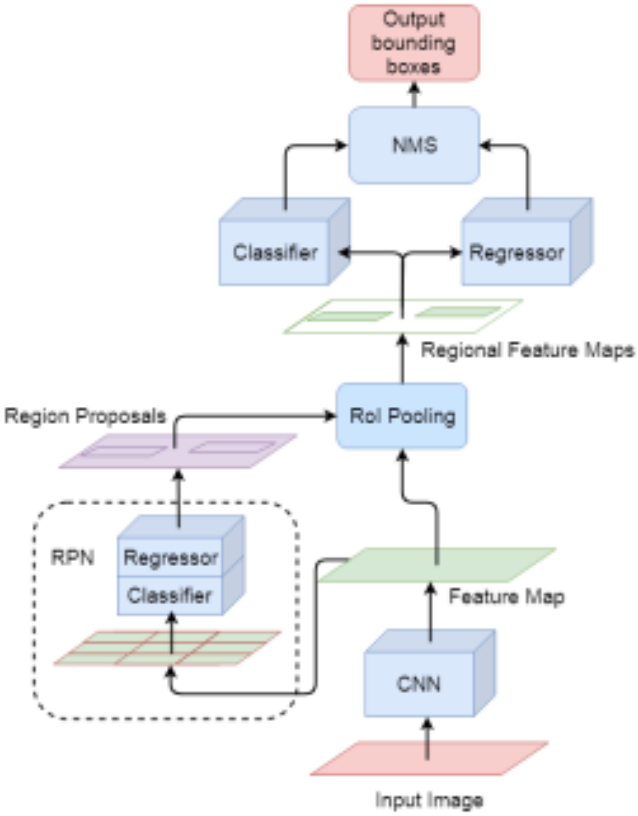


Figure 2-3. The Faster R-CNN method illustrates the simplified flow from an input image to the bounding boxes.

Regarding Figure 2-3, CNN produces feature maps based on input images. The generated feature maps are sent to RPNs generating region proposals. By pooling the ROIs, the feature maps are combined with the region proposal, and the resulting feature maps are reclassified and bounding box offset regressions are carried out to determine the bounding boxes. Following the production of bounding boxes, the NMS prunes them to produce the final outputs. In addition to the region proposal at each grid location, the network generates multiple bounding boxes based on the predetermined bounding boxes sizes, as illustrated in Figure 2-4. Predetermined bounding box sizes called anchors are assumed to be the default size for each region proposal. Therefore, RPN creates a total of W ×H ×k anchor placements, where W and H refer to the width and height of the feature map, while k is the number of anchor boxes. It is necessary to place additional larger anchor boxes so that RPN can handle objects of different sizes. Moreover, the RPN uses a simple classifier

to predict whether each of the initial anchor placements contains an object (binary classification). Using this simple classification, it could be determined whether to keep or remove the anchor placement. Furthermore, the bounding box placements generated from the positive anchors (the ones predicted to contain objects) are refined by a simple regression algorithm.

In the next step, the ROIs are merged with the features map by RoI pooling, resulting in the regional feature maps. A final classifier and a regression coefficient are applied to each regional feature map. As a result, the classification probabilities for each class are produced, with a background class that eliminates region proposals. Similar to the RPN, the last regressor attempts to refine the bounding boxes (location and size). Refinements include attempting to estimate the difference (size and position) between the suggested and actual bounding boxes.

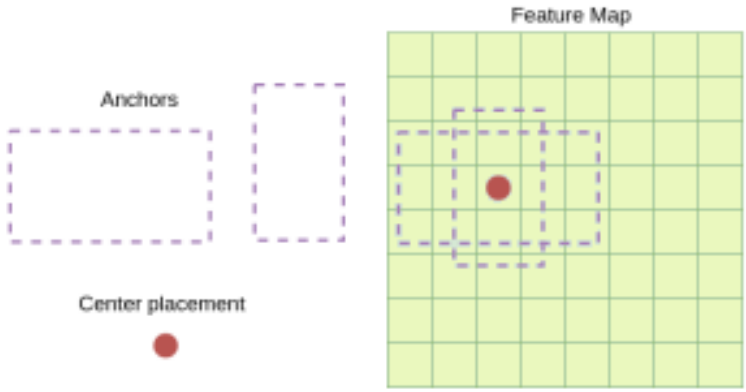


Figure 2-4. There is an example of how to create regional proposals initially using an RPN.

To perform training, target output should be created, which is produced by matching anchors with ground truths. Ground truth is matched with the anchor where it is placed and shaped closest to it. The offset and class values are filled into the matched anchors, while the unmatched anchors would have the background class applied, which create the supposed targets. Multitask loss[14] uses the outputs from the network together with the target outputs for its loss function, which is used to train the model.

2.5.2. Single Shot Multibox Detector (SSD)

In the SSD model rather than attempting to first propose regions that are then used for predicting bounding boxes as in the two-stage models, the SSD model by Anguelov et al. uses one deep neural network to predict bounding boxes directly [27]. Unlike two-stage methods where the different stages typically require multiple training phases, the SSD incorporates all computation into a single, integrated network, which makes it significantly easier to train. As shown in Figure 2-9, the SSD model places anchor boxes of default size on the produced feature map, similar to the RPN model. In SSD, features are not taken directly from the CNN extractor to produce the feature map. As a result of combining the features of multiple layers in the CNN architecture, the feature map can be created. In this way, the model can detect objects of various sizes by placing anchors at different scales of the feature map, as can be seen in Figure 2-5. Feature maps contain

a grid where anchor placements are used. A feature map with $W \times H \times k$ anchor placements, W and H being the width and height of the map, where k is the number of anchor placements.

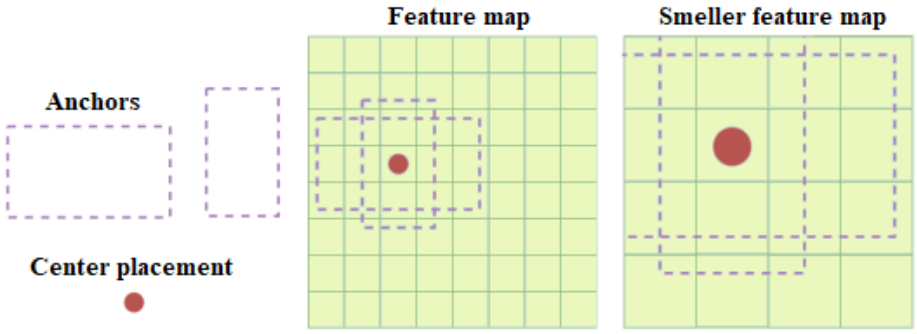


Figure 2-5. The SSD model places anchors in two feature maps of different sizes.

Anchors are bounding box predictions for two different scales given overlaid with the original input image. Different implementations extract different features at different scales, and the number of feature maps varies. A simplified version of the SSD flow is shown in Figure 2-6 along with three levels of feature extraction. The output 3, For each grid location in the feature maps, the SSD model yields Related Work in the form of k * (B + C), The anchor number is given by k, the bounding box number is given by B, and the class numbers are given by C.

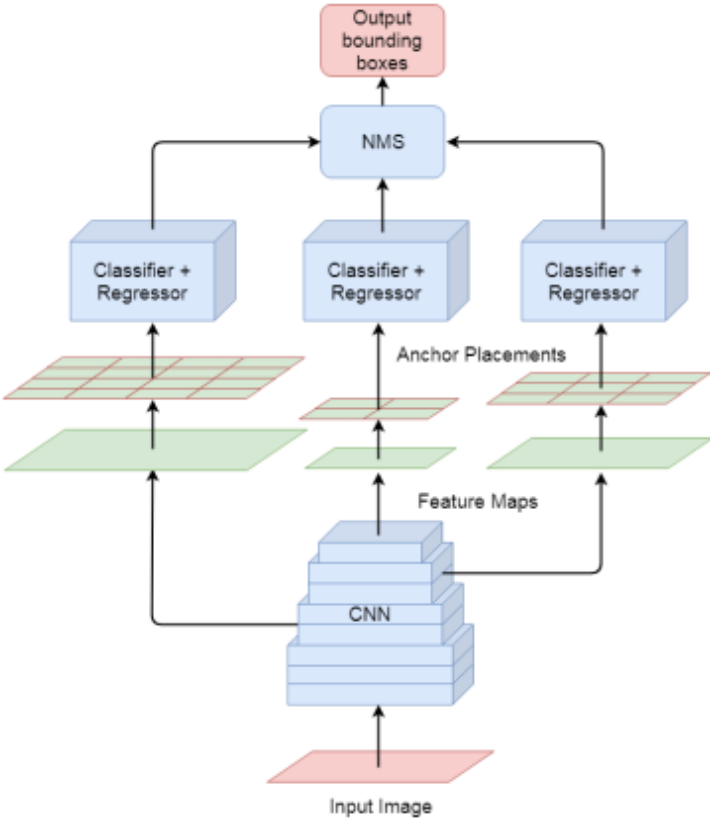


Figure 2-6. Showing how SSD adapts the original CNN architecture to extract feature maps at three different levels using the SSD model, simplified.

As shown in Figure 2-6, an anchor box is placed on each feature map, and the anchor boxes pass through a convolutional layer, generating classification and regression values. Then, NMS pruning is applied, which generates the final outputs. In a similar manner to Faster R-CNN, Anchor boxes are designed to match ground truths to generate targets. In SSD, ground truths and anchor boxes are matched by calculating the IoU between them. Overlapping data above a certain threshold is considered a match. Therefore, matching does not occur in pairs, meaning that the model can predict several boxes at once, which is typically pruned in the postprocessing stage. As a result of this simplification, the generated targets, along with the actual outputs from the network, are used to optimize the loss function.

2.5.3. You Only Look Once (YOLO)

In 2016, Redmon et al. released the first version of YOLO [36], it used the same concept as SSD, one pass through one unified network. The bounding boxes of each grid location are predicted based on the produced feature map Related Work. According to the YOLOv2 [37] done by the same authors, predictions are introduced as offsets from anchor boxes placed within the grid locations similar to the SS and Faster CNN algorithms. To make predictions, a feature map is extracted from the input image, using a CNN. Figure 2-7 shows simplified network flow. This shows a great deal of similarity to the SSD model. An output of the YOLOv2 network for each grid location on the feature map consists of the following form: k(B+1+C), with k indicating the number of anchors, B representing a bounding box, C defining classes, and 1 representing the additional confidence score. An additional confidence score reflects the likelihood that an anchor will contain a particular object regardless of the class it belongs to.

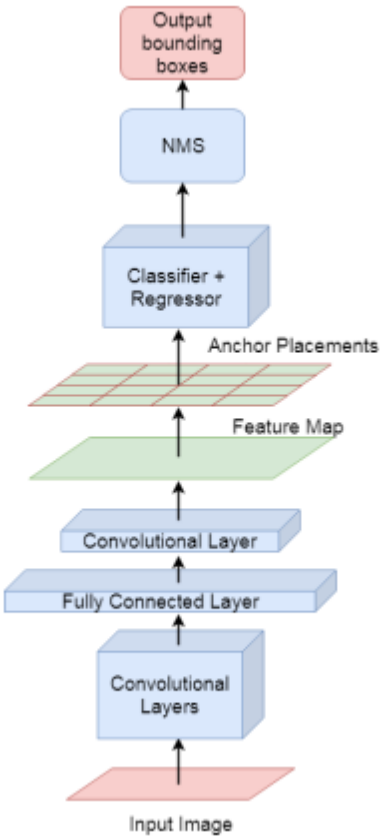


Figure 2-7. YOLO's simplified network architecture. The convolutional layer creates a feature map based on the input image.

In Figure 2-7, Classifiers and regressors are used to yield predictions based on the anchors that are placed on the grids of the feature map. A YOLO method is intended for applications that require a fast execution time. Therefore, the CNN structure devised after YOLO may not have achieved the highest rate of accuracy in these models, but low inference time for 2D object detection did reach its limits. Further improvements were made to the YOLO-series with the YOLOv3 model [38], which extracts. The CNN structure incorporates multiple layers of related work. By using feature maps of different scales, better accuracy regarding the prediction of smaller objects, which was one of the main problems in the previous versions of the YOLO series.

Due to how the encoded point cloud is structured, it can be interpreted as an information-rich dense image with a depth equal to the number of features output from the point feature extractor for each pillar. This makes it possible to effectively process the entire feature map by 2D object detection methods. The decoder used in the original PP implementation was a modified SSD. The modification added regressional targets in the shape of height, z position and direction, along with an additional binary classification for the direction. The purpose of the directional classification is to be able to predict the heading of an object in relation to the matched anchor, thus being able to more effectively use the anchor boxes original orientation as a point of offsets.

All the followed algorithms in the thesis were performed in ROS. Unlike the name indicates it's not an operating system, ROS or Robot Operating System. ROS is a middleware that serves as a bridge between applications and operating systems. ROS is designed to improve the architecture of the robot's hardware and software. ROS is a simple yet elegant message packaging and transport service that enables sensor data to be sent onto more than any other process requiring that robot control data. Here is some terminology needed to understand ROS:

- **ROS Node**: The code or software is executable and runs behind the scenes. The publisher, subscriber, or both may include the ROS 30 node. ROS is designed in a node unit which is the executable minimal unit that has decayed for maximum reusability. The node exchanges data with other nodes employing messages building a whole big program. The main notion here is the techniques of message transmission between nodes. In addition, you may change the settings in the node from the external node. This may also be seen in the wider perspective as a form of message transmission. Each theme, service, action and parameter must be used for their right purpose for ROS development. Figure 2-8 illustrates the transmission of messages;

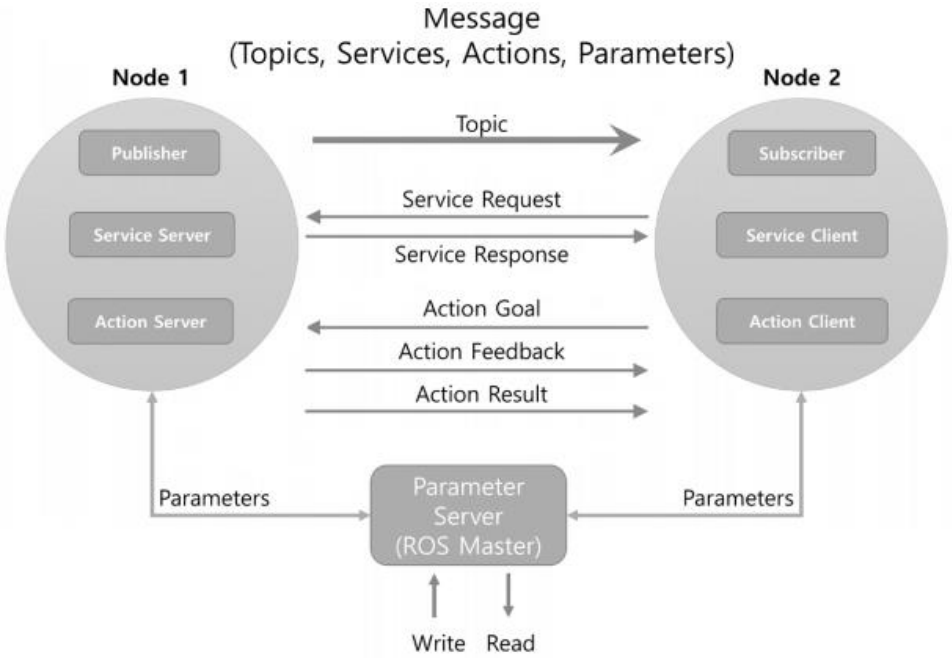


Figure 2-8. Message nodes communication

- **ROS Topic**: A subject of ROS may be seen as a newspaper column with a committed publisher and a large number of readers (subscribers). For both publisher and subscriber, communication on the topic utilizes the same sort of message as illustrated in Figure 2-9. The subscriber node gets the publisher node information corresponding to the same theme name in the master. The subscriber node is connected to the publisher node directly to receive messages using this information. For example, if an encoder value of both wheels of the mobile robotic is calculated in the form of odometry information the current robot location

may be constantly sent in unidirectional movement via a subject message Asynchronous Odometry Information (x, y, i). Because subjects are unidirectional and stay linked to transmit or receive messages constantly, they are suited for sensor data requiring regular publication of messages. Moreover, a publisher can send several abonnés a message and vice versa. There are also several links between publishers and subscribers

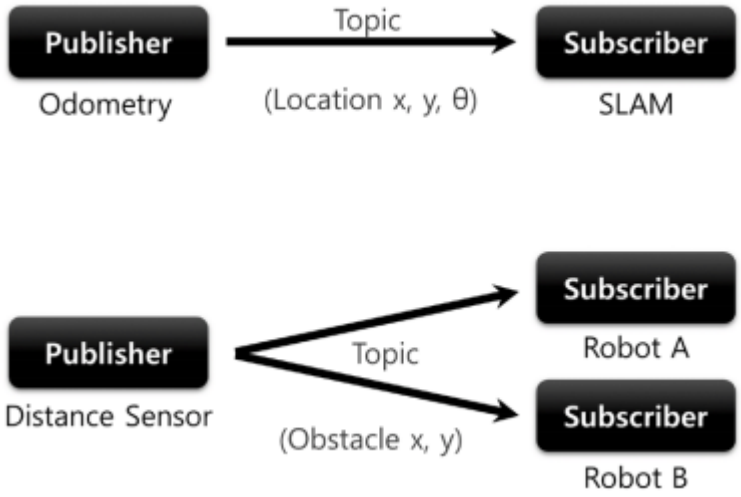


Figure 2-9. Communication of the theme message.

- **Publisher**: A ROS Node can post a ROS message to a ROS subject. The publisher, abonor, service server, service customer, action server and action customer can be built separately. The connection must initially be created with the aid of a master to exchange messages between these nodes. Masterworks as a name server, keeping node names, subjects, services and actions and URI address, port number and arguments. In other words, nodes register themselves with the master when they are started and get the related information from the master of other nodes. Each node then connects directly to the other to communicate the message. Figure 2-10 shows this.

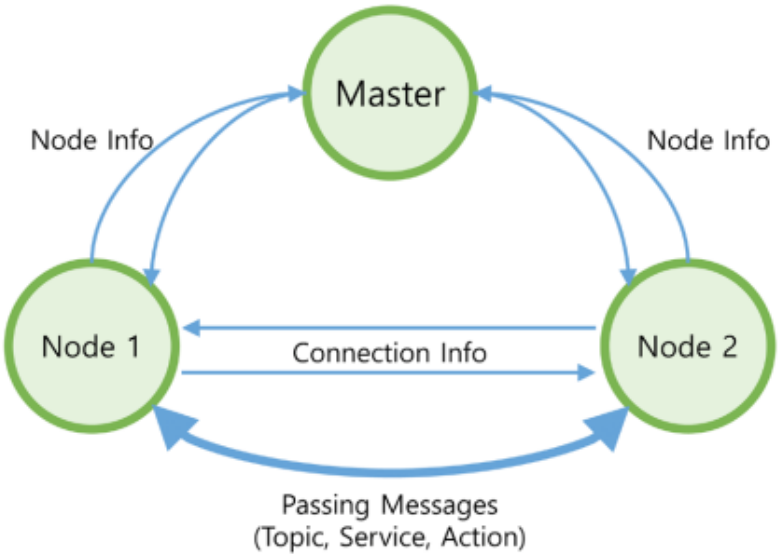


Figure 2-10. Announcement of the message.

Publisher nodes are run using 'rosrun' or 'roslaunch' commands, like subscriber nodes. The node of the editor records its node name, theme name, message type, URI address and master port. As demonstrated in Figure 2-11, the master and node communicate via XMLRPC.

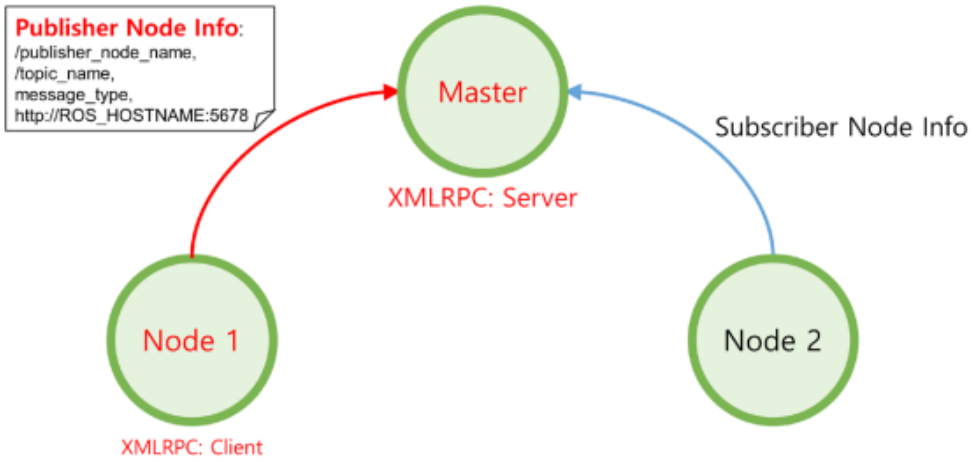


Figure 2-11. The node of the editor.

- **Subscriber**: A subscriber subscribes to a ROS topic and keeps the incoming ROS messages updated. Based on the publishing information from the Master, the subscriber node requires a direct connection to the publisher node. The subscriber node communicates information to the publisher node during the application process, such as the name of the subscriber node, the subject name and the kind of message. The editor node and the abonor node are sent through XMLRPC; as demonstrated in Figure 2-12.

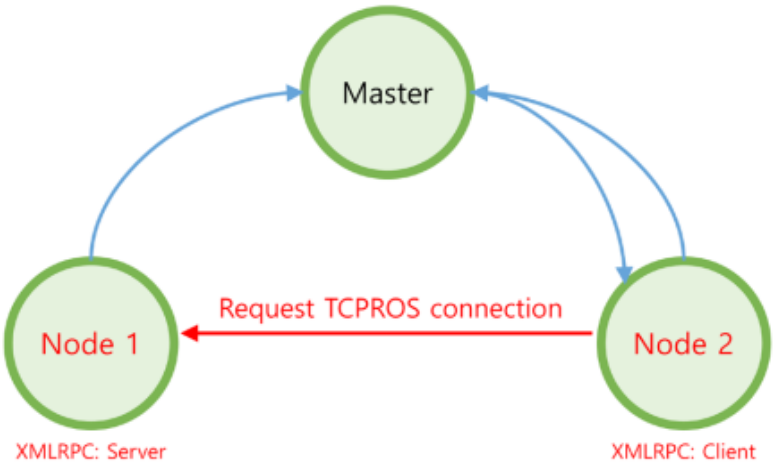


Figure 2-12. Subscriber node connection application.

- **ROS Message**: The Publisher sends a ROS message to a ROS theme. It might consist of sensor readings or parameters of control or any data in any format, as shown in Figure 2-13.

Figure 2-13. Request and Answer for Service.

For the generation of robot models, ROS uses a standard called URDF. URDF is a Unified Robot Description Format that utilizes and continually broadcasts static transformations between frames. ROS creates a transformation tree named tf tree showing the linkages and recent (if any) changes between these frames.

CHAPTER 3

STRATEGY FOR MAPPING GENERATION

3.1. Introduction

As mentioned previous chapter, object detection helps us to find the position of different object and their location. Therefore, in this section, the concept of mapping and the used methodology in this thesis for mapping are presented and discussed. The thesis examines the effectiveness of the Gmapping algorithm in localizing and mapping simultaneously in an extraterrestrial robotic object application. Thus, understanding the underlying theories and structures of the algorithm is crucial. SLAM algorithm was used to develop Gmapping. SLAM is a complex problem, and EKF solutions are robust solutions, however, they can only be used with Gaussian systems. PF-based approaches have been introduced as a way to add to the capabilities of SLAM. PFs are constructed using a mechanism in which each particle contains a map of the surrounding environment and a robot pose. The obstacles are updated every time the robot moves and the map is updated according to the new pose. Furthermore, each particle in the system has its own representation of the map and robot orientation, as well as its own weight. It makes use of this weight to filter out weaker samples and converge to the strongest representation. Consider this as a sort of natural selection process. It is impossible to have an infinite number of particles with a map and a pose if a set number of particles is kept. Resampling is possible when the less likely particles are removed, and the strength of the estimation is further confirmed. There is an extremely high computational complexity due to the significant number of particles, each of which contains a map and a pose estimation. In this marginalization, the overall sample space is reduced, thus optimizing computational complexity, as discussed below in this chapter.

3.2. SLAM

In an unknown environment, SLAM provides a method of autonomously navigating. The goal of SLAM is to map an unknown area by using a robot and then to navigate and localize using the map. SLAM has been developed to operate in complex environments, including those with varied terrains, those with a limited number of landmarks, and even situations in which aerial navigation is required. Hugh Durrant-Whyte and John J. Leonard proposed a first step toward solving this problem [5]. Extended Kalman Filter (EKF) was proposed as a solution to autonomous navigation. The algorithm makes use of geometric beacons as landmarks and their observations are matched with a priori maps using robot scanners. For the robot, in this case, the location of landmarks is familiar to it already. Using this information in an algorithm allows matching sensor data of observed landmarks with their known locations. If the sensors determine that landmarks should be in a different location than they actually are, the robot needs to change its pose accordingly.

SLAM is made robust thanks to the combination of both findings. When a localization solution is integrated with a map-building solution, it allows users to localize within an environment and map at the same time. Based on Smith, Self, and Cheeseman's pioneering work, these developments were built on the first instance of SLAM. Detailed instructions are included in the authors' work on constructing stochastic maps. Using a random process to represent a

constantly changing system would be called stochastic here. There is an explanation of how the map is built, how data is incorporated into it, and how to incrementally improve it. The data from both the robot (proprioceptive) and the environment (exteroceptive) is evidently needed for SLAM to work. There are numerous ways to generate various types of data. In the current thesis, data information was sorted using a LiDAR sensor.

The main objective of this research is to compare SLAM algorithms that utilize an external and proprioceptive LiDAR, camera, IMU and wheel encoder. The external sensors are utilized to identify environmental markers that are then used to create a map, by learning to localize them and inside the map that was constructed. The SLAM algorithm will seek to link these new landmarks with previously seen landmarks when new landmarks have been discovered and processed. Specific features might operate as barriers and barriers as markers. The SLAM system sometimes identifies and maps landmarks in the robot's real-time path planning. Barriers and markers are synonymous with such a planning effort. The SLAM method employs odometry data to evaluate the current location of the robot inside the map, while the robot is continuing to explore and scan its surroundings. This estimate is compared to environmental measurements collected from sensors and to make it easier to rectify if this is not true. The procedure continues until a full environmental map is generated. There are five key components of a basic LiDAR SLAM algorithm: landmark removal, data association, state estimate, state updated, and landmark update. Figure 3-1 explains this procedure. In this chapter, the procedure carried out for GMapping, in the current thesis, is described

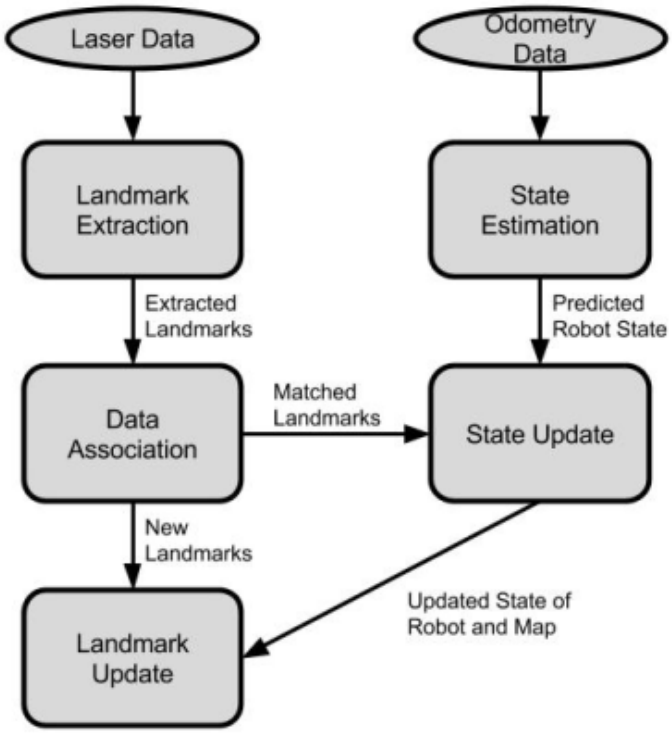


Figure 3-1. SLAM process flow chart.

In Figure 3-1, the state updating step corresponds to the filter updating step. Different SLAM approaches often use different filters in the step of updating the state of the target. The structure is generally the same. This figure shows laser and odometry data as a tool for producing mapping and predicting states. According to Figure 3-1, the following bullet point list, where to define point by point should be carried out:

- **Laser Data:** Data on the robot's surroundings is the initial stage in the SLAM process. It helps to obtain laser data by using a laser scanner. Finally, the laser scanners are incredibly quick to use;
- **Odometry Data:** The odometry data are a crucial part of SLAM. The purpose of the odometry data is to give the robot an estimated position measured by the robot wheels movement. The integrated telnet server makes it easy to obtain odometry data from a robot. A text string can only be sent on a specified port to the telnet server and the server returns the answer. It is tough to get the timing perfect for the dummy data and laser data. When the odometry data is found later, the laser data will be obsolete sometimes t. Since the controls are known it is easier to extrapolate the odometry data. The measurements of the laser scanner might be extremely difficult to anticipate;
- **Landmarks:** Landmarks are characteristics that can be easily replaced and recognized from the surroundings. The robot uses them to see where he is (to localize itself). One method of imagining how the robot operates is to blindfold yourself. Landmarks such as contacting the door frame can assist to assess the location of the sensor. Sonars and laser scanners are a sensation of touch for robots;
- **Landmark Extraction:** After selecting the type of robot operation, it should be able to extract them from sensory inputs on robots in reliable ways. There are several methods for extracting sights, depending on the type of signal being extracted and the sensors used;
- **Data Association:** The data association challenge is that the observed points of reference are matched from several (laser) scans. It also refers to that as markers for re-observation.

The following sections give an overview of the Particle filter and explain other SLAM options developed from their initial concept.

3.3. Extended Kalman Filter-based SLAM

Based on EKF, SLAM was the first robust method of obtaining object location and is the basis for most of the SLAM algorithms that are used today [10]. This process is composed of five key steps. First, landmark extraction. An exteroceptive sensor input (laser scanners, sonar, or vision system) is used in landmark extraction to identify landmarks. Random Sampling Consensus (RANSAC) is the most common method of doing so. A different landmark detection method can be found in [11]. RANSAC identifies a best-fit set of lines from a laser scan, which is then processed into representations of linear features in the environment, such as walls, and uses outliers to pinpoint smaller obstacles. The data in each subset of the sample is selected at random by RANSAC. Based on the small set of data, the algorithm constructs a line fitting model. A

comparison is then made between the model and the entire dataset to see which data fits [4-9]. Outliers are considered data that does not meet a defined tolerance for error. Obstacles are identified by taking outliers. Until there is consensus, the process is iterated. Data association is the second step of SLAM.

By this, different laser scans are taken and a common thread among them is identified. Every scan in the database is taken by data association and compared to previous ones, and any data recording the same landmark is matched up [4-10]. Despite appearing easy, this process can be very challenging. Problems associated with identifying earlier landmarks include incorrectly identifying previous landmarks, falsely identifying landmarks, and incorrectly associating a new landmark with a previous landmark [9-12]. A false landmark association can adversely affect the pose estimation algorithm. In order to generate an accurate map and localize the robot, data association is crucial. SLAM revolves around a concept called the EKF in its last three aspects. An update to the current state of the area, an update to the previously estimated state, and adding landmarks to the current state are the main components of SLAM. In the filter update step, the previous and current states are updated. In this step, the EKF is used to filter noise from the robot's estimated position using the data generated in the previous steps. EKFs remove unwanted components from signals as any filter would. The localization process relies heavily on this step, which sets SLAM apart from dead-reckoning localization [4-9].

To complete the SLAM process, one must complete the map insertion phase. As the last step, the new landmarks are added to the current state of the map. The map is updated with any unused landmarks from the previous steps. Due to their unassociation, the algorithm determined that they were not seen yet. Therefore, their inclusion on the map is necessary. The important thing to note here is that the map becomes more complete as the map insertion phase is repeated. A degree of uncertainty is included in each landmark's position for initialization purposes [8-9]. As a general rule, a map is composed of two parts. State estimates and their variances are these. It also shows whether the positions of landmarks are spread out or vary from the mean and a measure of how the estimates for each number differ from the mean [13]. It is in the map insertion phase that SLAM algorithms complete a complete iteration. The steps of the algorithm are once again iterated as the robot moves and the data from the sensors changes. While simultaneously establishing a map of an unknown environment, the robot is simultaneously able to localize itself within it. Using an illustration of this process, the following diagrams are provided to help with understanding.

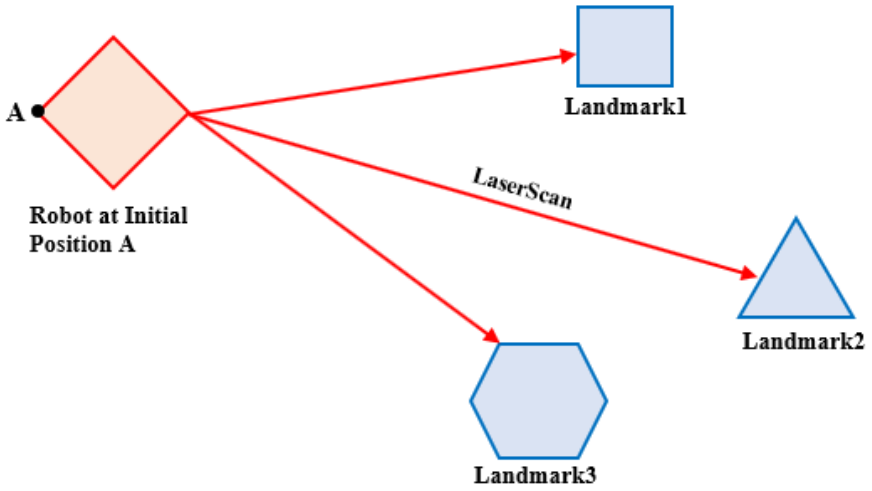


Figure 3-2. Landmark extraction step. Data is collected by scanning the environment by the robot.

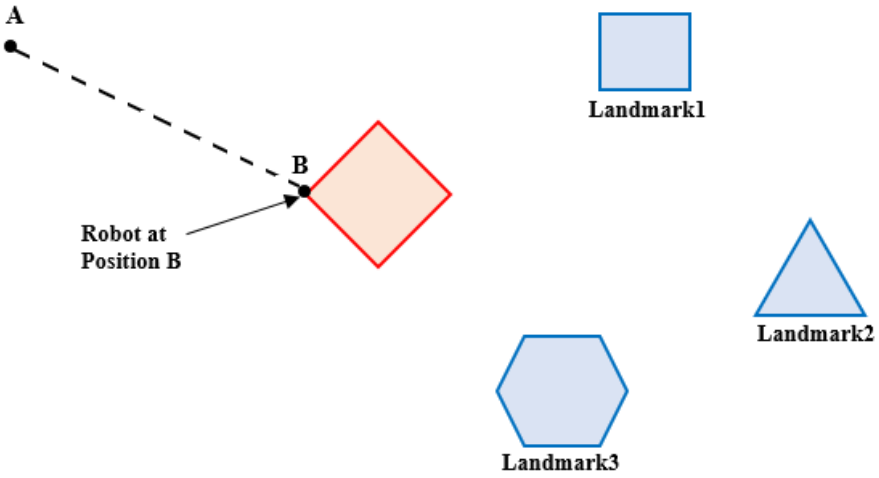


Figure 3-3. Robot moves. Positions moved by the robot

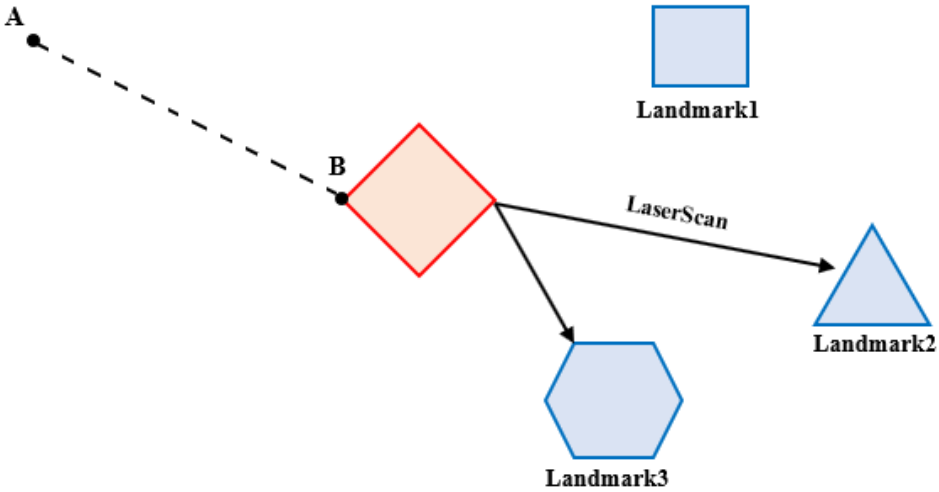


Figure 3-4. Scan & Data Association. Changes to the scan and odometry data are generated.

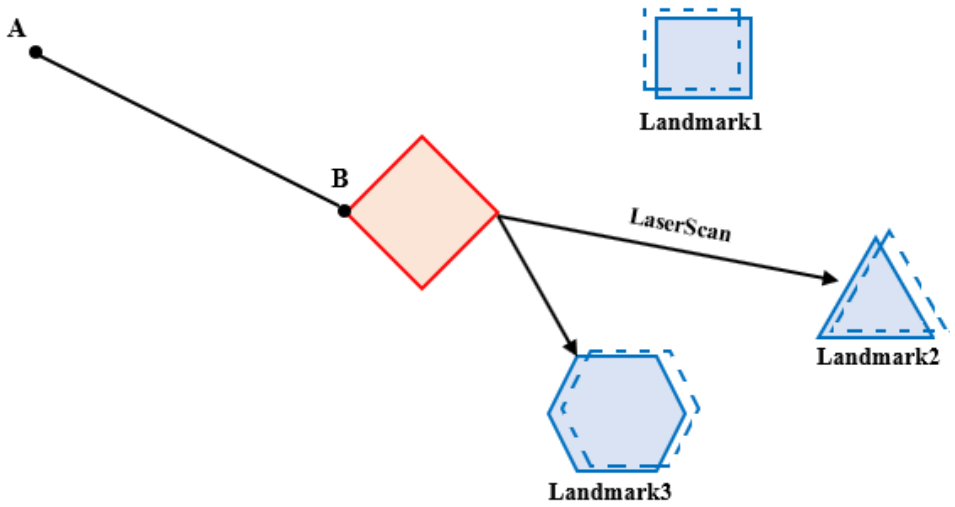


Figure 3-5. Update Step. Both previously collected data and new data obtained from the movement are considered by the robot. Actual landmark locations are indicated by dotted outlines.

SLAM has many moving parts, as shown in Figures 3-2 to 3-5. To find locations of objects, the robot must interpret data and also integrate movement information to determine where they are. As one can see in Figure 6, the dotted lines in the figure represent the actual location of the robot and landmarks in the environment. Between the actual and estimated locations of the robot, there are often discrepancies. The complexity of localization has caused this situation. However, odometry might indicate that the robot should be at a certain location. Exteroceptive laser scanners, however, might estimate the robot's pose differently. The localization estimate provided by combining the data is not perfect, but it is more accurate than before. Because there is a lack of perfect localization data, the robot has difficulties in providing accurate estimations of the positions of objects. The diagrams above show there are still issues with this solution, despite its well-vetted status. There are indeed some issues with the localization, mapping, and directions. Since its beginnings, SLAM has improved dramatically.

3.4. Current State of SLAM

Over 30 years have passed since Smith, Self, and Cheeseman [8] introduced the SLAM problem and proposed the first solution. Technology surrounding SLAM has advanced greatly throughout the years, from the algorithms and filters required to solve the problem to the implementation of mobile robots capable of solving it. An enhanced Kalman Filter is regarded as one of the most robust and proven SLAM techniques. SLAM has been developed and explored a number of times since, despite its vetting. Algorithms like the FastSLAM algorithm take advantage of particle filters, as do solutions based on graph data. The Rao-Blackwellized particle filter (RBPF) approach is different from the EKF approach in that RBPFs comprise an individual map of the environment and pose estimates based on individual particles [14]. The solution was first offered by Murphy, Doucet et al [5] [15] [16]. The authors demonstrate that RBPFs provide more accurate estimates than standard particle filters. Although the theoretical underpinnings are the

same, RBPFs achieve a higher degree of precision with marginalized variables [12][13][15]. With this marginalization, the inefficiencies associated with standard PFs are bypassed. Over one's state space, it involves the distribution of probabilities. It means that the algorithm samples over a subspace of the distribution to make sure the map is not included in the problem. Particle filters are used, and landmarks are mapped to each EKF [15] to predict localization. As a result, the sample space is reduced and the complex problem is more efficiently and accurately solved. Using the likelihood of each sample representing the posterior correctly, a weight is assigned to every instance in the space. New samples are selected from the areas with the highest weights when resampling is performed. Convergence towards the best solution results from selecting the strongest candidates.

Using FastSLAM, which estimates the full posterior distribution over robot pose and location, while scaling with the number of landmarks on a map recursively [12][17]. It can be used effectively in environments with a high number of landmarks owing to the benefits of this new method. It is necessary to develop FastSLAM due to the computational complexity of using an EKF and standard PFs [12][18]. The complexity of the algorithms limits the number of landmarks the algorithms are capable of recognizing. In [12], Montemerlo, Thrun, Koller, and Wegbreit present FastSLAM as a means of subdividing the original SLAM problem into subproblems. The FastSLAM algorithm handles the parameter estimation of landmarks in the map through the use of EKFs and the representation of the posterior of multiple robot paths by means of particles. This thesis focuses on G mapping, a strong solution to the SLAM problem, which is based on FastSLAM.

The final type of SLAM uses graphs to solve the problem, as the title suggests. There are nodes in the graph for each pose occupied by the robot during mapping. Every node has an edge that represents the constraints associated with it. Data about robot movements and the environment is used to calculate these constraints. Through graph-based SLAM, the most likely orientation of poses can be determined based on the data presented as edges [18]. In an array of applications, mobile robots are beginning to use SLAM algorithms to navigate autonomously using these algorithms. SLAM has been extensively researched, but there is still much work to be done. SLAM has also not been thoroughly studied from many angles and solutions. This presents a particular problem when operating robots that rely on SLAM in terrain that contains large undulations, such as outdoor environments. Further, there is a need to improve scalability and computing performance for SLAM applications. Also, there is little information on laser intensities used in SLAM processes or applications for humanoid robots [19].

Therefore, we have reached a true turning point. Due to the capabilities of modern technology, it has become possible to develop mobile autonomous robots for the general public. To deal with the SLAM problem one does not need to have extensive experience in linear algebra, EKFs, robotics, and computing systems. Numerous robotics platforms have been used to develop SLAM algorithms, many of which are heavily simulated. Most are open source, but some are proprietary. Several open-source algorithms are included in Robotic Operating System (ROS), so

anyone with an interest in learning about the technology can learn about and use them. Robotic systems that utilize SLAM can also be simulated with ROS without the requirement of any hardware. By doing so, expensive sensors and robots are no longer required. SLAM is easily implemented by simulating a mobile robot. The SLAM community is now accessible to anyone who has a computer powerful enough to run ROS. The change has enabled many more people to look at the problem, contributing to its growth and development, as you can imagine. A key aspect of SLAM for mobile robots is the ability of the system to execute autonomous navigation, which has been widely applied since its inception. As SLAM developed, indoor operations were the focus. Environments that are similar to the office but are unchanging or semi-dynamic. A large part of the solution was determined due to the issue of tracking indoors without using GPS [10]. Algorithms developed in the new environment were optimized to work in these contexts. The implementation changes were mainly intended to reduce computational complexity and improve general performance. This has resulted in SLAM being robust in these environments. Walls are an important feature of environments of this type. Most often, the robot will be viewing some kind of landmark or wall. For SLAM algorithms to function properly, exteroceptive data must be present at all times in order to ensure accurate localization and mapping. The pitfalls of the SLAM algorithm become apparent only when robots operate outside on unstructured terrain. If laser-based SLAM is used in the outdoors, it is subject to two key challenges.

LiDAR data is sometimes unavailable due to its nature of being used outdoors. The scanner will not return any useful data unless there are landmarks in its field of view. In contrast, indoor environments provide a constant wall presence, allowing the algorithm to work as expected. In addition, common LiDAR laser scanners acquire readings as 2D planes. When the terrain is undulating, the scanner is often pointing toward the ground as the roll and pitch of the scan change. SLAM algorithm is affected by the data generated while scanning the ground. The contribution section of this thesis provides more details about this topic and provides support for it through findings. Robot Operating System is, in its simplest form, a software development framework for robots. Robot behaviour can be built using ROS' tools, libraries, and conventions regardless of the platform. The keyword associated with ROS is collaborative. It was created to combine the contributions of developers of the robot software. Nodes written by others can be easily integrated into ROS systems by users. Researchers involved in robotics and software research can continue to push the boundaries of this technology by working together. Moreover, this framework boasts a huge community that contributes to its success. Several contributors continuously contribute to advancements on ROS Answers [33] through troubleshooting and educational posts. In this thesis, ROS is used to create a robust system. Evaluation of the ROS GMapping node will involve the use of a simulated robot designed for object detection.

3.5. Particle Filter-based SLAM

The most common object-tracking algorithm is the Particle Filter. The Monte-Carlo location with particle filters is a typical instance. The Kalman filter presented above ensures precision only in a linear and Gaussian noise system. Nonlinear systems are the biggest challenge

in the actual world. Based on the small set of data, the algorithm constructs a line fitting model. A comparison is then made between the model and the entire dataset to see which data fits. Outliers are considered data that does not meet a defined tolerance for error. Obstacles are identified by taking outliers, as shown in Figure 3-6.

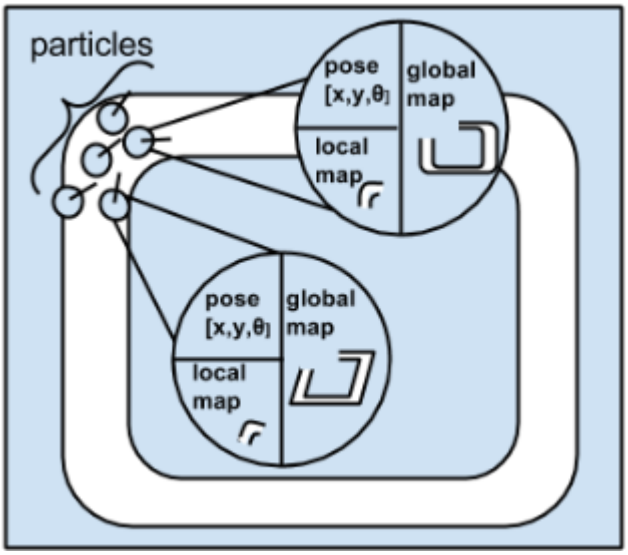


Figure 3-6. A collection of particles is a faith that the robot poses with a circle.

To complete the SLAM process, one must complete the map insertion phase. As the last step, the new landmarks are added to the current state of the map. The map is updated with any unused landmarks from the previous steps. Due to their unassociation, the algorithm determined that they were not seen yet. Therefore, their inclusion on the map is necessary. The important thing to note here is that the map becomes more complete as the map insertion phase is repeated. A degree of uncertainty is included in each landmark's position for initialization purposes. As a general rule, a map is composed of two parts. State estimates and their variances are these. It also shows whether the positions of landmarks are spread out or vary from the mean and a measure of how the estimates for each number differ from the mean. It is in the map insertion phase that SLAM algorithms complete a complete iteration. The steps of the algorithm are once again iterated as the robot moves and the data from the sensors changes. While simultaneously establishing a map of an unknown environment, the robot is simultaneously able to localize itself within it.

Like other pose estimation techniques, the particle filter predicts the position of the object if the mistake is included in the incoming input. The use of SLAM to determine the present position of the robot is the odometric value and the measurement data using the distance sensor. A collection of particles named samples describes the unknown posture in the particle filter technique. In accordance with the robot's motion model and probability, we transfer particles into a new estimated location and orientation and progressively measure the weight of each particle by its actual measurement value. In the case of a mobile robot, each particle is shown as a particle = weight (x, y, i) and each of these particles is an arbitrary tiny particle, which is the predicted

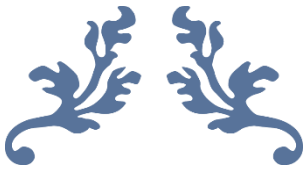
position and direction of the x, y, and i robot robots, as well as their weights. The following 5 processes apply to this particle filter. With the exception of initialization in step 1, steps 2~5, the robot posture is estimated periodically. This is a way of assessing the robot's position by updating the particle distribution, which demonstrates that the robot is likely to be on the X, Y, coordinate plane based on the value of the measurement sensor.

- **Initialization:** The first location (position, orientation) of the robot is unknown so that the particles are placed randomly within the scope of N particles. The weight of each original particle is 1/N and the particle weight total is 1. N, generally in centuries, is calculated experimentally. The particles are located next to the robot if the original position is known;
- **Prediction:** Based on the system model defining the robot movement, each particle moves with odometry and noise information as the amount of observable motion;
- **Update:** The probability is computed and the weight value of each particle updated based on the calculated probability, based on the observed sensor information;
- **Pose estimation:** For determining the robot's average weight, median value, weight, and orientation, the position, orientation and weight of all participants is calculated;
- **Resampling:** The process of producing new particles is to eliminate fewer weighted particles and generate new particles to inherit the weighted particle information. The N particle number has to be retained here.

3.6. FastSLAM and its Derivatives

The thesis examines the effectiveness of the G mapping algorithm in localizing and mapping simultaneously in an extraterrestrial robotic object application. Thus, understanding the underlying theories and structures of the algorithm is crucial. RBPF-based FastSLAM algorithm was used to develop G mapping. SLAM is a complex problem, and EKF solutions are robust solutions, however, they can only be used with Gaussian systems [49]. PF-based approaches have been introduced as a way to add to the capabilities of SLAM. PFs are constructed using a mechanism in which each particle contains a map of the surrounding environment and a robot pose [49-50]. The particles are updated every time the robot moves and the map is updated according to the new pose. Furthermore, each particle in the system has its own representation of the map and robot orientation, as well as its own weight. It makes use of this weight to filter out weaker samples and converge to the strongest representation [14][49]. Consider this as a sort of natural selection process. It is impossible to have an infinite number of particles with a map and a pose if a set number of particles is kept. Resampling is possible when the less likely particles are removed, and the strength of the estimation is further confirmed. There is an extremely high computational complexity due to the significant number of particles, each of which contains a map and a pose estimation. Rao-Blackwellized particle filters were introduced to reduce the computational complexity of this problem. RBPFs was first proposed by Doucet in his work [49-50]. Murphy and Doucet argue the Rao-Blackwellization particle filter uses the same basic premise as those previously described, however, it takes advantage of the marginalization of state variables. In this

marginalization, the overall sample space is reduced, thus optimizing computational complexity [49]. This approach, however, still needs a lot of work.



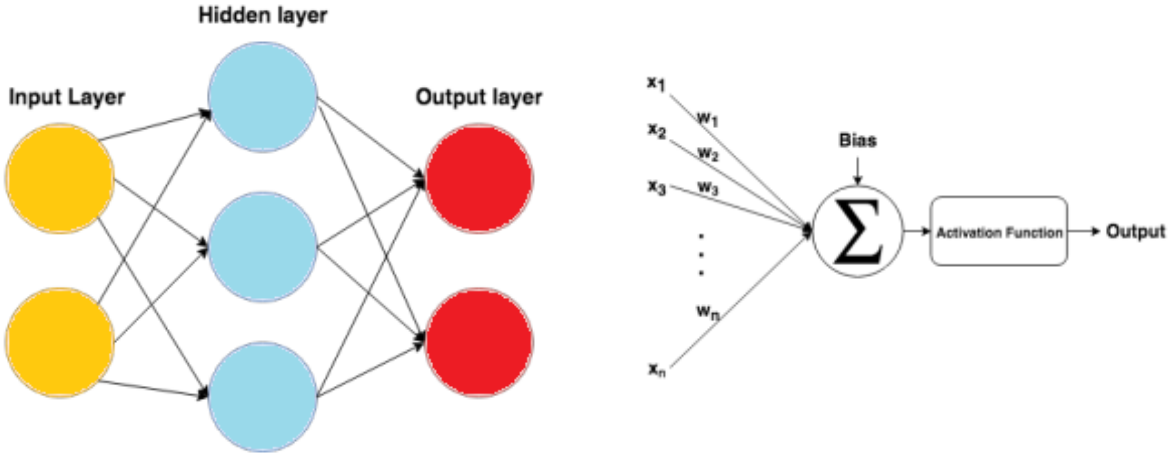
CHAPTER 4

ARTIFICIAL NEURAL NETWORK FOR NAVIGATION

4.1. Artificial Neural Networks

Neural networks have been used successfully for predicting objects distance. The purpose of this section is to provide an overview of the basic concepts of neural networks and to explain how techniques for neural networks have improved the viability of such networks. This section covers several activation functions, optimizers and how to avoid common mistakes with a network during its training phase [32]. Machine learning techniques like ANNs are designed to mimic the brain in a loose way; their fundamental idea is based on the Hebbian theory, which attempts to explain how neurons adapt to learning [16]. An artificial neuron is the main building block of an ANN. In 1943, Walter Pitts and Warren McCulloch introduced the first model of an artificial neuron [31]. During the training phase of an ANN, the goal is to perform analytical tasks based on feedback from the performance of previous tasks. It is essential to consider the learning technique when providing feedback. This project can be classified as supervised learning, which means that the network can be fed feedback based on the relevant training data, or ground truth. There are many layers as shown in Figure 4-1a of an ANN that is composed of a series of computer nodes called neurons. Depending on the type, these layers may be:

- Input layer: Samples fed into the network are processed first by the input layer, which must be capable of representing samples in an appropriate form: In this layer, all that is done is send the data to the subsequent layer.
- Input neurons' outputs are then passed sequentially to hidden layers (intermediary neurons layers between input and output layers). To perform more complex nonlinear work, these layers are essential. In this part, neurons represent abstract features of the input and help to generate the correct output. (A neural network with at least one hidden layer can be called a deep neural network (DNN). Thus, deep learning is called deep neural networks.)
- Output layer: The last layer is the output layer. Depending on the task, the output should be in the form necessary to indicate the required data. (a) Layers in a NN (b) Neuron in a NN.



(a) (b)

Figure 4-1. A neural network is divided into three layers a) layers in a NN and b) neuron in a NN

A neuron in an ANN is mathematically represented as the weighted sum of its input plus a bias, followed by a decision function that determines which information is to be passed forward. Being activated is when a neuron passes information forward. The following mathematical formulae are used to calculate a neuron's output:

$$O_i = g\left(\sum_{j-1}^{N} \omega_{ij}\, x_i - b_i\right) \tag{4-1}$$

x is the input, ω is the learnable weights, and b is the bias. N is the number of input values, and g() is the activation function. Basically, every node has a weight wij concerning every input. This represents how much attention should be paid to each input. By enabling shifting, the bias parameter improves the representation of the desired function. Neurons' output is controlled by the activation function. A forward propagation algorithm calculates an output from the input using a new input. Backpropagation is the actual learning process; it updates the parameters, weights, and bias. The following section describes this technique.

4.1.1. Learning

Weights and bias can be learned by an artificial neuron, as mentioned previously. Initializing these parameters with random values and updating them progressively with backpropagation are common practices in the training phase [17]. It is necessary to find the value of trainable parameters that will result in the smallest loss based on the so-called loss function. The loss function is a metric of how far off the neural networks output was from the desired output. The Mean Squared Error (MSE) is a simple loss function presented in this manner:

$$MSE(O, \hat{O}) = \frac{1}{N}\sum_{i=1}^{N}(O - \hat{O})^2 \tag{4-2}$$

N represents the number of outputs, $\hat{O}$ represents the number of desired outputs, and O represents the output of the network. A global minimum of the function is the goal of achieving the best weights and biases for all neurons with the lowest loss. Optimizers based on gradient descent update the learnable parameters by adjusting the parameters according to:

$$\theta = \theta - \eta \nabla J(\theta) \tag{4-3}$$

$\nabla J(\theta)$ is the gradient of the loss function $J(\theta)$, where θ is the parameter to be updated, and η is the learning rate. During backpropagation, by taking the partial derivative of the loss function, the gradient of the backpropagation algorithm is calculated concerning the parameter that should be updated. Partially derivatives for a specific weight depend on all subsequent layers, so the backpropagation has to start from the last layer, and it has to go backwards. Chain rules are used to determine how a layer's weight depends on a layer below it. According to the concept, more updates should be made to the neurons that have the greatest influence on the output.

A network's final output is based on a generalization of what it has experienced, i.e., on what it has trained on. During network training, it is common to find that the model learned much detail from the data. On the training set, the network performed well; however, on the new data, the network performed worse. The problem is called overfitting. During the training stage, the available data is typically split into two sets: A training set consisting of the majority of the data where based on the loss, the network updates its learning parameters. Additionally, a validation set is used to test if the model generalizes.

4.1.2. Optimizers

Each of the available optimizers has advantages and disadvantages for updating the trainable parameters. Following a network's complete processing of the whole dataset, the gradient descent algorithm can perform one update.

4.2. Theory

It means, the update takes into account the average of the entire dataset, so it is common to converge with a local minimum that is not optimal. For solving the aforementioned problem, gradient descent optimizers can be used:

- For each training sample, stochastic gradient descent is performed once. Because of the frequent updates, the trainable parameters have a high variance because of the different input samples. With the high variance, a more efficient local minimum could be found than with the standard gradient descent. Furthermore, oscillations could lead to overestimation of the network's updating value and complicate the convergence process.
- Once a set number of samples has been processed (called a batch), the parameters are updated. Batches should be smaller than the full training set. This method involves a batch of training samples is processed, and then an average loss of the batch is taken as the basis for updating the network. Even so, gradient descent and its variations present some challenges, such as choosing the right learning rate and avoiding sub-optimal local minima. Listed following are some additional techniques that enhance gradient descent, thereby mitigating the above challenges.
- By adding a fraction of the update vector from the previous step update V (t − 1) to the current update V (t), a technique known as Momentum softens the oscillations of updates. Using the following mathematical formula:

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta),\ \theta = \theta - V(t) \quad (4\text{-}4)$$

In which, θ is the learnable variable, η indicates the learning degree and γ denotes the momentum degree, defining the portion of the momentum from the preceding phase that must be added in the update [43].

- This can be expressed as a three-part definition, with θ being the learnable parameter, η being the learning rate, and γ being the momentum rate, which is defined as the fraction of momentum from the previous step that needs to be added in the update [43].

- A specific parameter's learning rate is adjusted based on the frequency with which that parameter is updated by Adagrad. A parameter that is rarely updated maintains a higher learning rate than one that is frequently updated. With this kind of updating, it is less likely that the learning rate would need to be manually calibrated. Using Adagrad also has a disadvantage, that it calculates the learning rate through the accumulation of the previous gradients, which forces the learning rate to always decrease, resulting in that the model would stop learning.
- An extension of Adagrad named Adadelta, which provides a solution to the decaying learning rate issue. This is achieved by only considering a fixed number of previous gradients in defining the adaptive learning rate. Similar to how momentum parameters are updated, the learning rate is also updated in the same way: a fraction of the previous gradient means is added to it.
- Also, Adadelta, a method that computes adaptive rates of learning for each parameter, is called Adaptive Moment Estimation (Adam). Nevertheless, every parameter is also associated with a specific individual momentum. Due to this, Adam can calculate both the momentum and the learning rate for each parameter and avoids most of the issues.

4.3. Activation Functions

The activation functions can be used to introduce non-linearities into the neural networks, thus enabling complex nonlinear problems to be solved. Activation functions come in several different forms with different properties. Based on the tasks the network is intended to accomplish, all are appropriate. The pros and cons of activation functions are discussed in this section. For classification problems, the Sigmoid function is widely used [32]. It restricts the neuron's output to a range between 0 and 1. As follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4-5}$$

Backpropagation is a process in which based on the gradient in the previous layer, the weights are updated backwards depending on the backpropagation. The gradients are not steep, as shown in Figure 4-2, along the edges of the sigmoid function. As a result, nodes in the last layers could have small gradients, which would lead to minimal updates for the first layers. Consequently, the learning process for the network is prolonged or completely stopped. A vanishing gradient [19] is known as an issue that historically prevented deeper networks from convergent within a reasonable period.

4.3.1. Hyperbolic Tangent (tanh)

Tanh, or hyperbolic tangent activation function, is a scaled version of the sigmoid function, thus resulting in a steeper gradient. Figure 4-2 shows the Hyperbolic Tangent. it is defined as the best formula where, a range from −1 to 1, allowing the values to be zero-centred. However, the gradient still vanishes for such activation functions. This is a widely used activation function for deeper networks called the Rectified Linear Unit (ReLU) [24]. The figure shows that for negative numbers, it outputs zero and for positive numbers, it outputs the input.

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{4-6}$$

4.3.2. ReLU

ReLU is a non-linear activation function. This is because gradients of zero or one diminish the problem of vanishing gradients. It is possible to ignore neurons that have zero output, reducing the number of computations necessary, thus improving the efficiency of the network.

$$f(x) = max\ (0, x) \tag{4-7}$$

The backpropagation update, however, would be zero if the neuron has negative weight and bias, implying that the activation function would be at a zero gradient. For the neuron to start emitting anything other than zero again, it will need to be adjusted by some external force, a phenomenon called the dying ReLU problem. There is also the issue that the amount of output from ReLU is not limited, so activations could blow up (become disproportionally large) [32]. Activation functions, such as the one mentioned earlier, have a simpler mathematical formula, which may contribute to their increase in computation speeds. ReLU is a method of solving the dying linear unit problem called the leaky rectified linear unit (LReLU). This technique changes the gradient for negative numbers to make them smaller as:

$$f(x) = ma\,x(0, x) - \alpha ma\,x(0, -x) \tag{4-8}$$

α is set to a small value near zero in the function given by the next formula. When solving classification problems with more than one class, softmax is often activated in the output layer. As a result, logit (the raw output scores from the last layer) is converted into probabilities that add up to one. This formula has the following form:

$$S(y_i) = \frac{e^{y_i}}{\sum_j^c e^{y_j}} \tag{4-9}$$

y_i represents the output probability of a particular class i in the output layer, and c represents the classes number.

4.3.3. Regularization

It is possible to avoid overfitting and achieve a standardized output with the concept of regularization. Regularization forces the network to generalize better by making slight modifications to it as it learns. There is a regularization technique named dropout. Consequently,

each neuron is not able to update its parameter and produce an output with probability P. Regularization also can be performed using L_2 regularization, which forces the weights to decay towards zero by adding a term to the loss function. The technique is described as follows:

$$L_{new} = L_{perv} + \frac{\lambda}{2m}\sum \|\omega\|^2 \quad (4\text{-}10)$$

Here the term on the right of the plus sign indicates the L_2 regulator, L_{new} as the total loss, and also L_{prev} as an arbitrary loss function. The regularization parameter represents by λ, where m is the number of outputs and w is the learnable weights. Therefore, the gradient would be affected by the addition of regularization when the loss function is different for each weight.

4.4. Formulation and algorithm of used Convolutional Neural Networks

In a CNN, the architecture is intended to leverage the structure of the input (typically images) to improve the response. Mainly, a network is normally constructed of two building blocks, the convolutional layer, and the pooling layer. These two layers are discussed in more detail below. By weighting and modifying identifiable characteristics of smaller patches inside each layer, these building blocks are used to represent the data as features. By using building blocks in this way, the level of abstraction of the input becomes gradually greater as it progresses through the layers. Later layers contain more complex information about how specific objects actually appear, whereas earlier layers contain simpler structural information.

4.4.1. Convolutional Layer

As with any ANN, the convolutional layer is made up of neurons that can be trained. Convolutional layers work by learning weights and biases for each neuron in the layer and calculating its output based on those values. Alternatively, neurons can be visualized as matrices called filters where the weights can be shared between them. By virtue of convolutional filtering, the input is applied to these filters. Convolutional operation is applied to all input, both rows and columns, by sliding the filter across them. Each location is multiplied element-by-element and summed together to produce the result that is included in the output feature map. Each convolutional layer has four significant parameters:

- As a first step, specify the number of weights within each filter by specifying the size of the filter. As they can handle smaller and local features of objects, filters in the range of 3 × 3 to 5 × 5 are usually used. In addition, the depth of each filter is adjusted according to the channels present within the input;
- The second step pertains to the number of filters, indicating how many will be applied to the data on each layer. Based on the number of filters present in a layer, the final output feature map contains a set of channels. Figure 4-2 illustrates how five filters can be applied to a simple image. An example of output depth when using multiple filters is shown in Figure 4-2. This results in a five-depth feature map since five filters have been applied to a three-channelled image.

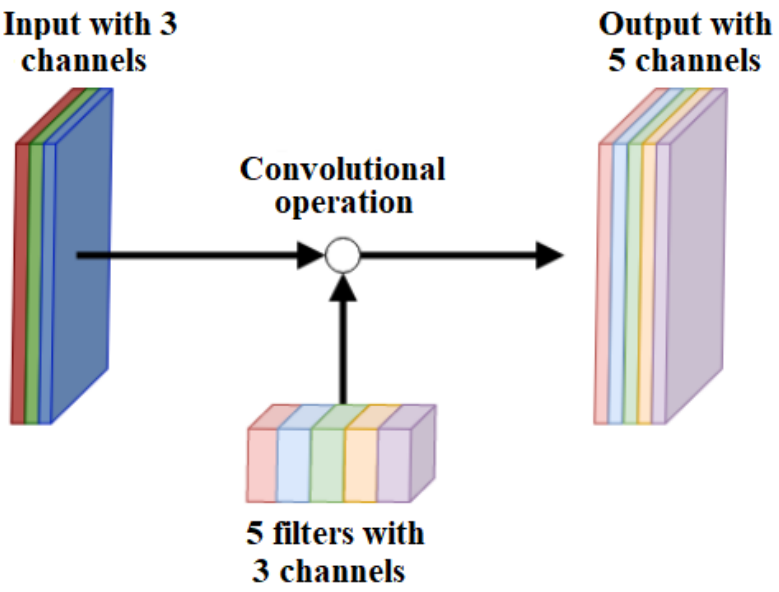


Figure 4-2. Example of the output depth by multiple filters

- Third, As the filter slides through the input data, the stride indicates how much it shifts at each step. Figures 4-3 shows examples using a stride of one. Increased stride results in a significantly smaller output feature map.
- Finally, padding specifies whether and how many padding borders should be added

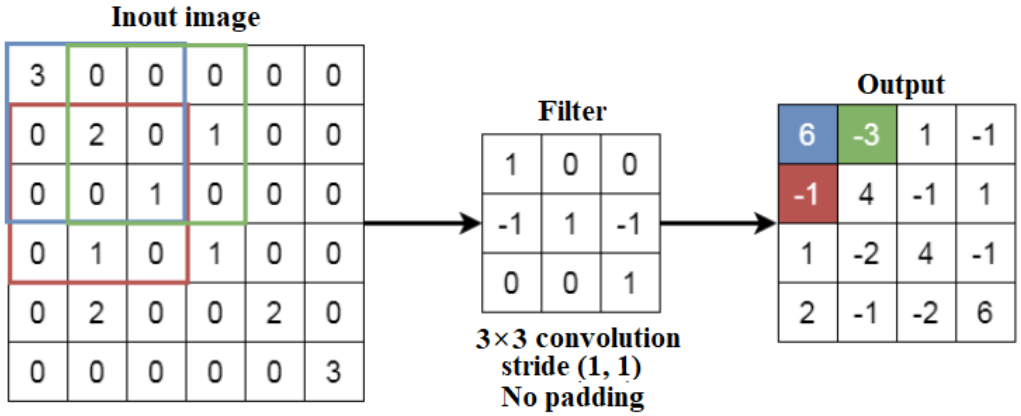


Figure 4-3. Here's a convolution operation example applied to a single filter.

An output feature map of each layer has the following dimensions:

$$O = \frac{N - F + 2P}{S} + 1 \qquad (4\text{-}11)$$

N is the input map size, O is the output dimension of a row or column, F represents the filter size, P stands for padding row/column size, and also S is for stride. The resulting values are rounded towards zero if they are fractions.

4.4.2. Pooling Layer

The output feature map is downscaled using pooling. As a result, fewer parameters and computations will be used in the network, and the amount of overfitting will be reduced altogether (by only keeping the most promising features). The most common type of pooling is max pooling. According to Figure 4-5, only the maximum value in each region is kept as output. This is done by splitting the input feature map into equal-sized regions. Pooling affects output size as expressed without padding depending on the size of the regions and the stride. Figure 4-4 shows an example of using max-pooling on a 4*4 input feature map with a size of 2 * 2 and a stride of two. An output is generated for every section of a 2 x 2 feature map in which the highest value appears.

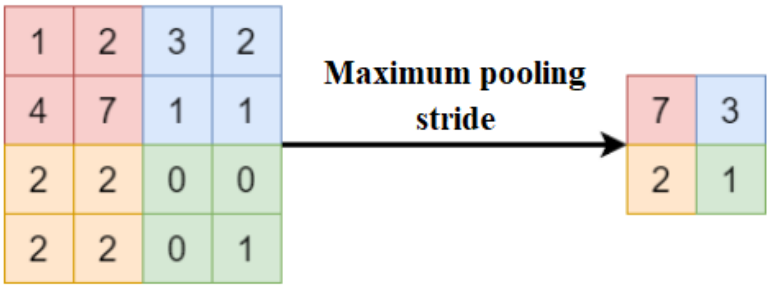


Figure 4-4. Example of the claim of max-pooling.

4.4.3. Transposed Convolutional Layer

The opposite of the convolutional layer reduction in the size of the output feature map can be achieved by transposing convolutional layers. Convolutional operations can be reversed to perform upsampling of feature maps. The weights could be learned as well as visualized as filters in the same way as convolutional layers. The filter is multiplied by the input value for each input location, and the result is placed on the output map at the corresponding location where overlaying values are summed. There, No padding was added to the stride and the filter was applied. The value at each input location is multiplied by the filter's entire parameters and positioned at the same location on the output map. The output feature map is a 6×6 representation of the overlapping values.

4.4.4. Unpooling

A transposed convolutional layer reverses the state of convolutional layers, whereas an unpooling layer reverses the pooling layers. To reverse the operation of max-pooling, there is max unpooling. In the process, it records the positions of the maximum activations during the max-pooling [34], where the maximum values during downsampling were found. Afterwards, the unpooling reverses the maximum operation while filling the remaining positions with zeroes. In Figure 4-5, we can see the same result from max unpooling as shown in Figure 4-4.

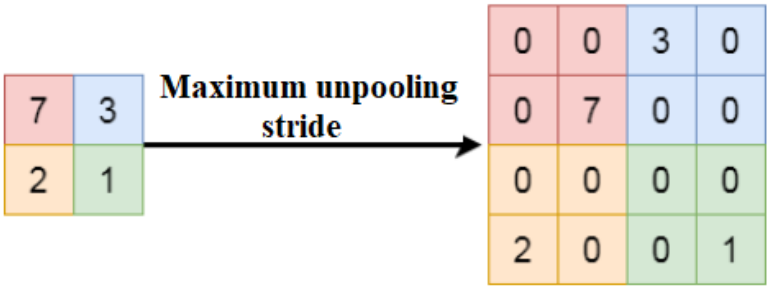


Figure 4-5. This example illustrates how max-pooling can be applied to a feature map and a stride.

4.4.5. Fully Connected Layers

Convolutional layers and pooling layers learn features, and fully connected layers then increase the ability to reason with these features at a high level. As they are capable of moving from grid representations to single values, fully connected layers heavily depend on the type of output intended. By using convolutional networks to represent data, we create feed-forward networks. It may prove useful in cases of classification or regression using input data as a whole. Data translation occurs through a flatten layer that creates a vector from each matrix feature value by translating each position into an input value that is then passed on to each fully connected layer following. Figure 4-6 illustrates how a feature map of size 2 × 2 × 2 can be converted into a layer containing four neurons through the use of a connection layer.

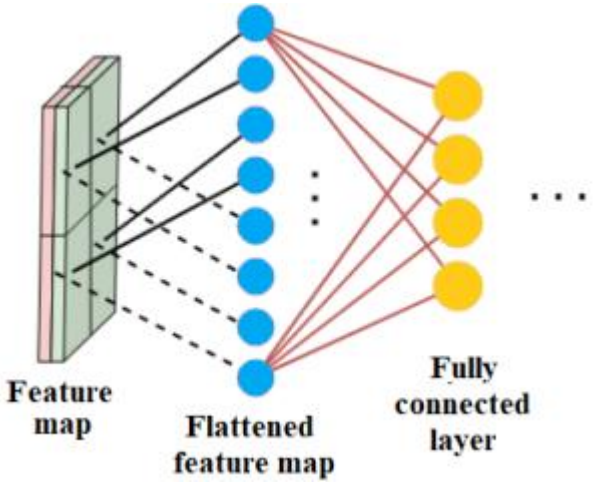


Figure 4-6. A feature map with a layer of four neurons.

4.5. Transfer Learning

Traditionally, deep learning algorithms require a large amount of data to perform their tasks effectively. Despite this understanding, there is a workaround known as transfer learning when neural networks are large. It becomes possible to transfer weights that have already been learned on a model created from the same architecture by using some pre-defined network architecture. By using transfer learning, the features extracted from training on larger datasets can be used to generate feature representations instead of creating them from scratch. By utilizing this method

correctly, it not only increases performance but also reduces your training time. However, to be profitable, transfer learning depends primarily on one big assumption, which is that patterns found in the original dataset are applicable in the new data. There is essentially no difference in the type of data. It can be difficult to obtain information from another network that is trained on images of cars when trying to determine whether a tumour is benign or malignant using scanned images.

CHAPTER 5

DEVELOPMENT OF LIDAR NAVIGATION SOLUTION: THE CASE STUDY

5.1. Introduction

In this chapter, the performed coding and algorithms used in the current thesis are presented. For this aim, this chapter is categorized into three main parts: Object detection, Mapping and Artificial neural network, as also shown in Figure 5-1.

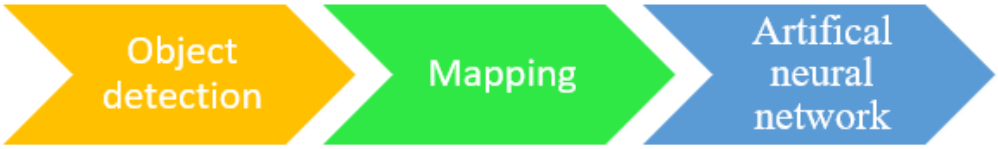


Figure 5-1. performed steps in this chapter

5.2. Object detection algorithm and formulation

In this section, based on those mentioned in the previous section to find an object using LiDAR and ROS and their connection, the formulation and coding performed in this thesis to find objects locations and coordinates are discussed. Due to how the encoded point cloud is structured, it can be interpreted as an information-rich dense image with a depth equal to the number of features output from the point feature extractor for each pillar. This makes it possible to effectively process the entire feature map by 2D object detection methods. The decoder used in the original PP implementation was a modified SSD. The modification added regressional targets in the shape of height, z position and direction, along with an additional binary classification for the direction. The purpose of the directional classification is to be able to predict the heading of an object concerning the matched anchor, thus being able to more effectively use the anchor boxes original orientation as a point of offsets.

In this thesis, the algorithm accomplishes its function by a pipeline of 4 processes when each new scan is received as below and Figure 5-2:

1. **Scan conversion:** Object-oriented scan data representation
2. **Clustering:** divide scan into different clusters
3. **Identify the closest cluster:** Define the closest cluster-object from the rover
4. **Control rover speed:** Modify the speed and heading of the rover depending on the environmental perception

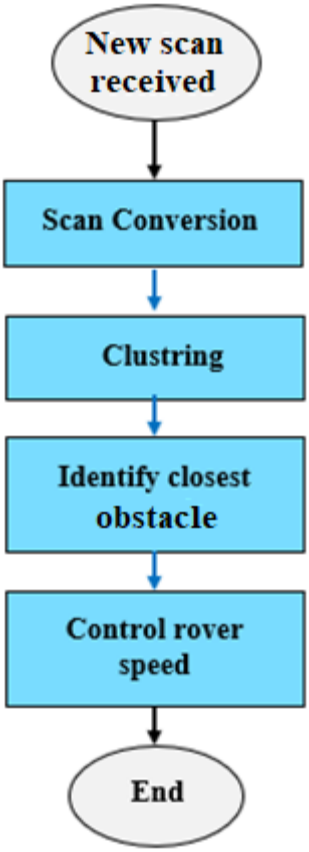


Figure 5-2. Accomplished algorithm used to identify objects

5.2.1. Laser scan processing

The source of laser scan is the lidar sensor, a typical lidar sensor that emits pulsed light waves into the surrounding environment. These pulses bounce off surrounding objects and return to the sensor. The sensor uses the time it took for each pulse to return to the sensor to calculate the distance it travelled. Each laser scan has a predefined angular position, thus by arranging the totality of range scans, we obtain a 2D space representation from the robot perspective. The lidar sensor is widely used in obstacle detection since it gives both the distance and angular position of a lying obstacle. In ROS the laser scan message is defined by sensor_msgs/LaserScan type, each laser scan is a single scan line at a given moment. The message information is presented in **Appendix I**. Figure 5-3 illustrates the laser scan message components:

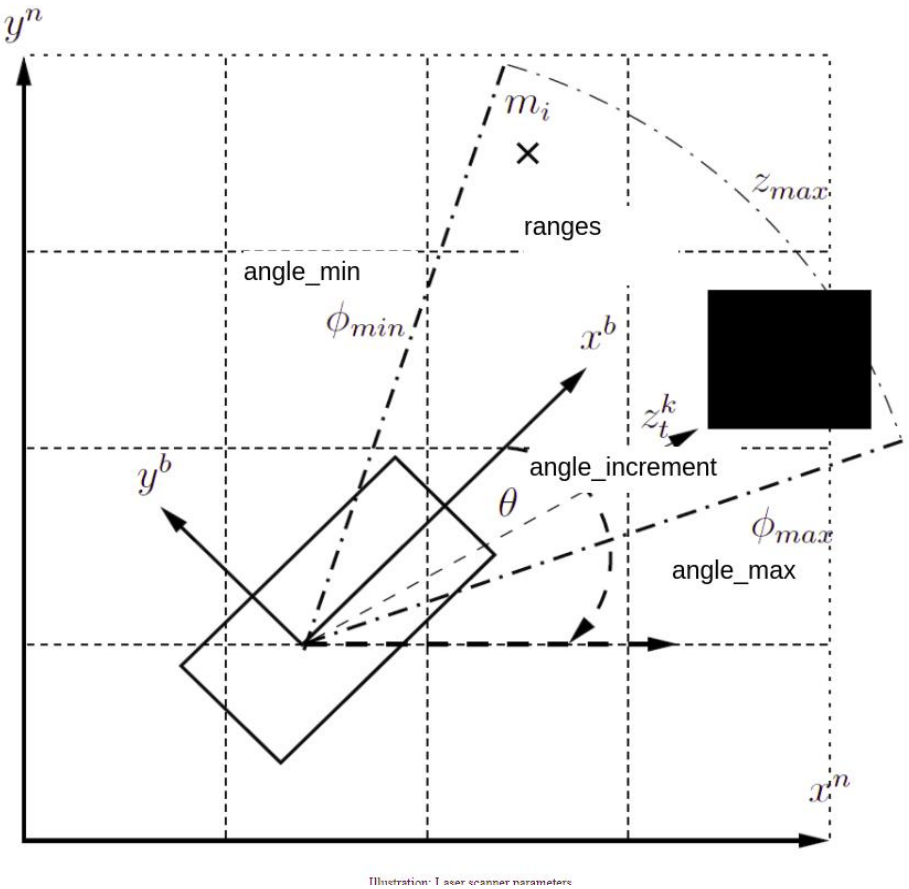


Figure 5-3. Laser scan message components diagram

In our case, the laser scan message is obtained in cmd by running a rostopic echo /scan. The performed laser scan message coding is provided in **Appendix II.**

In our application the lidar sensor is positioned on top of the rover, to make the obstacle detection trivial we align the sensor in a way that the transform between its frame and the base_link frame of the rover are aligned, thus the x-axis of the laser aligns with the frontal direction of the rover which gives an angle of 0 in the vertical direction of the rover, in rospy, to use the laser data, the ROS node subscribes to the /scan topic and assigns a function that serves as callback when each new laser scan message is received. In the callback the laser scan is processed in the scan_processing method, to perform the following tasks:

5.2.2. Scan data conversion

To simplify the message representation we convert the scan array (view message definition), to object representation: Each range is converted to a measurement object. The measurement object consists of the two fields:

- Range:
- Angle :

The conversion method consists of iterating over the array structure, the angle is obtained by adding incrementing the indexes of the scans to angle_min parameter as:

```
angle = angle_min + idx * angle_increment
range = ranges[idx]
```

Where idx is the scan index in the array. The values are passed to the measurement object constructor and add it as a new item in the measurement list. In the current thesis, the data was sorted from objects that existed on the campus of Politecnico di Turino

5.2.3. Clustering

The measurement list is passed to the closest_cluster method to perform laser scan clustering to define the closest obstacle. Clustering is an approach to divide the scan data points into independent groups of points which defines a full object or a portion of a body. Prior to the clustering phase, We discard the ranges that are further than a prefixed distance. This allows us to avoid unnecessary readings beyond our area of interest as well as enhancing the chances for more separate clusters. The separation criteria are explained in Figure 5-4. Additionally, the total considered clustering is presented in Figure 5-5.

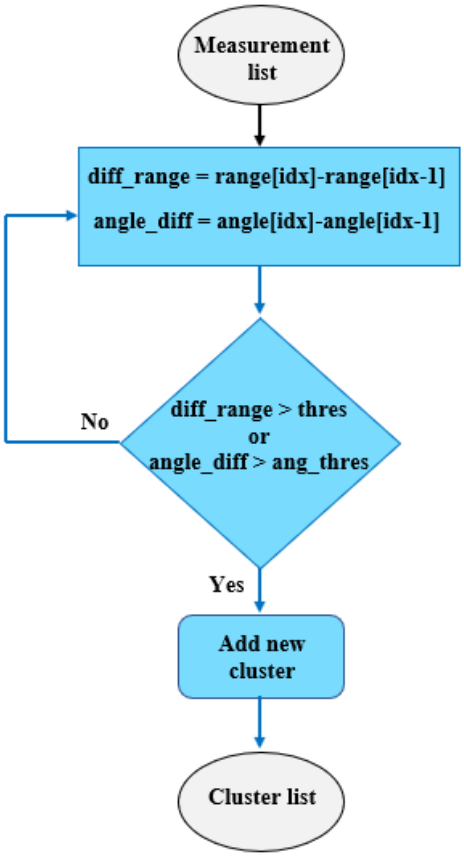


Figure 5-4. Separation criteria considered in the clustering stage

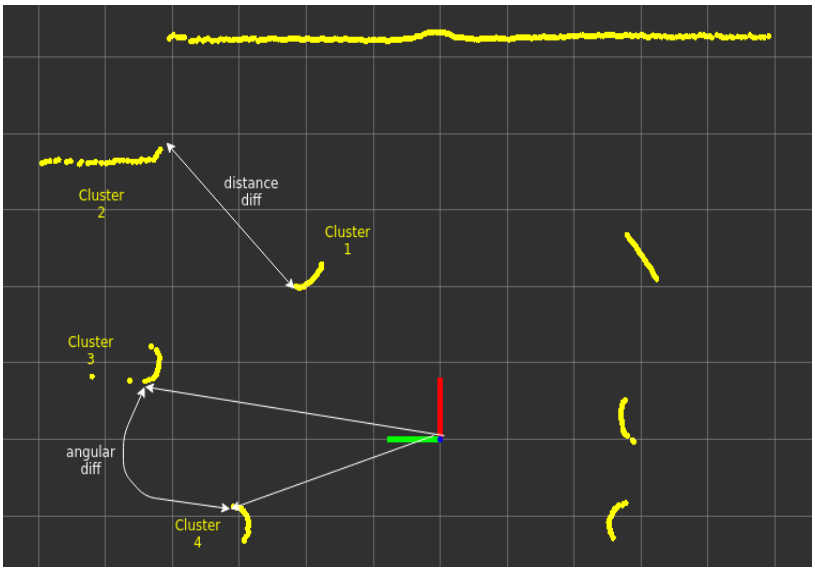


Figure 5-5. Number and distribution of clustering

According to Figure 5-5, a total of 8 clusters were found as below:

- ✓ Cluster 2 is identified by detecting range difference: The range difference between the first point from cluster 2 and the last point of cluster 1 exceeds the range_diff threshold.
- ✓ Cluster 4 is identified by detecting angular difference: The angular difference between the first point of cluster 4 and the last point of cluster 2 exceeds the ang_diff threshold.
- ✓ In the closest_cluster function which performs in the first part clustering, we fixed the range_d_thres to 0.2 m and the angle_d_thres to 0.17 rad.

5.2.4. Closest obstacle identification

From the obtained list of clusters that form candidates of the closest obstacle, we form another list containing the closeness criterion of each cluster. The closeness criterion is based on the horizontal distance noted dy, that separates the object and robot centres, such that

$$d_y = |\, y_o - y_r \,| \tag{5-1}$$

with y_o , the obstacle y coordinate in the robot base_frame and y_r, the robot y coordinates in base_frame which is equal to zero. To calculate the y coordinate of the obstacle, We choose the centre of the cluster representing the object as the medium-range point, given that scan, points are taken in the base_link frame, the y coordinate of the medium point is the projection of range on the y axis of base_link frame and this it is calculated as follow:

$$y_o = R \times sin\theta \tag{5-2}$$

Where R is the range of medium point and θits angle. From Equations 5-1 and 5-2, we define the closest cluster by minimizing the function noted below,

$$\widehat{d_y} = min|\, R \times sin\theta \,|^2 \quad (5\text{-}3)$$

The cluster with the minimum value is selected as the closest obstacle. Another test is performed on the cluster to check the possibility of passing beside the obstacle without collision. We fix a value of $1.75 \times W$, where w is the rover width as a security distance. If the selected cluster is beyond the security distance, the closest_cluster function returns the cluster data, else the function returns an empty list. The function output is passed to the velocity control bloc discussed in the next paragraph.

5.2.5. Velocity control

In the laser callback function, when each new scan data arrives it is processed to identify the closest cluster, and depending on whether there is an obstacle or not the velocity commands are set as the diagram describes (Figure 5-6):

In the callback function, the velocity control takes initially forward speed value as it is the default navigation mode for the rover, if the closest_cluster function returns a cluster, its centre angle is compared to the window_in parameter which represents the navigation direction of the robot in our case we set the window_in parameter to 0 rad as it is aligned with the x-axis of base_link frame. If the angle of the cluster centre is inferior to window_in, the obstacle is on the right side which initiates stopping the rover and turning it left, likewise in the second case.

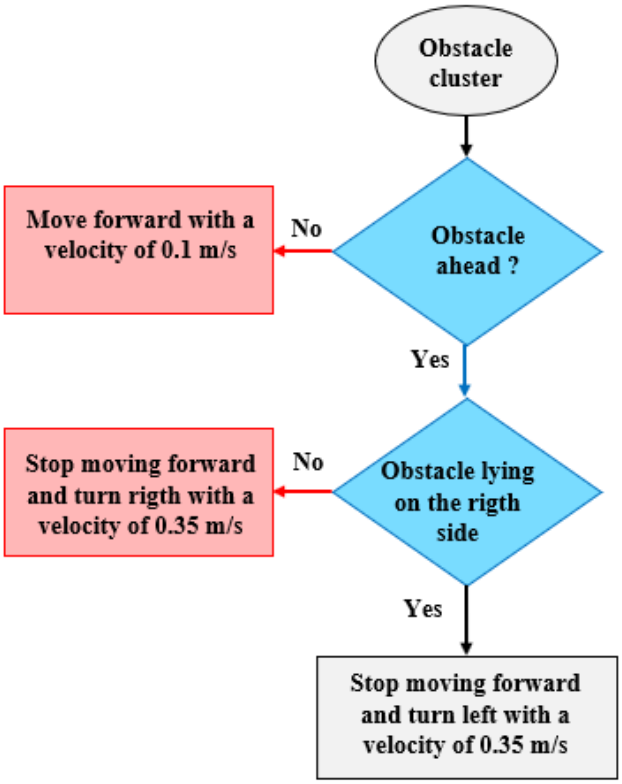


Figure 5-6. Laser callback function flowchart

5.3. Gmapping

OpenSLAM Gmapping is the most often used LiDAR SLAM-based method, following efficient Rao-Blackwellized Particle Filter (RBPF) implementation. The sampling history of each particle is a sample of the history and the background of robot postures given the tale of the sample, that is, each particle is shown with a map of the environment. The main goal is to minimize the number of particles used in an adaptive method. Recent observations are utilized to correct the map, rather than depending only on the movement of the robot (the history of the robot positions). Previously compared two methods, i.e., online EM and bayesian inference. The map was regarded as a fixed parameter at the online EM, whereas the map was viewed as a matrix of random variables in the Bayesian inference. The online EM algorithm has been demonstrated to readily stick to a local minimum, which makes the robot seen to be lost. Recent observations are utilized to correct the map, rather than depending only on the movement of the robot (the history of the robot positions). The map was regarded as a fixed parameter at the online EM, whereas the map was viewed as a matrix of random variables in the Bayesian inference. The online EM algorithm has been demonstrated to readily stick to a local minimum, which makes the robot seen to be lost.

1. The first guess of the posture of the robot represented by component i is $x_t'^{(i)} = x_{t-1}^{(i)} + u_{t-1}$. The above position $x_{t-1}^{(i)}$ and odometry measurements u_{t-1} obtained from the latest update of the filter is computed for this. The operator here is the usual operator for compounding;
2. A scan matching procedure is conducted based on the $m_{t1}'^{(i)}$map starting from the beginning $x_t'^{(i)}$. The search is restricted to $x_t'^{(i)}$ approximately. In the event of an error, poses and weights using the motion model are computed and steps 3 and 4 are omitted;
3. A set of sample points will be picked in an interval around $\hat{x}_i{}^{(i)}$ provided by the scan matcher. Based on this measure, by assessing the goal $p(z_t m_{t1}^{(i)}, x_j) p(x_i x_{t1}^{(i)}, u_{t1})$ at the sampling position x_j, the mean and covariance matrix of the suggestion will be determined. The weighting factor $\eta^{(i)}$ is computed at this step;
4. Calculate the gaussian approximation $N(\mu_t^{(i)}, \sum_t^{(i)}$), and the new $x_t^{(i)}$. the posture of the i particle will be drawn 4;
5. Weights of significance are revised. The map $m(i)$ of the particle I shall be updated according to the posture $x_t^{(i)}$ and z_t observation.

In the current thesis, based on the steps discussed above, we used the gmapping ROS package which is a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides a ROS node called slam_gmapping which performs laser-based SLAM. Using slam_gmapping, We obtain a 2D occupancy grid map from laser and odometry data collected by the rover. To use slam_gmapping, It needs a mobile robot that provides odometry data and is equipped with a

horizontally-mounted, fixed, laser scanner. The slam_gmapping node transforms each incoming scan into the odom (odometry) tf frame.

To function correctly the node needs:

- A laser scan source publishing sensor_msgs/LaserScan message;
- A tf transforms from the base link to the odom link;
- A tf transform from the laser link to the base link

The nodes take the following parameters:

- Base_link: The frame attached to the rover base, we set these parameters to the "/laser" as we are using only the range finder;
- odom _link: the frame attached to the laser link, we set these parameters as the laser link frame as they are identical

Since the base, odom and laser frames are identical there is no transform needed by the node. In addition to the map the node provides a transform from the odom frame to a map frame, we set the map frame as "/map". We use the obstacle avoidance node to provide information about the location of objects at 5 m or above on the map. The obtained clusters beyond that specific range will be processed to determine the obstacle position in the map frame.

The cluster position is defined by its closest point; we convert the coordinates of this point from polar to cartesian by projecting the range value on horizontal and vertical axes.

The coordinates are calculated as follow:

$$x_{obs} = R \times cos\alpha \quad (5\text{-}4)$$

$$y_{obs} = R \times sin\alpha \quad (5\text{-}5)$$

With R the range of the closest of cluster and α its angle. Figure 5-7 outlines the calculation process.

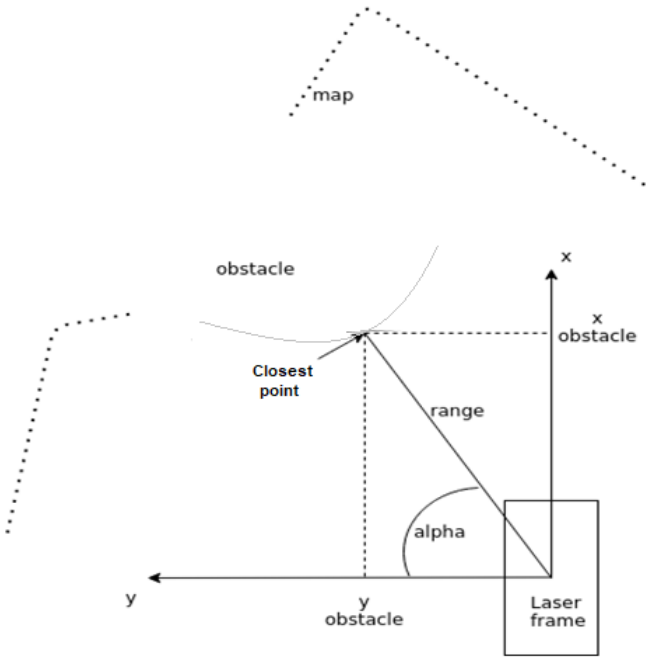


Figure 5-7. Object coordinate and location in the map

Therefore, we associate the position to a visualization marker that will show up in RVIS at the object location the position is formatted in a text message displayed. The visualization marker definition given by ROS documentation is presented in **Appendix III.** Furthermore, Figure 5-8 summarizes the node graph of mapping and obstacle avoidance

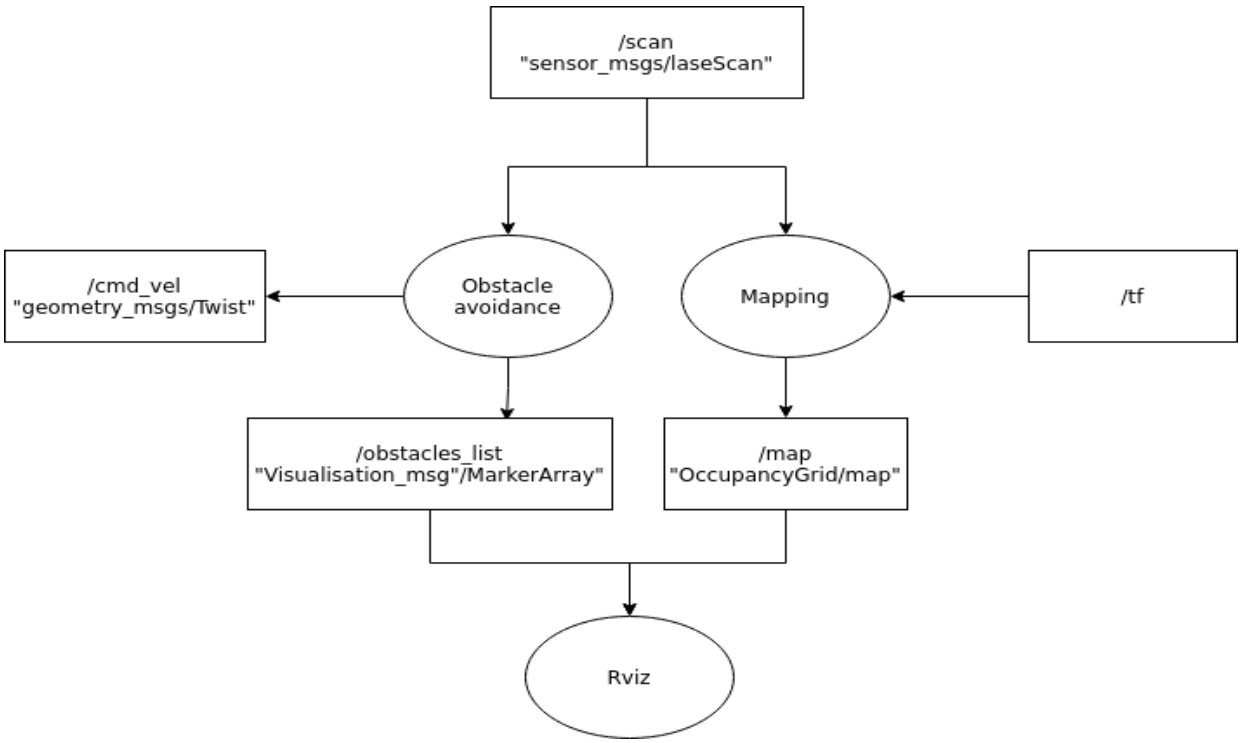


Figure 5-8. ROS node graph for Gmapping

ROS has a modular structure. Communication between nodes in a network is what makes it work. With a structure such as this, it is easier to implement the complex system required to control robots autonomously. It is because each node is designed for accomplishing its own specialized task. The task at hand is simplified by breaking the problem down into numerous smaller ones. Having each node independently controlled reduces the difficulty of debugging. When you know what to focus on for a given problem, the time spent on this process and the confusion you experience is greatly reduced. The ROS framework is described above. Users can create a system of nodes by using packages. Robot operating systems are made up entirely of this system of nodes. The nodes each perform a specific function and communicate with each other to control the robot. ROS nodes perform computations by definition. An executable is essentially a piece of software that communicates with other nodes in the system when it is being executed. By publishing information on a topic, nodes are able to communicate with each other. As shown in Figure 5-9, this is the case.

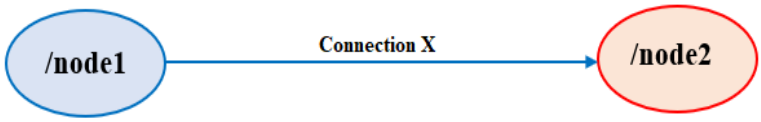
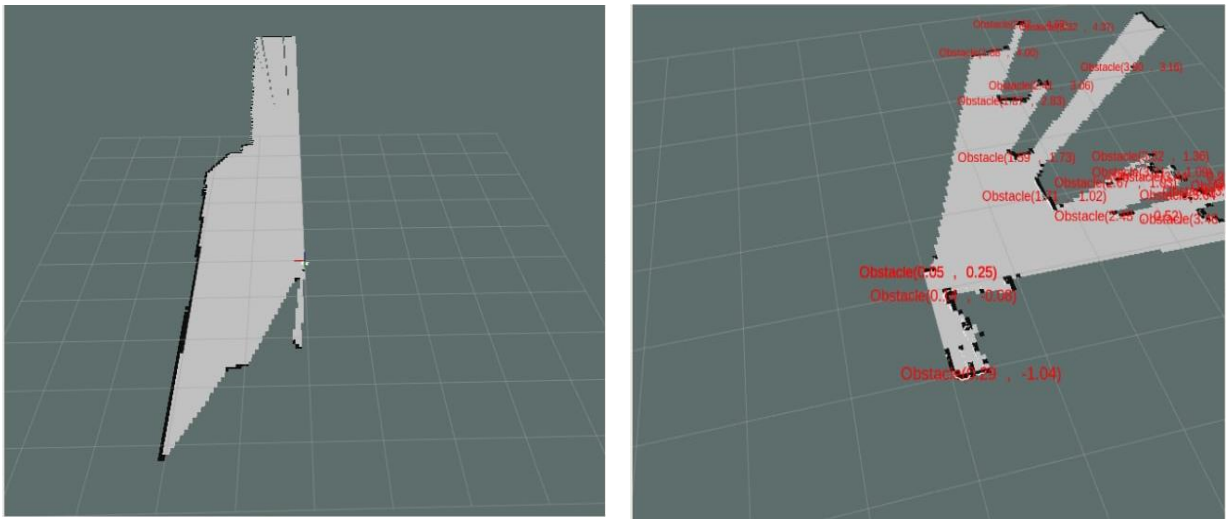


Figure 5-9. Communication from node to node via topics.

Each node has no idea whom it is communicating with. Nodes can subscribe to the data they publish to topics when they publish data. Nodes that subscribe to topics, such as /node1 Topic X /node2, will view all the data contained within those topics. Streaming communication between nodes is accomplished with the use of TCP or UDP, both of which fall under the Internet Protocol category. UDP sends data in packets and does not require a connection to send data while TCP sends data bidirectionally once a connection is established. ROS assigns a message type to each topic to format its sent and received messages. Data can be differentiated using these message types. Furthermore, ROS uses a concept known as a service to manage request-reply cycles. Messages that constitute a service consist of one item for your request, and another for your response. Nodes either wait for a request or send a response after using a service. Upon requesting an answer, the node waits for an answer. ROS utilizes bags as a means of storing data for offline use or other purposes. By subscribing to one or more topics, ROS bags are generated. Upon saving the data, it will be "bagged up" and sent to another node. Subscribing to a certain topic can be bypassed this way. Last but not least, one must know the Master to fully grasp the ROS architecture. During inter-node communication, the Master helps the nodes to locate each other. Topics and services can be matched between publishers and subscribers. This publisher-subscriber framework is possible with the Master because it maintains operational complexity. Therefore, in the current thesis, the gmapping in ROS was performed. The used Gmapping coding is provided in **Appendix IV**.

The G mapping method is based on SLAM and odometry data from a robot, which optimizes laser scanner data. Using this data, the algorithm generates a 2D grid map of the robot's operating environment [When the lack of odometry is accommodated by using the scanner's high update rate, the gaps in useful data can be filled. A map and a localization solution in an unknown environment are both viable options. Based on the Navigation Stack I mentioned above, this thesis explores the use of the ROS node G mapping in a simulation of a robot. Therefore, using the presented algorithm presented above that was utilized in the current thesis, a user can see laser scanner data, occupancy grids, as well as paths that have been planned locally and globally using this tool. Even though this tool has many capabilities, this research primarily used it for teleoperating the robot and viewing and evaluating the map. Robot goals can be chosen as waypoints on a map using Rviz. By subscribing to the desired pose, the ROS Navigation Stack navigates autonomously until the addressed goal is reached. By adding a mapping node to the G node, Rviz displays occupied grid messages. In this way, you can see the map is created and monitored in real-time. It is possible to measure the distance between two points on the map using the Rviz tool. Using this approach allows the distance between the mapped location of an object and an origin point (origin) to be measured. By taking advantage of the recorded data, G mapping can be evaluated for its validity and accuracy. Therefore, an example of used mapping with and without the objects' location and coordinate are presented in Figure 5-10.



(a) (b)

Figure 5-10. An example of obtained mapping in the current thesis a) without objects location and b) objects' location and coordinates

5.4. Application of ANN for navigation

Several human brain mathematical and software models, which are utilized to address a broad variety of science, engineering and application issues in different disciplines, were suggested in the field of ANN. Over the last few decades, usage in mathematical processes of intelligent systems, particularly ANN, has grown so prevalent that they may be regarded as fundamental and

common tools. ANNs have been incorporated into many forms of the computational model, each of which is suitable for a variety of applications. A mathematical structure and a set of adjustment parameters are considered in all such networks. Using a training process, this whole structure is tuned to offer acceptable performance. The model provides In the human brain the learning process is produced by weakening or consolidating the connections between the nerve cells. This mathematical learning is characterized by certain parameters called weights and it is stated as a setting. Figure 5-11 shows the whole network set up with numerous inputs and a single output.

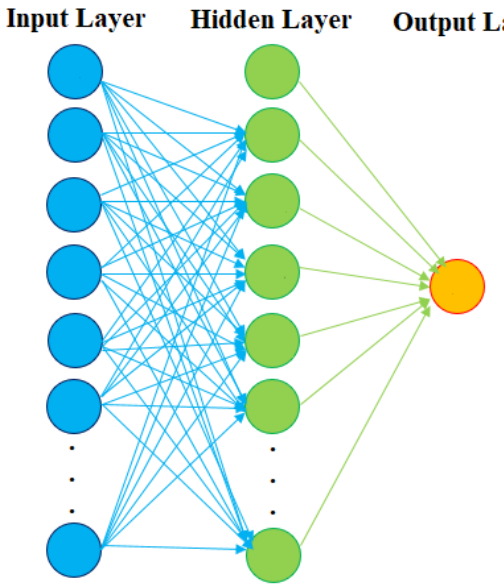


Figure 5-11. General network design with several inputs and one output

There are three levels in the used network in this thesis. To select data on the network, one layer termed the input layer, is utilized. The buried layer includes neurons that are linked to all input units. As concealed layers, several layers can be employed. The last layer (output layer) is made up of the neurons producing the model output. Each neuronal link has a weight. Each of the neurons in the hidden layer and output layer conducts two input operations which are shown in Figure 5-12.

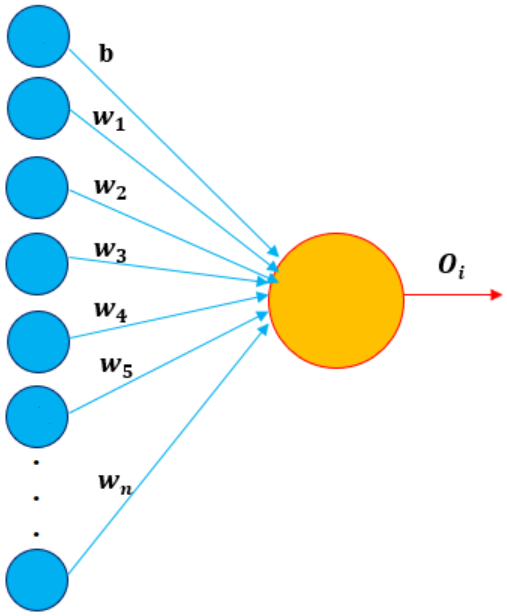


Figure 5-12. Structure of every neuron in the hidden layer and output layer

The first thing that was computed as the weighted total ($\sum$) of the inputs ($x_i, i = 1,2, \dots, n$) and deviation (b). The value was mapped using an activation function to other space (f). The neuron output I was therefore computed according to the following formula:

$$O_i = f(b + \sum_{j=1}^{n} w_j x_j) \quad (5\text{-}6)$$

In numerous rounds, neural network training with a considerable influence on network performance is performed. The network error value is calculated according to training data in each iteration of the loss function. The brain system aims to learn the right weight to minimize loss. Backpropagation of errors is one of the most used ways of neural networking training. The first stage is to feed the input data into weights and subsequent putting them into a deviation. This process comprises two parts. The first step is The network output is computed after this phase, which is possibly different from the current result. The loss function value is this difference. In the second stage, termed reverse propagation, weight and deviations are modified by the value of the loss.

Many genuine word problems require enormous computer resources and time complications to find a suitable answer (sub-optimal). To overcome these issues, optimization procedures like statistical, mathematical and stochastic programming must be used.

These techniques have been used to investigate and improve the fitness of the answer for a better generation. CMA-ES is one of the low-time complexity evolutionary algorithms used to optimize continuous domain issues. As follows, the life cycle of CMA-ES is:

1- Initialization;
2- Set the mean, covariance and step size of each measure;
3- Generate descendants;
4- Process for selection and recovery

5- Go to step 2 if you do not meet the exit requirements.

The first stage involves initializing the CMA-ES settings. Includes these parameters:

- D: number of dimensions of the issue.
- λ: The population size of offspring is generally determined by the formula $4 + 3Log(D)$.
- μ: Next-generation parent population and may be estimated with the formulation "free of money" and "free." Parents are chosen from the finest options for each generation. The following vector is provided to each individual population:

$$X_i = [x_1, x_2, x_3, \dots, x_D] \; for \; i = 1,2,3, \dots, \mu \tag{5-7}$$

- M and σ: Initial mean and default deviation and update correspondingly for each generation.
- P_c and P_σ: Path of evolution and cumulative step size.
- C: covariance matrix expressing the dependence between distribution variables. The initial value for C is the identity matrix (I).
- g: Contra generation.
- c_{cov}: Learning rate of covariance.
- c_c: The rate of learning for the upgrade of the cumulation of the covariance matrix is less than one.
- μ_{eff}: Variance efficient middle mass. It can be equal to $\lambda/4$.
- μ_{cov}: greater than zero and can be equal μ_{eff}.
- d_σ: Damping parameter can be about one step-size update.
- c_σ, The cumulation step control learning rate is less than one.
- $N(0, I)$: Normal multivariate distribution of zero average covariance and unity matrix.

In the second stage, the search distribution must update the covariance matrix, step size and mean value:

- In the second stage, the search distribution must update the covariance matrix, step size and mean value:

$$C^{(g+1)} \leftarrow (1 - c_{cov})C^g + \frac{c_{cov}}{\mu_{cov}} P_c^{(g+1)} P_c^{(g+1)^T} + c_{cov}\left(1 - \frac{1}{\mu_{cov}}\right) \times \sum_{i=1}^{\mu} w_i \left(\frac{X_{1:\lambda}^{(g+1)} - M^g}{\sigma^{(g)}}\right)\left(\frac{X_{1:\lambda}^{(g+1)} - M^g}{\sigma^{(g)}}\right)^T \tag{5-8}$$

Where

$$P_c^{(g+1)} = (1 - c_c)P_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{eff}}\,\frac{M^{(g+1)} - M^{(g)}}{\sigma^{(g)}} \tag{5-9}$$

- The following formula updates the step size:

$$\sigma^{g+1} \leftarrow \sigma^g \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|P_\sigma\|}{E\,N(0,I)} - 1\right)\right) \tag{5-10}$$

In the third stage, the following formula is used to produce the sample population for the next generation:

$$x_k^{(g+1)} \backsim N\left(M^{(g)}, \left(\sigma^{(g)}\right)^2 C^{(g)}\right) \; for \; k = 1, \dots, \lambda \tag{5-11}$$

Then, in the fourth phase, the following formula will update a certain new generation (μ):

$$M^{(g+1)} \leftarrow \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} \tag{5-12}$$

$$\sum_{i=1}^{\mu} w_i = 1 \qquad w_i > 0 \; for \; i = 1, 2, \dots, \mu \tag{5-13}$$

5.4.1. Models Robustness

The sensitivity analysis was also carried out to assess the accuracy of prediction ANN models. The reciprocal information between the two functional rooms defined the mRMR function. Increasing the probability of two vectors being shared leads to increased interaction. The mutual information between the two variables x and y is calculated as below, based on the probability density $p(x), p(y)$, and $p(x, y)$.:

$$I(x, y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)}\right) \tag{5-14}$$

The function x_i is picked based on the maximum relevant technique if $I(x_i, t)$ is higher than other features of the objective feature t. Maximum relevance is one of the ways of searching for optimum features, as given forth in Equation (5-15), based on the average of all common quantities of information between the different x_i function and goal feature t.

$$\max D(S, t), \quad D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i, t) \tag{5-15}$$

The attributes that are most pertinent to the target feature are selected according to Equation (5-15). The significance of the specified characteristics might also be significant. To decrease repetition, Equation uses mutual information between the specified functions (5-16)

$$min \; R(S), \qquad R = \frac{1}{|S|^2} \sum_{x_j, x_i \in S} I(x_j, x_i) \tag{5-16}$$

The two Equations (5-15) and (5-16) are merged to form Equation (5-17) to get the best feature set based on the mRMR-algorithm.

$$\max_{x_j \in X - S_{m-1}} \left[I\left(x_j, c\right) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j, x_i)\right] \tag{5-17}$$

In which, m is the designated features number of the set S. Also, X is the features vector.

In Figure 5-13, we can find the whole research flow chart. After the data set was supplied, the mRMR algorithm prioritized the features. The following equations have been computed at each phase of model assessment criteria comprising MAE, MSE and run-time:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |T_i - O_i| \tag{5-18}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (T_i - O_i)^2 \tag{5-19}$$

Where, n is the number of samples. T_i and O_i are the target and model output for the i-th sample, respectively.

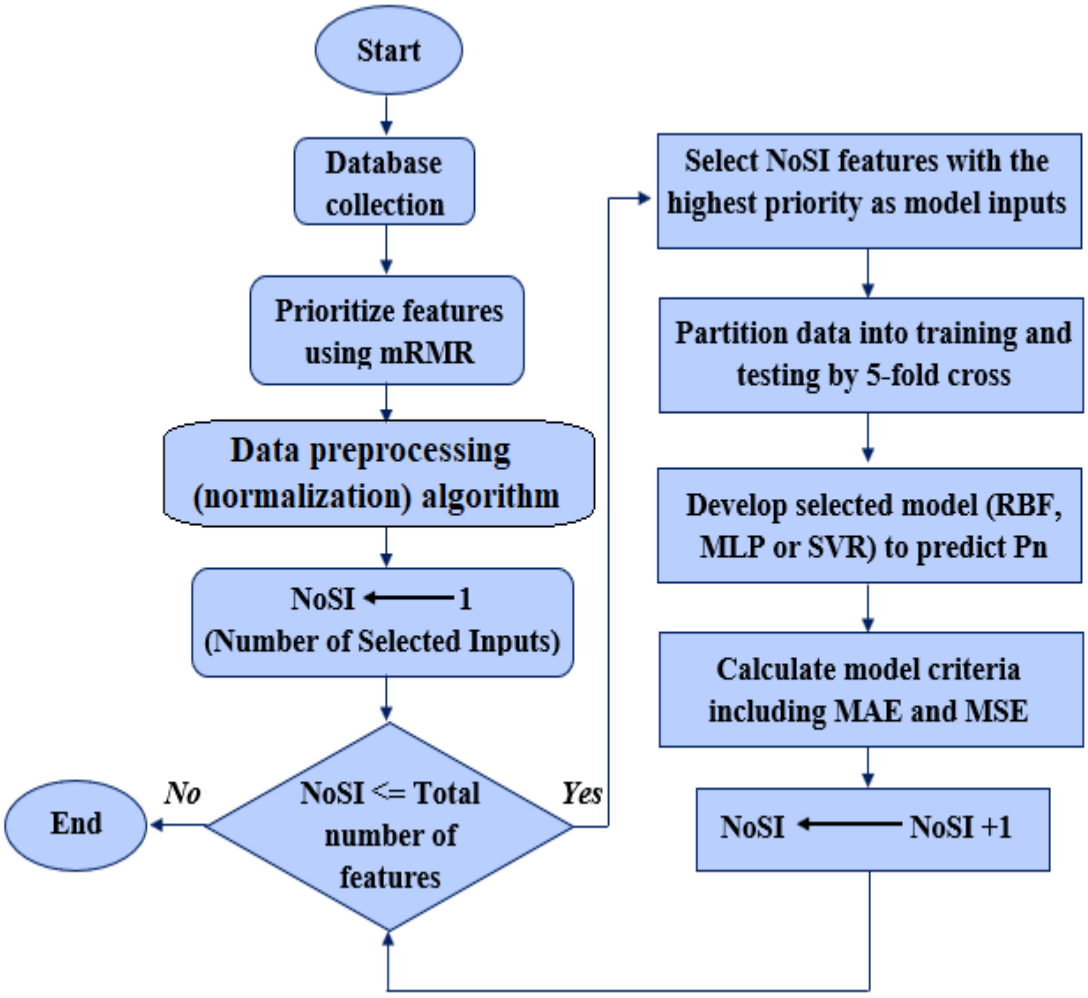


Figure 5-13. General research flowchart

Therefore, ANN was utilized in MATLAB software to predict the location of objects based on their distance. The main advantage of using ANN in auto-driving cars is eliminating the costly LiDAR sensor. As a clear issue, using a LiDAR sensor is costly and high tech equipment. Therefore, in the light of growing the ANN applications, prediction algorithms could be used with a simple camera to find the location of objects. It should be noted that in the current thesis, only the ANN algorithm are presented, discussed and their accuracy was examined. So, in future works, a camera could be linked to the presented ANN algorithm which was out of the scope of the presented thesis. In the current thesis, three defined algorithms bt MATLAB suggested as high accurate tools, Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate were utilized. There are many different algorithms in MATLAB software but the previous investigations showed that the accuracy and performance of Levenberg-Marquardt, Bayesian regularization, Scaled Conjugate were superior to others. In this thesis, 2100 objects location including angle and distance

were utilized, 80% of data was utilized for training and 20% of data was used for the test. The performed coding for artificial neural network prediction is presented in **Appendix V**.

CHAPTER 6

RESULTS AND DISCUSSION

6.1. Introduction

This study aims to measure object detection using a LiDAR sensor, UTM-30LX model in Politecnico di Turino, Italy. In this chapter, the obtained results are presented and discussed. For this aim, presented algorithms and theories in the previous chapter have been followed. Following the previous chapter, the obtained results are sorted using the steps below, as shown in Figure 6-1:

1- The LiDAR sensor was installed and connected to the computer;
2- The ROS was utilized to find the location of the object and object detection discussed in Chapter 2;
3- The data was sorted using the LiDAR sensor;
4- After object detection and finding the distance between the sensor and real object, the accuracy of the sensor was evaluated by measuring the exact distance using a tape measure;
5- The results were converted in ROS and maps including the location and coordinate of objects were determined using Gmapping using algorithms discussed in Chapter 3;
6- The obtained results were then used in MATLAB for developing an artificial neural network using algorithms presented in Chapter 4.

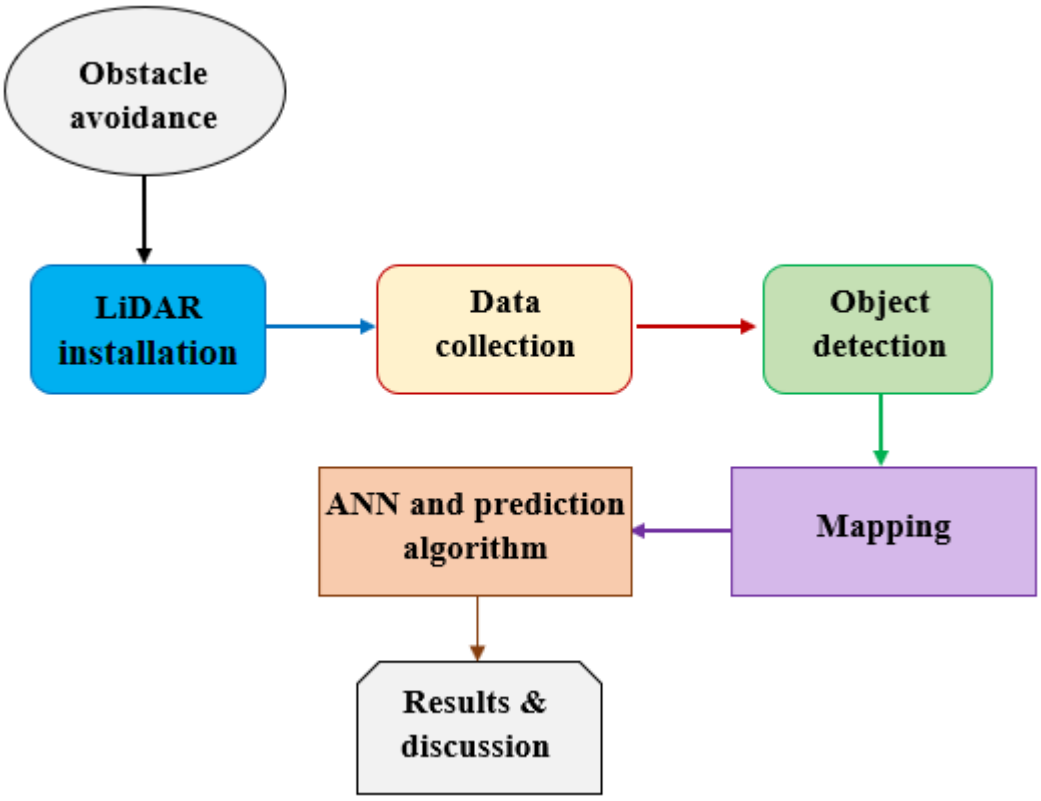


Figure 6-1. Overall overview of investigation performed in the current thesis and in this chapter

6.2. LiDAR installation

In the current thesis, a LiDAR sensor, the UTM-30LX model is a highly accurate and common sensor, as illustrated in Figure 6-2.

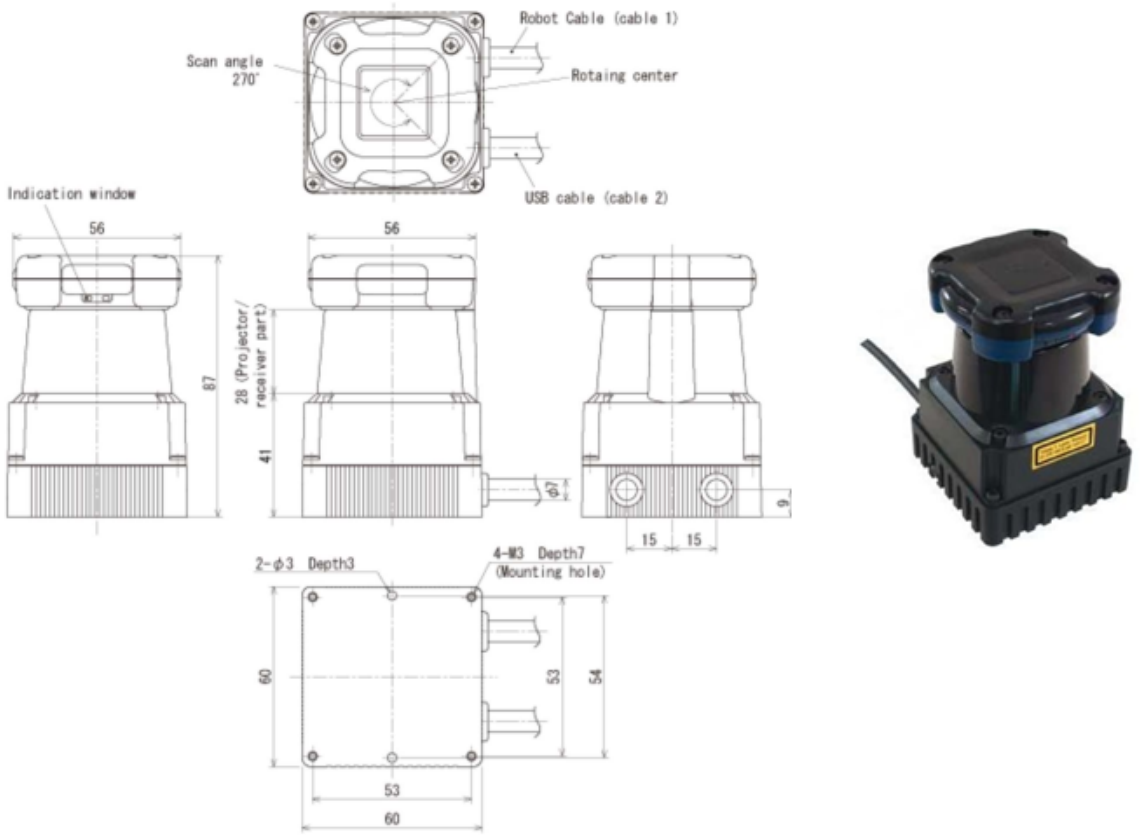


Figure 6-2. Used LiDAR sensor

The UTM-30LX-EW scans a 270-degree semicircular field using a Laser Source (Ţ=905nm) (Figure 6-3). It calculates the distance between objects in its range for each angular step. The motion-based technique was utilized to detect objects. Though, in experiments, all sorts of moving objects could be measured and present in a scene. In addition, in certain claims (eg independent driving), the detection of moving objects and the size of speed by scheming numerous pulse beats using standard LiDAR pulse scans are slow and error-prone. Consequently, preceding studies aim to discourse these barriers and change a modelled detection-based tracking technique for sensing and tracking moving objects in point clouds achieved by an advanced UTM-30LX model LiDAR sensor. This sensor can not individually collect three-dimensional spatial data, but the comparative speed between moving objects and the sensor in the direction of the stright direction.

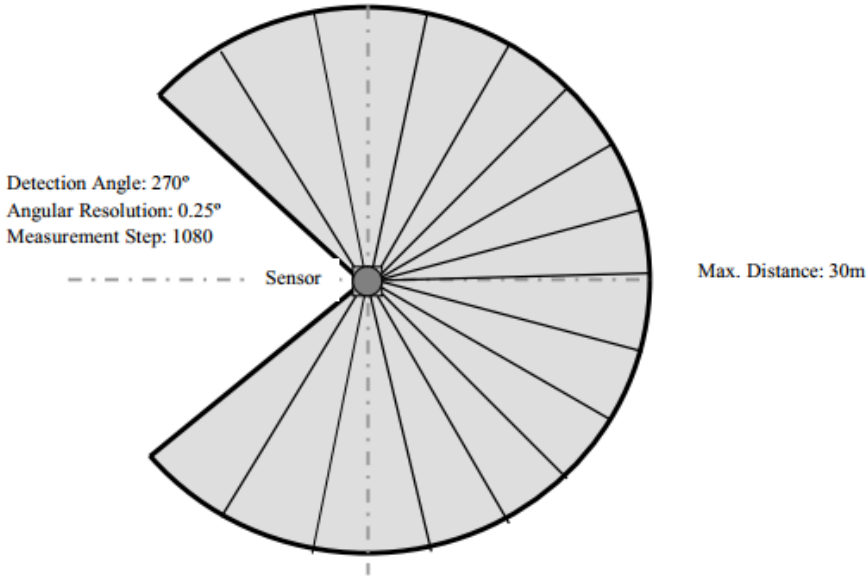


Figure 6-3. Diagram of the scanned area of used UTM-30LX model sensor in this thesis

To begin the data collection, first of all, to collect data, the LiDAR sensor was connected to the computer, as shown in Figure 6-4.

Figure 6-4. Used sensor and connection to the computer

The exact connection between the LiDAR sensor and computer is a very sensitive and important step. A driver for connecting the scanners is already available in ROS. An interface is known as urg_node. Our URG series sensors are supported by our urg_node already in the Robot Operating System (ROS). To start using the hokoyou utm LiDAR. First of all its primarily packages should be installed on the Ros. The robotic operating system has already dedicated some

specific node .called urg_node. For interfacing this scanner. Therefore, a total of six-step were carried out as shown in Figure 6-5.

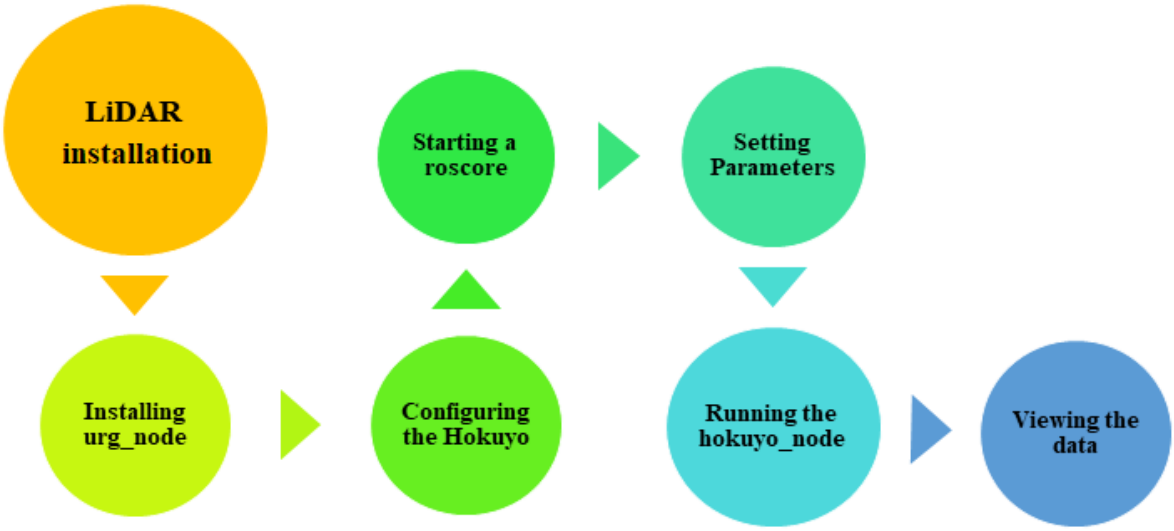


Figure 6-5. LiDAR-computer connection steps

Furthermore, the commend carried out in each step are defined for connecting the LiDAR sensor to the computer, as below: :

6.2.1. Installing urg_node

To start data retrieving by the sensor, we have to urg_node using the next code:

```
sudo apt-get install ros-${ROS_DISTRO}-urg-node
```

6.2.2. Configuring the Hokuyo

Make sure that the hokuyo_node will be able to access the Hokuyo laser scanner to configure the sensor with urg_node. To provide this connection and configuration, the following code was used:

```
$ ls -l /dev/ttyACM0
```

6.2.3. Starting a roscore

After the connection, we need to open the ROS master for data collection. To make to accurate collaboration between the ROS master and data collection packages ROS core was employed using the next code:

```
$ roscore
```

6.2.4. Setting Parameters

Before we can run the hokuyo_node we need to make sure that we have the correct configurations loaded on the parameter server.

```
$ rosparam set hokuyo_node/calibrate_time false
```

6.2.5. Running the hokuyo_node

Now, the sensor should be turn on and data collection should be started using the next code:

```
$ rosrun hokuyo_node hokuyo_node
```

6.2.6. Viewing the data

To observe that everything is working and data is being published to ROS, the next code was utilized:

```
$ rosrun rviz rviz -d `rospack find hokuyo_node`/hokuyo_test.vcg
```

6.3. Data Collection

In this section, an object, located at a 2 m distance from the sensor, was considered and the data was sensed using the sensor to control the accuracy of the LiDAR sensor. Figure 6-6 shows the provided results for one 270°C rotation cycle of the sensor. In this figure, the centre shows the location of the sensor and the red point showed the obtained results. Also, the external circle shows the area covered in a rotation phase by the sensor. Therefore, the obtained results were compared with the exact location of the obstacle (2m), as illustrated in Figure 6-7.

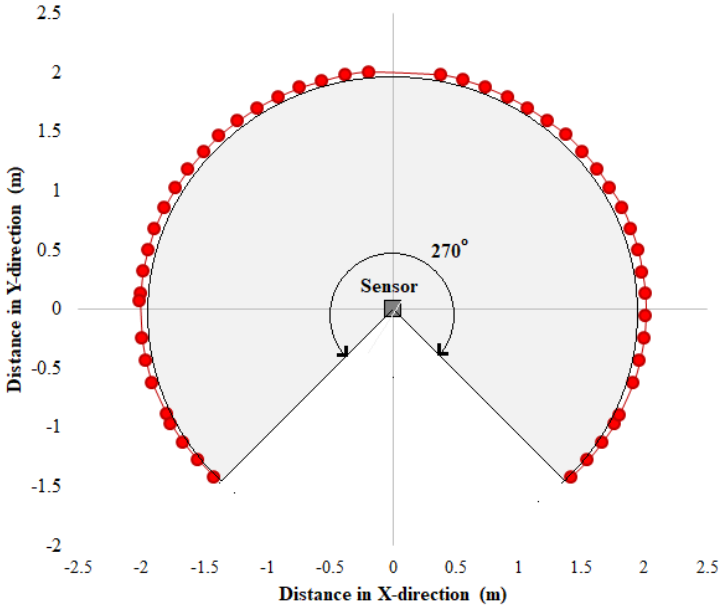


Figure 6-6. Performance of sensor in measuring the distance

In Figure 6-7, the red points show the exact distance between the obstacle and sensor and the blue points show the obtained distance between the sensor and obstacle using the sensor. There, there is an insignificant difference between the real and obtained results up to 0.017 m. In addition, the minimum error distance was obtained at 0.005 m. These results showed that the accuracy of the sensor is acceptable and the sensor works well.

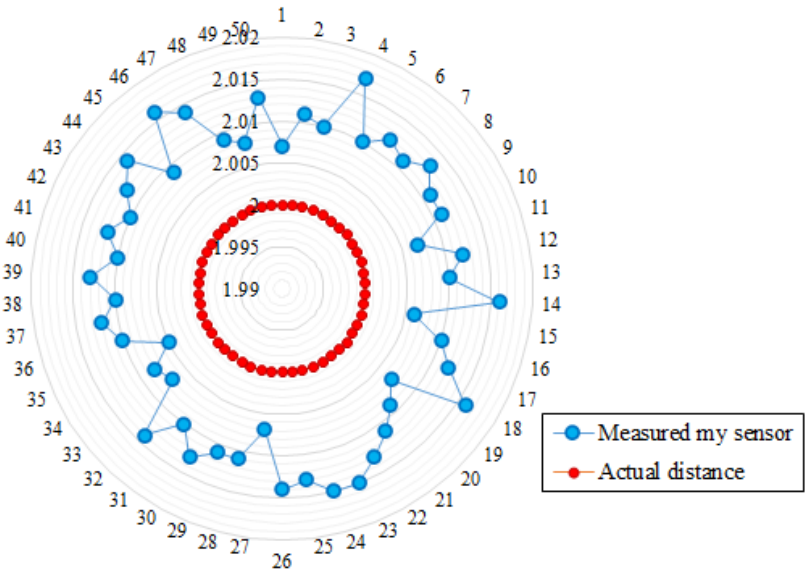


Figure 6-7. Comparison between the exact values and those obtained using sensor

6.4. Sensor accuracy

In this section, to make sure about the performance of the sensor, the accuracy of the obtained results is measured. For this aim, different techniques were utilized including: mean squared error (MSE), normalized mean squared error (NMSE), root mean squared error (RMSE), mean absolute error (MAE), normalized mean absolute error (NMAE) and mean absolute percentage error (MAPE), according to Equations. (6-1) to (6-5):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(C_{ki} - \tilde{C}_{ki}\right)^2 \tag{6-1}$$

$$NMSE = \left[\frac{\frac{1}{n}\sum_{i=1}^{n}\left(C_{ki} - \tilde{C}_{ki}\right)^2}{\max(C_{ki}) - \min(C_{ki})}\right] \times 100 \tag{6-2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(C_{ki} - \tilde{C}_{ki}\right)^2} \tag{6-3}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|C_{ki} - \tilde{C}_{ki}\right| \tag{6-4}$$

$$MAPE = \frac{1}{n}\left[\frac{\sum_{i=1}^{n}\left|C_{ki} - \tilde{C}_{ki}\right|}{\sum_{i=1}^{n}\left|C_{ki}\right|}\right] \times 100 \quad (6\text{-}5)$$

In Equations (6-1) to (6-5), C_{ki} indicates the measured value and $\tilde{C}_{ki}$ represents the estimated value ($k = c$, t and r as a representative of C_c, C_t and C_r, respectively), n is the number of considered data and $\bar{C}_k$ is the average of the measured values. Also, $\bar{\tilde{C}}_k$ shows the average of the estimated values. The results are provided in Table 6-1. Regarding this table, all used models showed the low error between the real distance and those obtained by the sensor. In this among, the NMSE showed the highest value by about 1.09884 value for the error.

Table 5-1. Investigation the accuracy of collected data

Model	MSE	RMSE	MAE	MAPE (%)	NMSE (%)
Results	0.00013186	0.011483031	0.011139998	0.011078292	1.098847537

Table 6-1 provides the general error between the exact distance and those obtained by using the sensor. For a better understanding of the accuracy of the obtained results, a side-by-side comparison was carried out between the exact and obtained results. The error distribution of the side-by-side comparison was also demonstrated in Figure 6-8. According to Figure 5-5, the error between each real and it's corresponding obtained result is also low. The maximum distribution of 16 was observed with the error ranging between 0.0049 and 0.0055 (or 0.49% and 0.55%) which showed the high accuracy of the collected data.

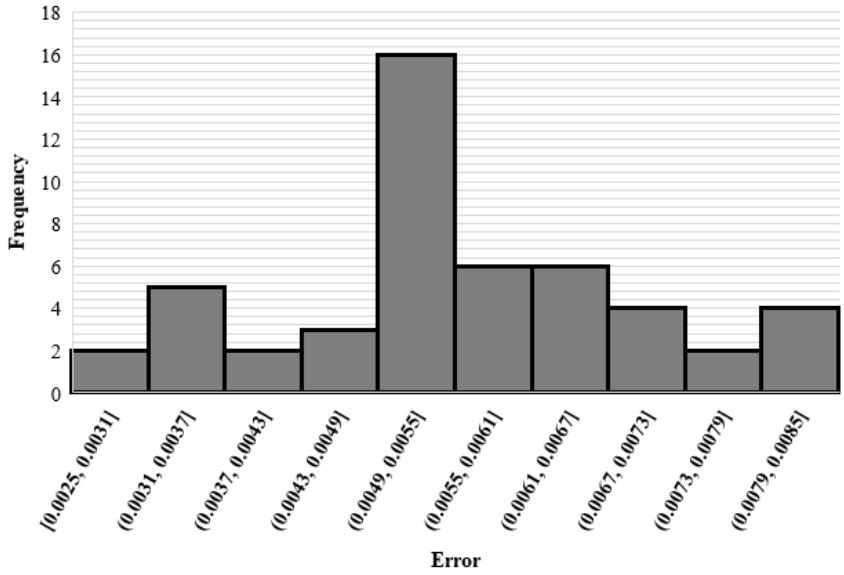
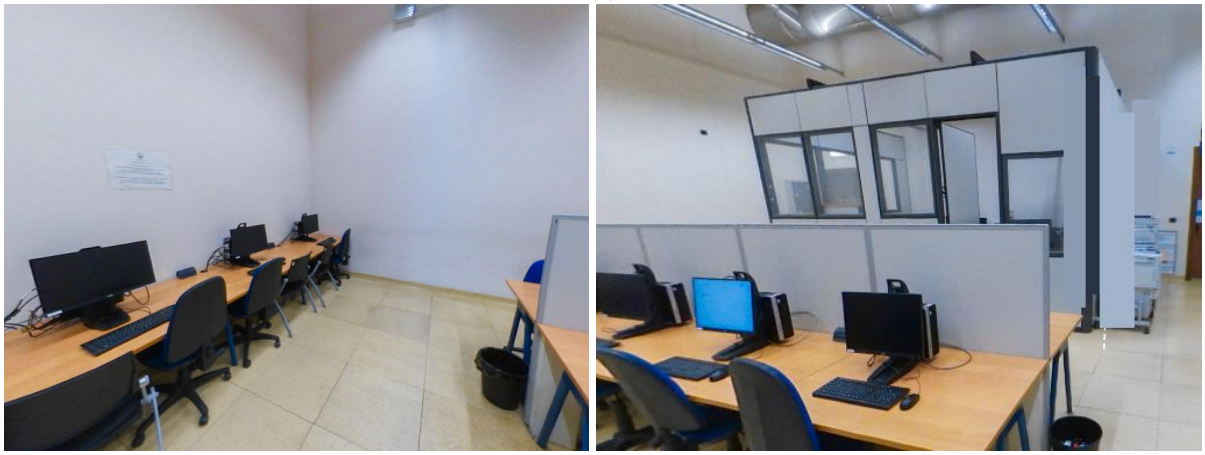


Figure 6-8. Error distribution for side-by-side comparison between the real and obtained results in terms of percentage (both frequency and absolute error are with no unit)

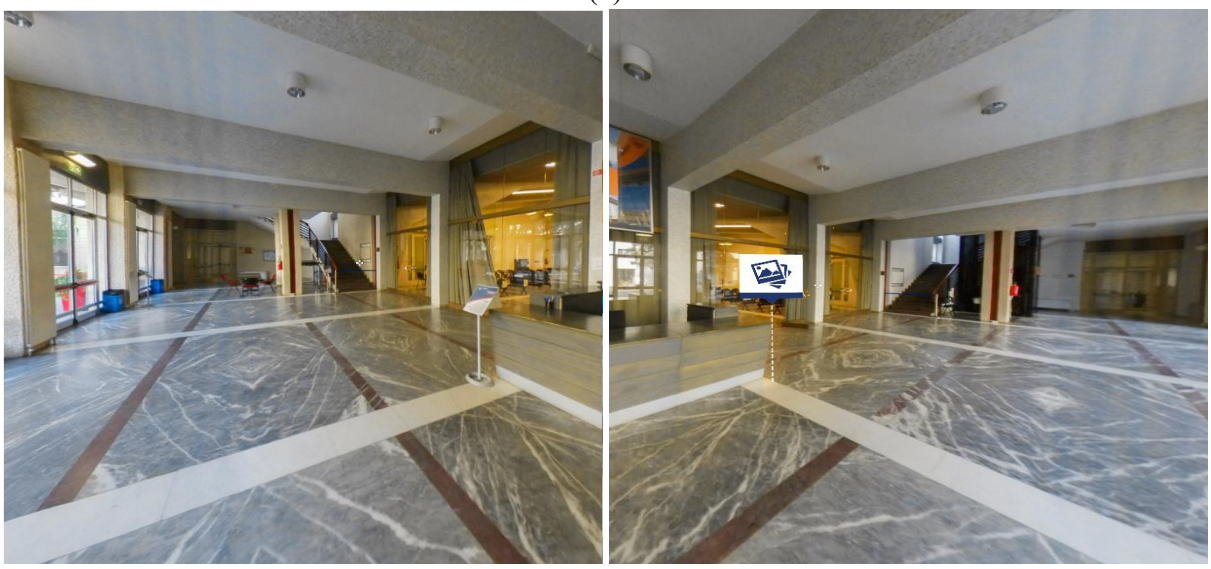
6.5. Data collection results

After finding the high accuracy of the used sensor and performed coding in this thesis, the data collection was developed for different places and again the accuracy of the results was measured. For this purpose, three open-loop experiments were run with the LiDAR sensor was driven around different locations on the Politecnico di Turino campus, Italy. The first case was a comparatively simple case in which the LiDAR was moved along a straight path in a corridor with two desks serving as obstacles at different locations. The desks were placed on the side of the sidewalk and corners and the UGV was driven between them. The desk was white and the sidewall (map) was grey. In the second case, a larger number of obstacles including a white desk, black chair and other laboratory equipment were considered and the LiDAR sensor was driven around the lab site. Finally, the third case was the main hall with white walls. In this case study, a large brown wood wall was kept vertically with the help of a person in a different location of the hall. Then the LiDAR sensor was driven around different locations and the data was sorted. These three cases were considered as cases 1 to 3, respectively. The considered cases location is presented in Figure 6-9.

(a)



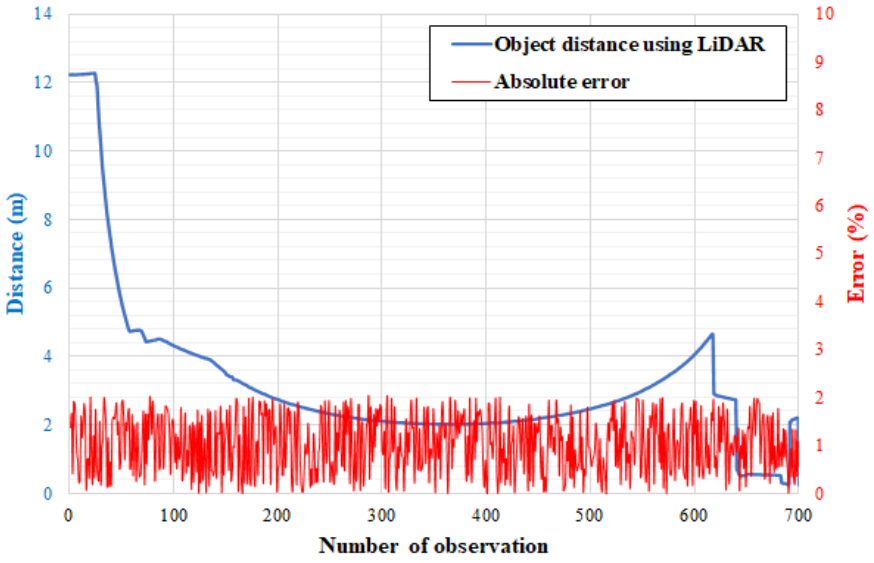
(b)



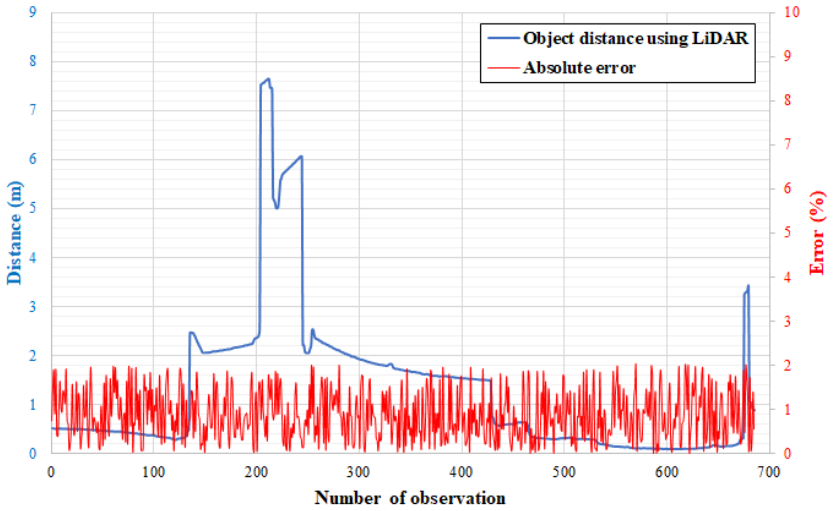
(c)

Figure 6-9. Considered cases study in the current thesis a) case 1: corridor, b) case 2: computer lab and c) case 3: main hall

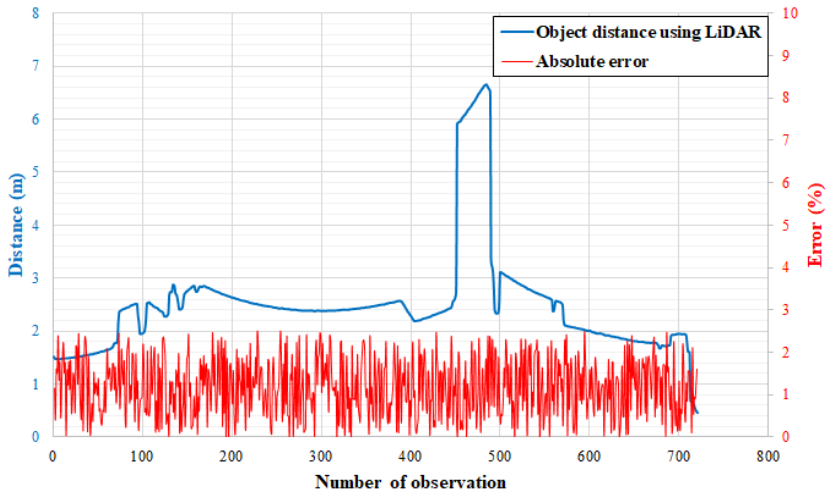
The obtained results are presented in Figure 6-10 for three cases studied. In this figure, the measured values by the LiDAR sensor and the error between the exact value obtained using a tape measure and LiDAR results are presented. Regarding this figure, the maximum absolute error of 2% was obtained for object detection using the mentioned LiDAR sensor for objects from different distances, near and far from the sensor location. Therefore, the installed LiDAR sensor worked well and the obtained results had a high accuracy in terms of distance. To better understanding about the error between the detected distance and actual results, the error distribution was determined using a Histogram chart, as shown presented in Figure 6-11 for three studies cases.



(a)



(b)



(c)

Figure 6-10. Collected data and the absolute error between the obtained results using LiDAR sensor and the exact distance for three cases a) case 1, b) case 2 and c) case 3

As shown in Figure 6-11, the maximum error of 2%, 1.1% and 2% was obtained respectively for cases 1, 2 and 3, respectively, which show the high performance of the used LiDAR sensor. In addition, there is almost the same error distribution and amplification for case 1. However, the error distribution in case 3 particularly in the case was more concentrated on the left side of the graph which shows that error was slightly increased with increasing the distance from the sensor. Because the distance of the detected object in case 1 was slightly higher than those detected in cases 2 and 3, relative to the location of the sensor. It should be mentioned that in Figure 6-11, the distribution means the number of errors based on their value show in the horizontal axis.

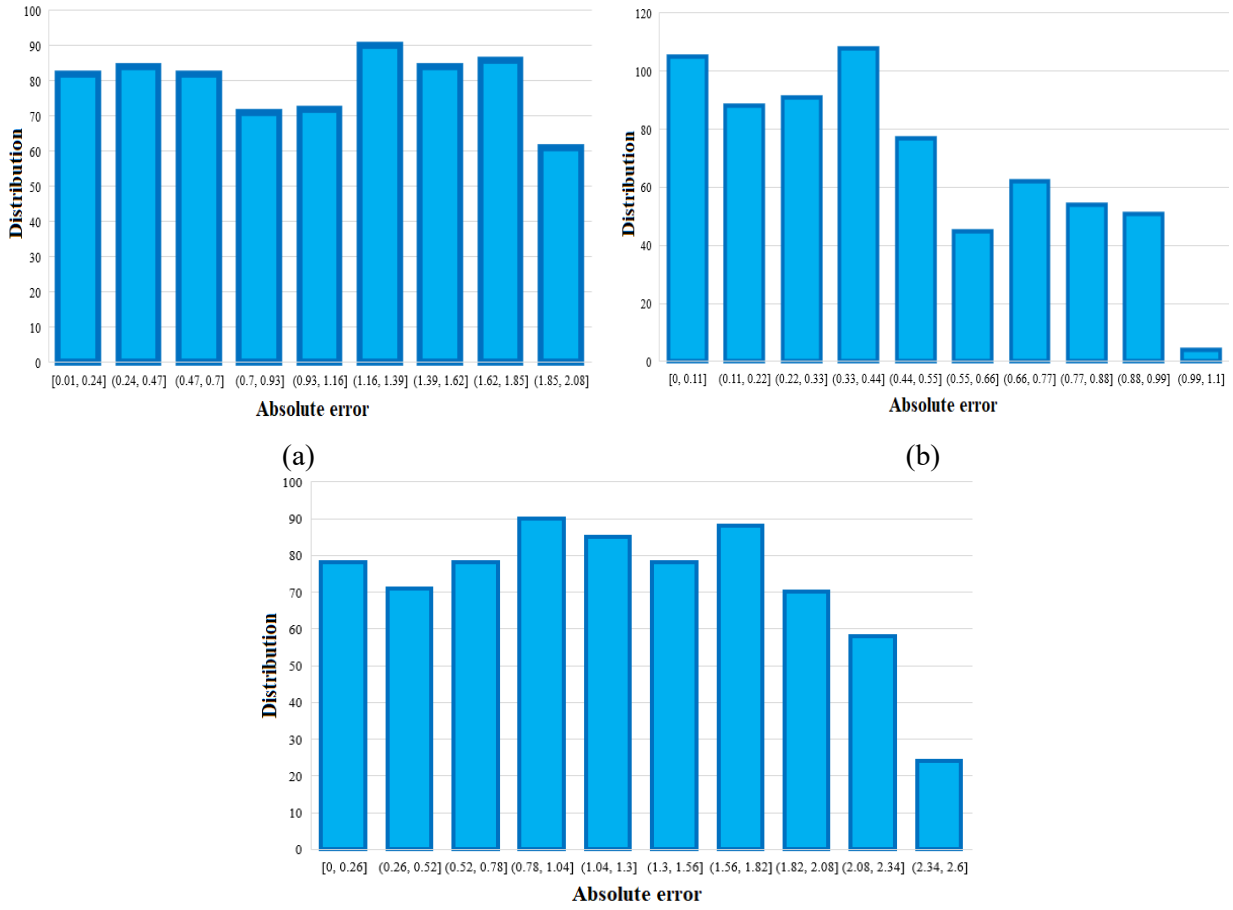


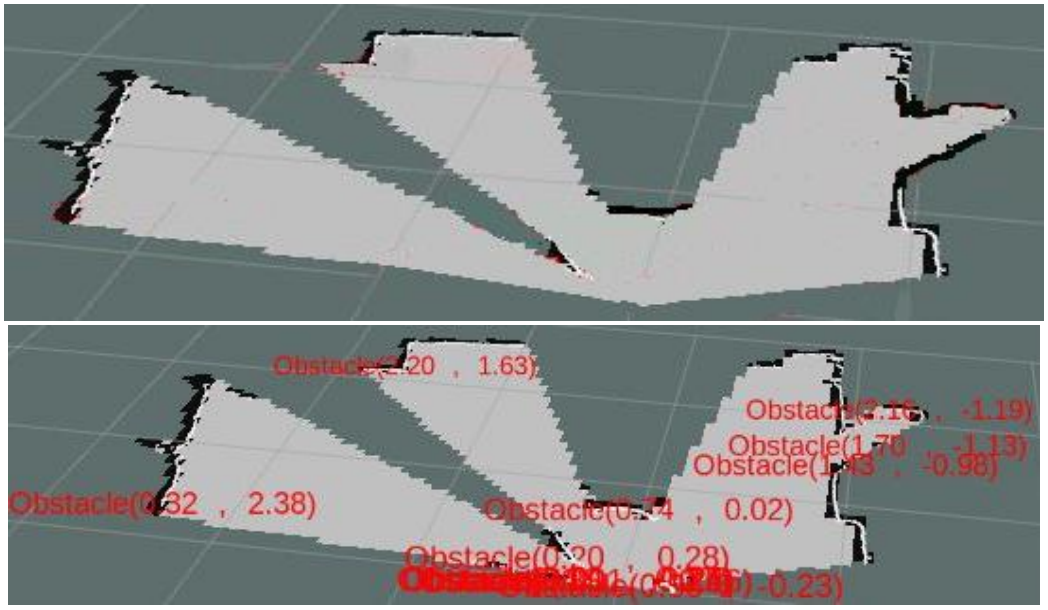
Figure 6-11. Comparison between the exact values and those obtained using the sensor for three cases a) case 1, b) case 2 and c) case 3 (both frequency and absolute error are with no unit)

6.6. Mapping

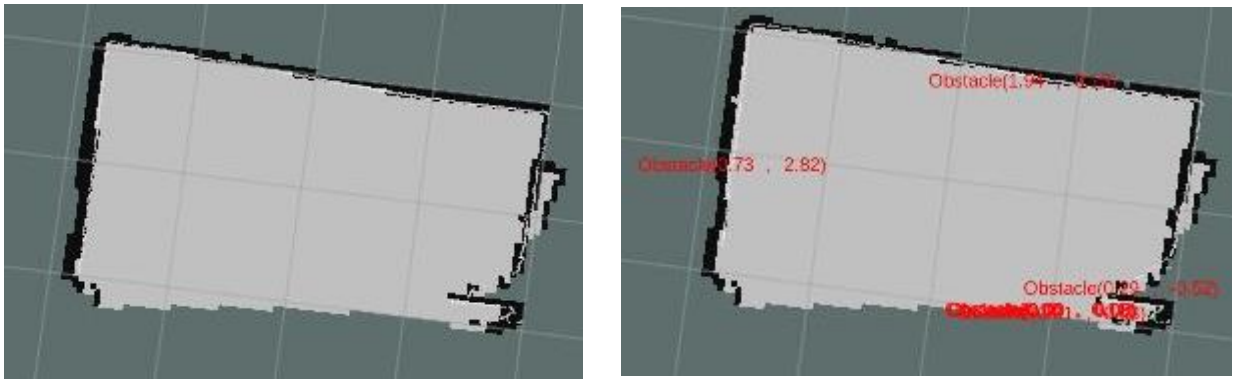
Up to now in this chapter, the accuracy of the employed LiDAR sensor and the results of the detected object were presented and discussed. As mentioned in previous chapters, one of the main aims of this thesis is mapping and finding the location of detected objects on the map. As discussed in Chapter 3, the SLAM package in ROS was utilized to create maps. In this investigation, three cases were studied as maps. The obtained results were used and mapping was carried out. The results are presented in Figure 6-12 for three studied cases with and without object location.

According to Figure 6-12, three different simple and complex maps were utilized. In the left-hand side figures, only maps were presented with no objects location. According to these figures, the presented algorithm and utilized LiDAR sensor could be utilized for mapping accurately. Additionally, the sensor was able to detect the corners and deference location shapes precisely. Conversely, in the right-hand side figures, the location and coordinate of objects are presented. Regarding these figures, not only is the LiDAR sensor able to find the location on

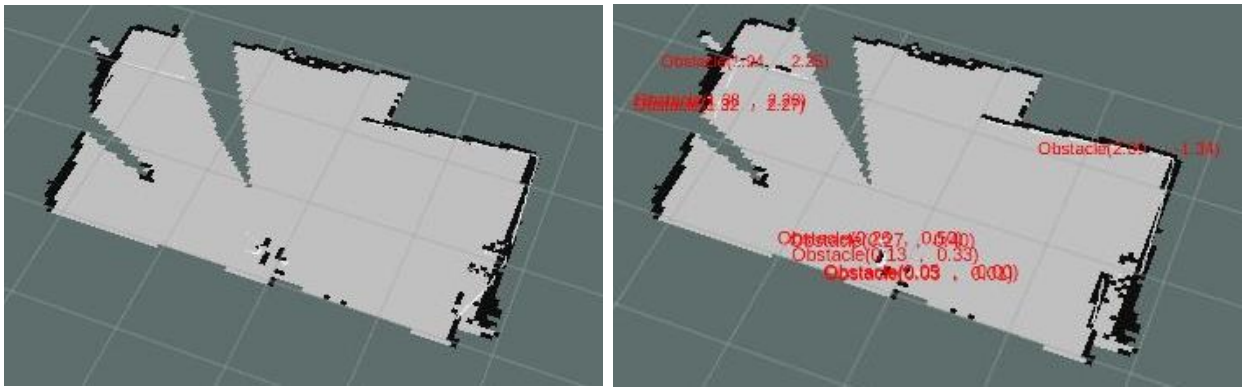
objects but also it is able to determine their coordinate in terms of x and y-axis. As a result, the presented algorithm in the current thesis and coding (Chapter 3) could be utilized for mapping and object detection which helps to find important issues such as pedestrian location around a car considering the street route and other places around an auto-driving car.



(a)



(b)

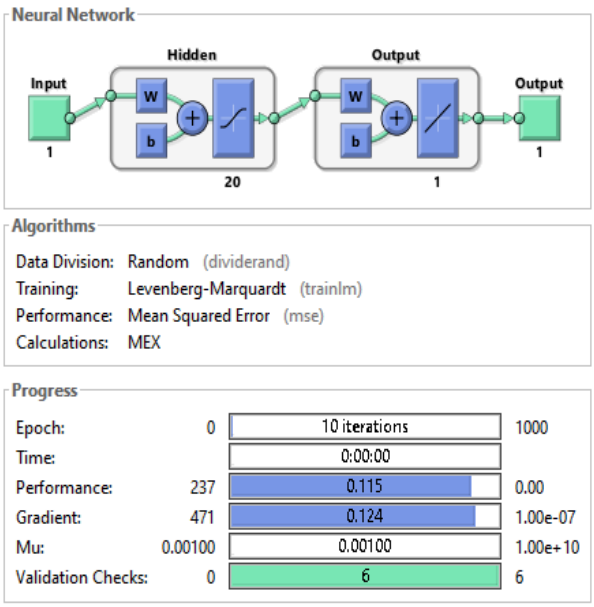


(c)

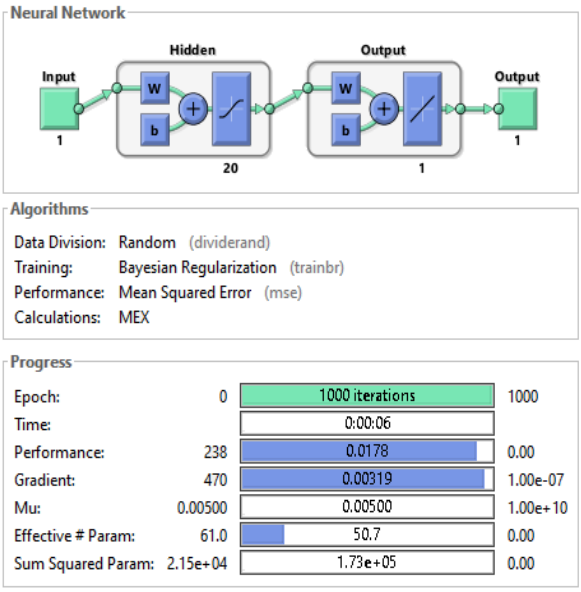
Figure 6-12. Mapping and object coordinations for studied cases a) case 1, b) case 2 and c) case 3

6.7. Artificial neural networks performance

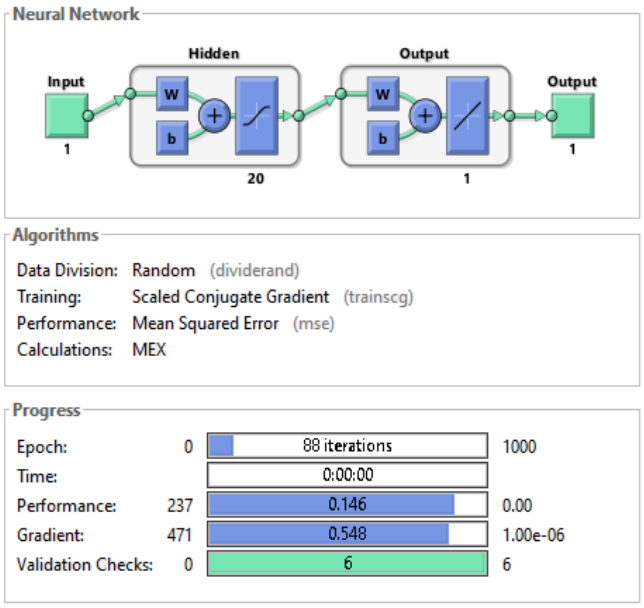
In the previous section, the LiDAR performance, object detection and mapping results were presented and discussed. To replace a LiDAR sensor with a simple camera, an ANN algorithm should be utilized to connect the simple camera to the computer for object detection prediction. Therefore, in this section, three ANN algorithms are presented and their efficiency and accuracy in comparison with the real-time detection by LiDAR sensor are measured. For this aim, three Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate algorithms are used, as described in Chapter 5 and its coding in **Appendix V.** The results of networks performance are demonstrated in Figure 6-13. According to Figure 6-13. Levenberg-Marquardt showed the fast-tracking performance in object detection. This issue helps an auto-driving car to detect objects in a very short period which increases a high reaction behaviour for a driver and higher safety. This could be associated with Levenberg-Marquardt because this is a newly developed algorithm with MATLAB company. There, Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate detected the distance of objects using 10, 1000 and 88 iteration which show the high performance of Levenberg-Marquardt as a fact object detection, relative to two other algorithms. To better understanding the performance of algorithms for object detection, their performance was measured as well and the mean square error, as illustrated in Figure 6-14.



(a)



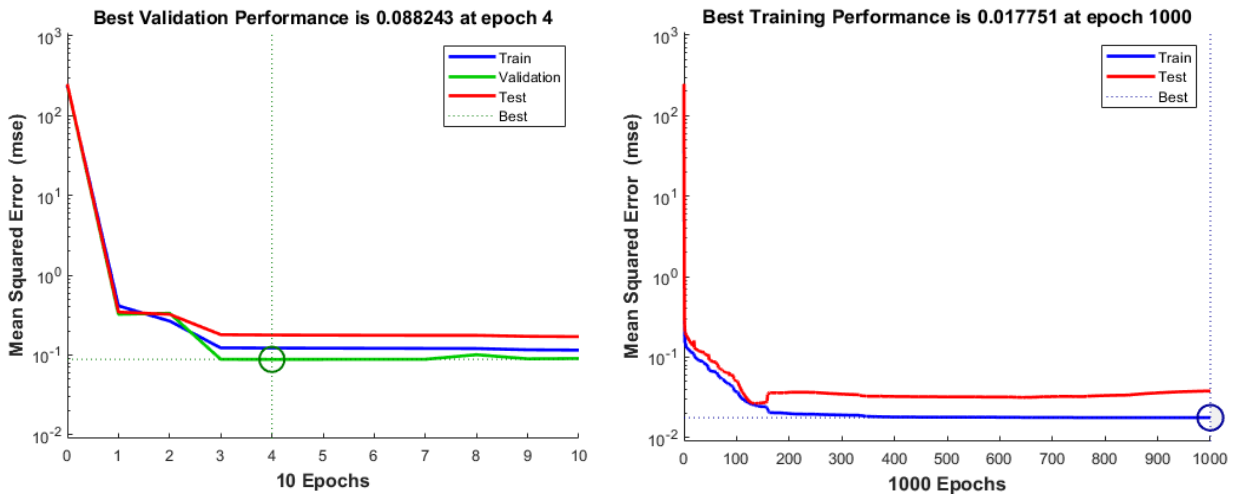
(b)



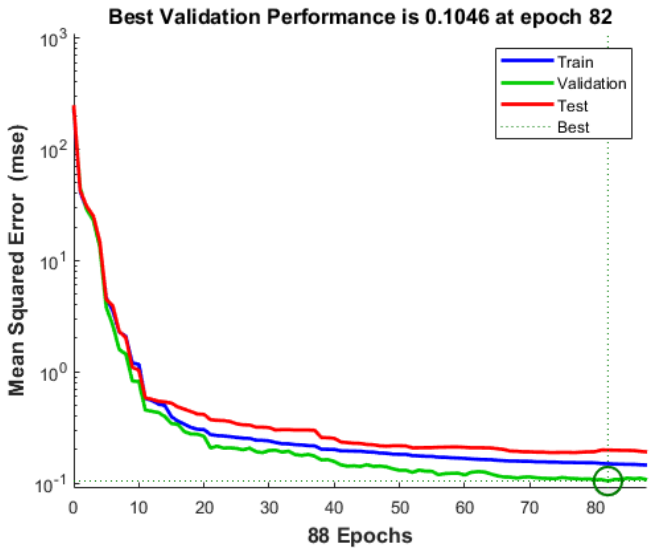
(c)

Figure 6-13. General performance of used algorithm a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

According to Figure 6-14, there is a very low mean square error between the distance of the detected object using the LiDAR sensor and those predicted by ANNs ($< 10^{-1}$). In addition, the results were converged faster in Levenberg-Marquardt and the almost constant high accuracy prediction was stable after 4 iterations which the stable prediction was obtained at 1000 and 81 iterations for Bayesian regularization and Scaled Conjugate, correspondingly. Additionally, the error between the train, validation and test results was insignificant that show the high accuracy and efficiency of the proposed ANN algorithms in the current thesis. Besides, the mean square error for all steps, train and test, was low, between 4×10^{-2} and 3×10^{-1}. In addition, the output results for the training layer for used ANNs are presented in Figure 6-15.



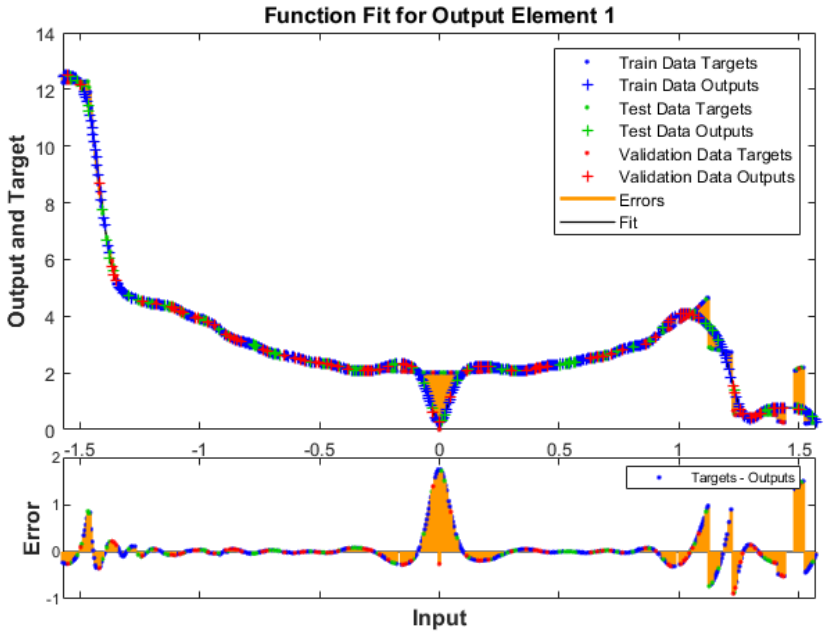
(a) (b)



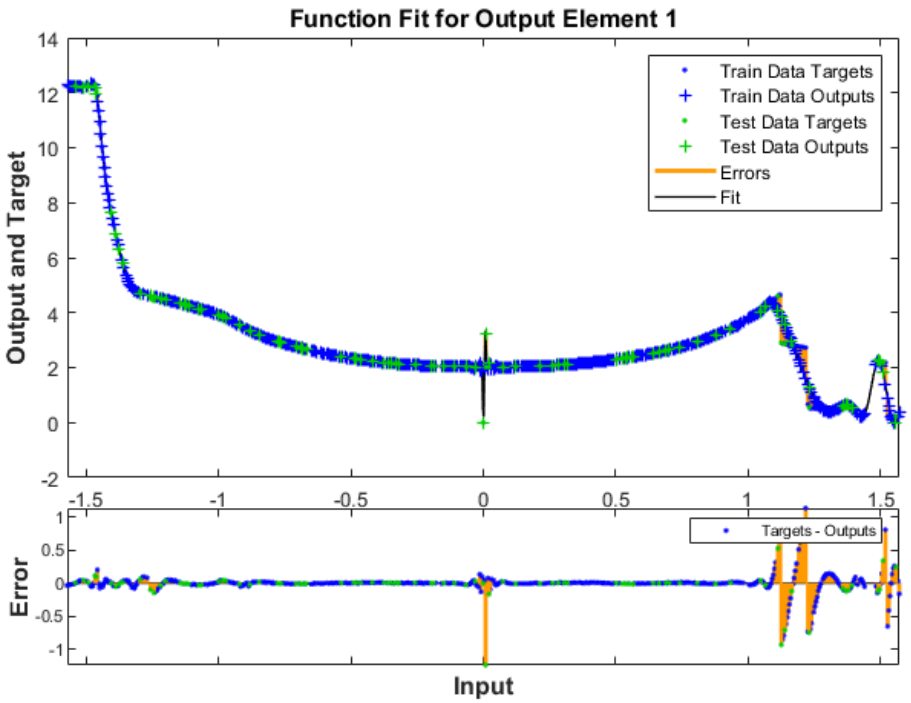
(c)

Figure 6-14. Mean square error in the training step a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

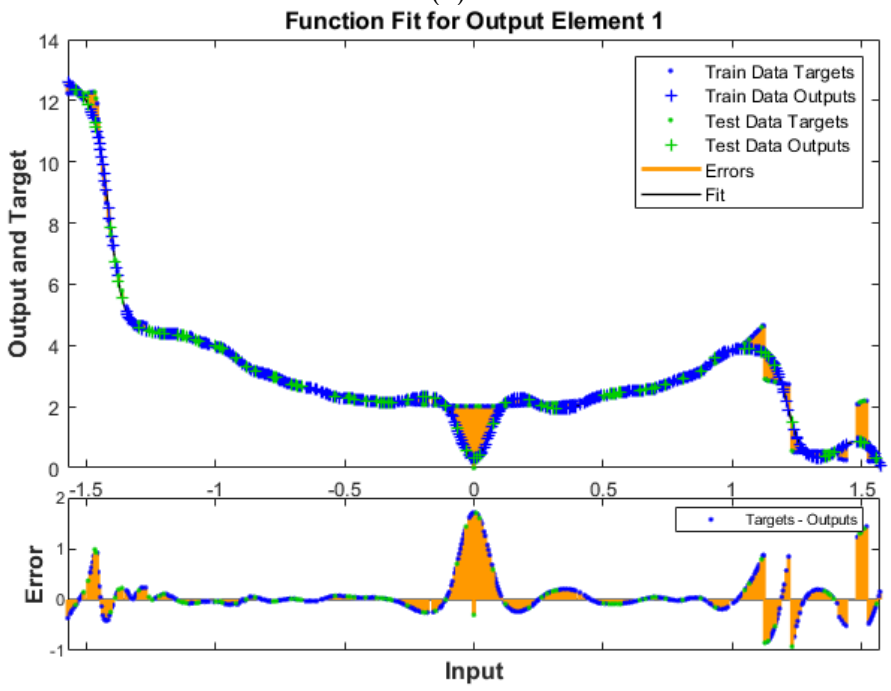
According to Figure 6-15, the results of predicted and exact values were presented with their error. Regarding these figures, the predicted values were the same as those exact values obtained by the LiDAR sensor in terms of angle and distance. In addition, in all processes, training, test and validation, all algorithms played an effective role to predict the object distance with the highest error of 2%. In this among, the minimum error was achieved when the Bayesian regularization technique was employed. This could be attributed to the high number of iterations performed by this network. Consequently, the Histogram and distribution of error in all three layers are presented in Figure 6-16.



(a)



(b)



(c)

Figure 6-15. Validation and error of trained algorithm a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

As shown in Figure 6-16, the distribution error for Levenberg-Marquardt and Scaled Conjugate was concentrated to the left side of the Histogram graph which shows the error in these networks is slightly greater than those obtained by Bayesian regularization. Therefore, generally, by considering the performance of networks for training and validation parts, Bayesian

regularization was the lowest error and higher efficiency for predicting the distance of objects from the sensor location with error ranging from -0.09297 to 0.06467.

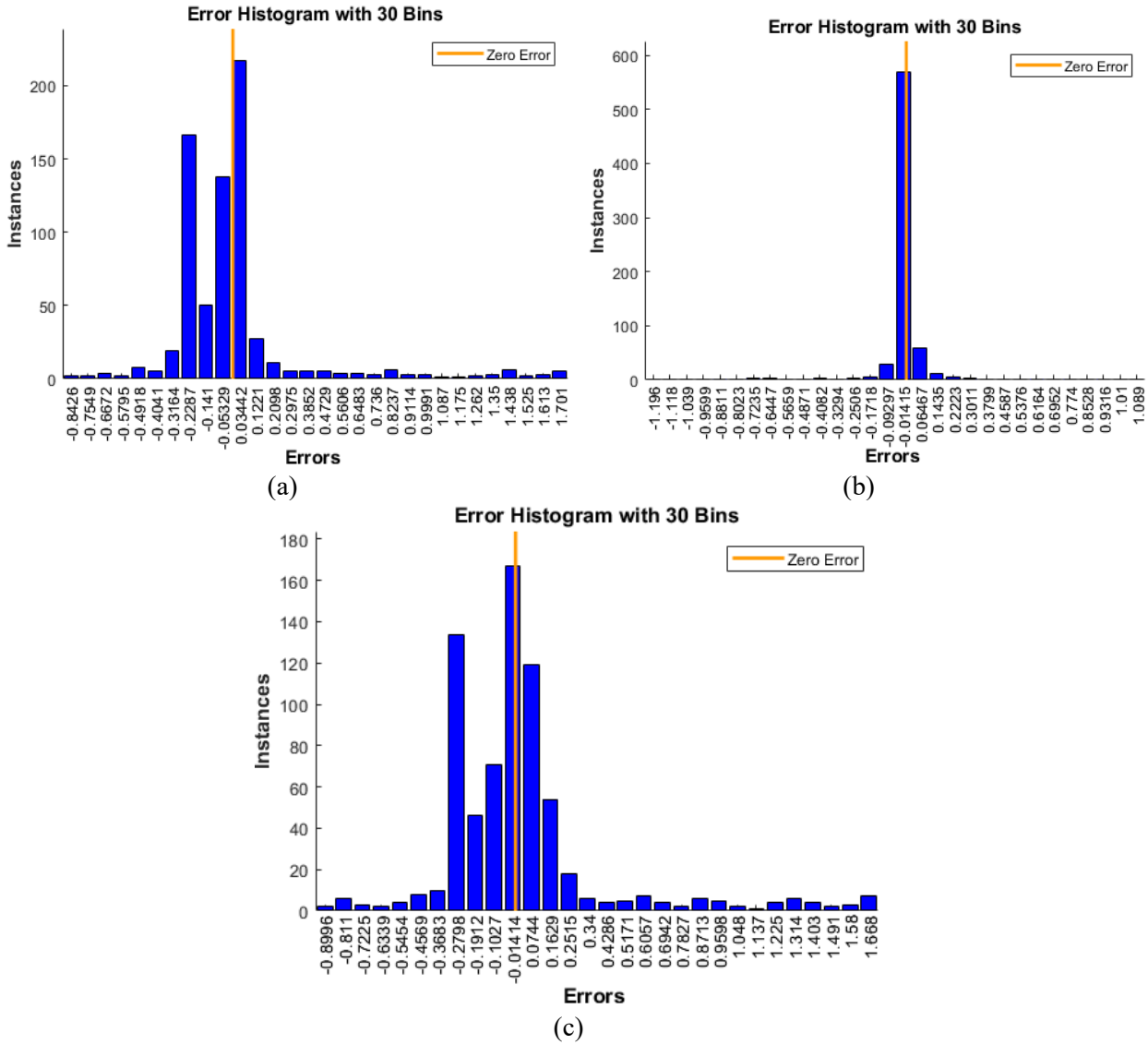
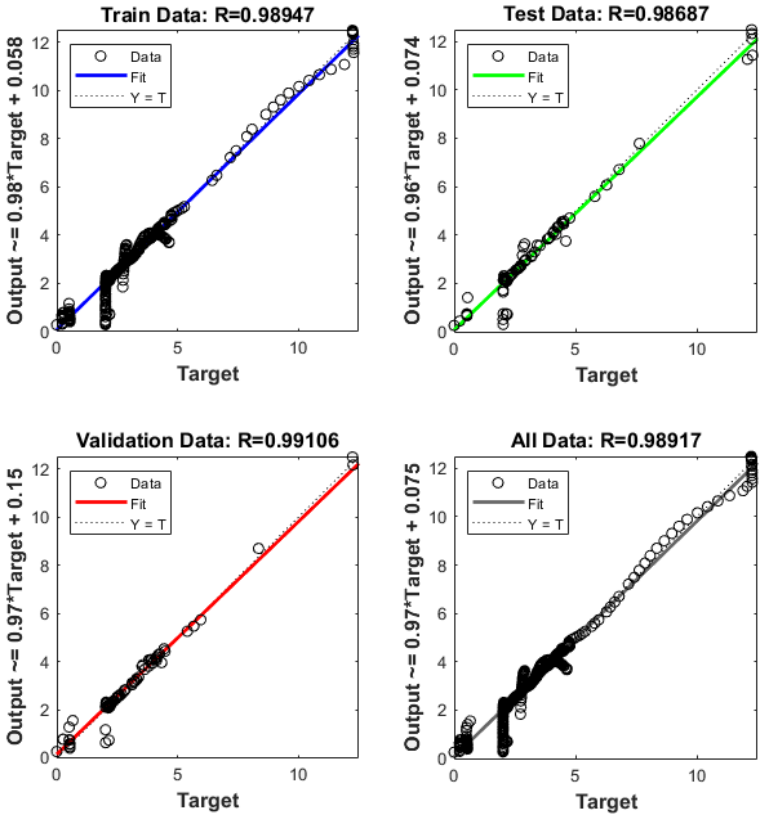


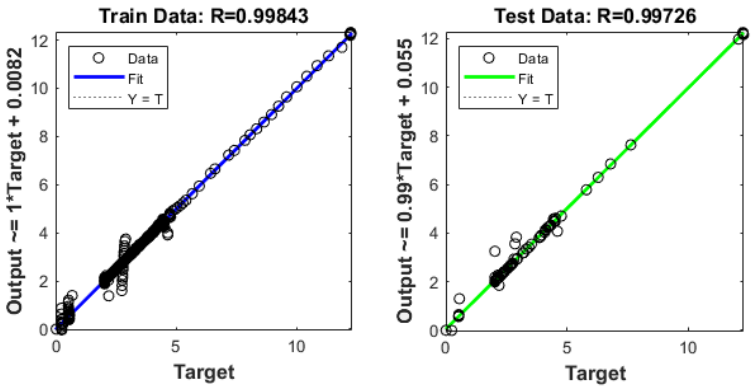
Figure 6-16. Histogram graph of the trained algorithm a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

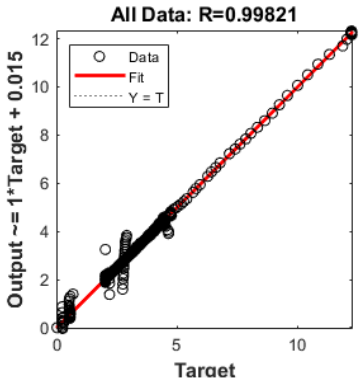
In this step, the regression was developed between the predicted and actual values to find the R-value. The closer the R-value is to one, the higher the accuracy of the algorithm in predicting the distance of objects from the sensor. It should be noted that the Levenberg-Marquardt is a new and complex algorithm developed by MATLAB. Therefore, this network can measure the accuracy of the prediction values for three steps of training, test and validation by regression, while Bayesian regularization and Scaled Conjugate are only able to develop a regression between predicted and exact values for two training and testing steps. According to the obtained results, the R-value was obtained by 0.98974, 0.98687, 0.99106 and 0.98917 for training, testing, validation and general performance steps when the Levenberg-Marquardt ANN algorithm was employed

which shows the accuracy and efficiency of this network for object detection. Additionally, using the Bayesian regularization ANN, the R-value for steps of training, testing and all results performance phases were obtained by 0.99843, 0.99726 and 0.99821, respectively. Therefore, by comparison between Levenberg-Marquardt and Bayesian regularization networks, it was found that the performance of Bayesian regularization for object detection prediction was superior. Conversely, when Scaled Conjugate was utilized, the R-value for training, testing and all results performance steps were obtained correspondingly by 0.9715, 0.98554 and 0.9706 which also indicates the accurateness and productivity of this network. Up to now, the concentration of the presented results was on the accuracy of training performance to find an algorithm for prediction. Right now, the used algorithm predicts the new values based on their trained algorithm and the obtained results are compared with the new data set to show the ability of used networks for future predictions. Therefore, the error of prediction and a new set of testing results (20% of the rest of the whole results) are presented in Figure 6-17.

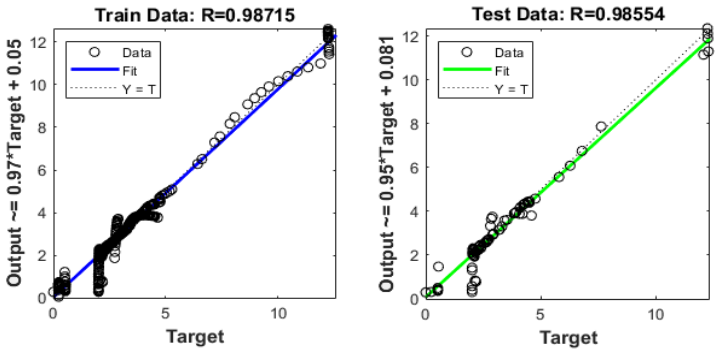


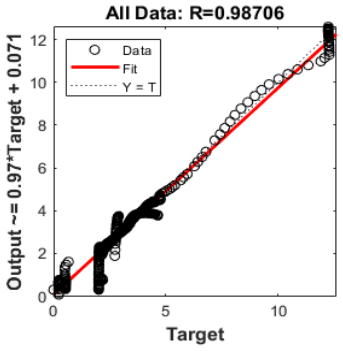
(a)





(b)





(c)

Figure 6-17. Accuracy and R-value of each algorithm a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

According to Figure 6-18, there is no significant difference between the error distribution of three used ANNs for the new data set testing. However, the higher concentration for Bayesian regularization ranging from -0.02926 to 0.06713 shows the performance of this network still is superior for the future predictions while the maximum error distribution was ranging from -0.1337 to 0.3299 for Levenberg-Marquardt and ranging from -0.1787 to 0.284 for Scaled Conjugate. After finding the distribution of the error, the fitting curve of a new set of testing results is presented in Figure 6-19.

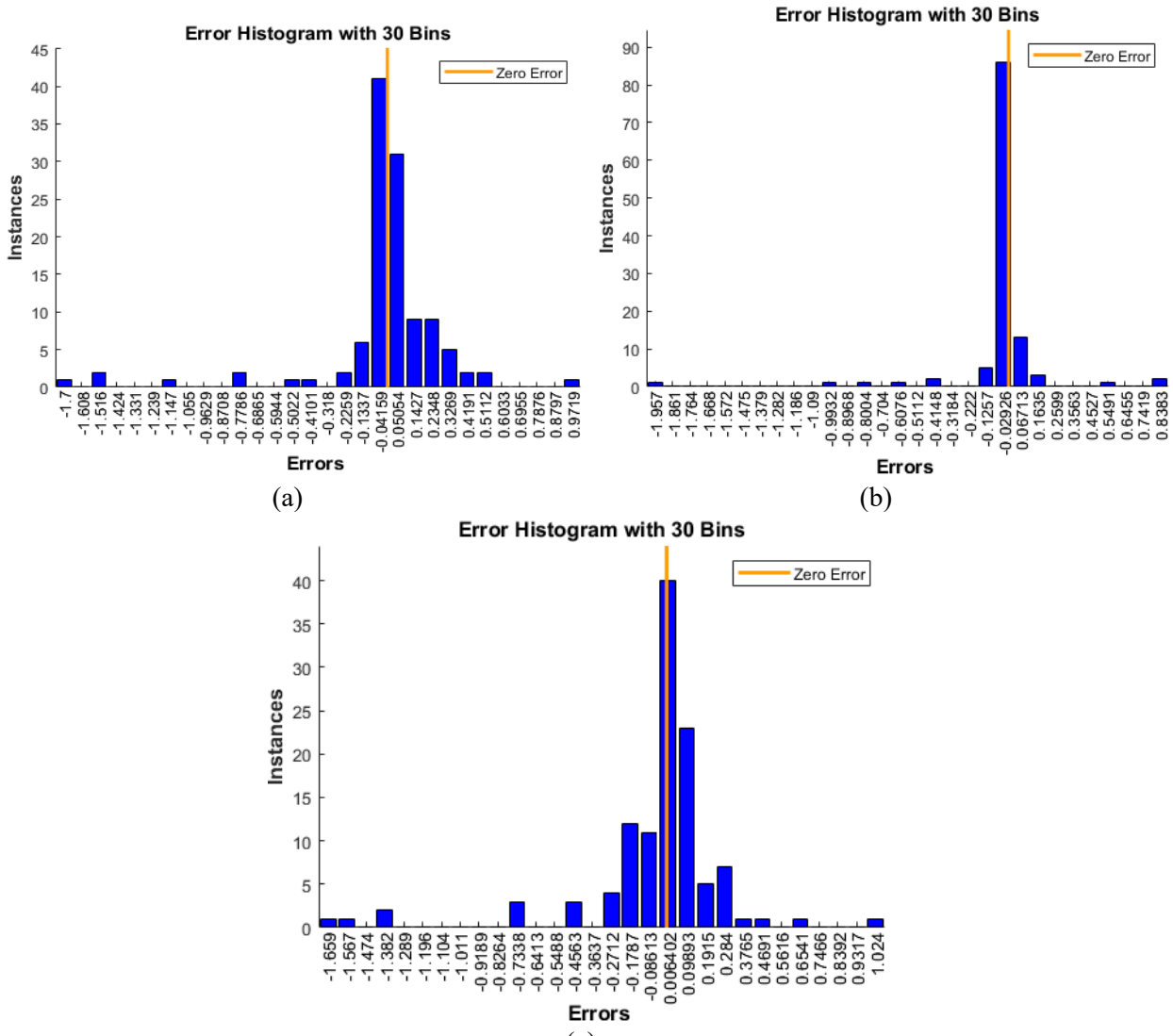
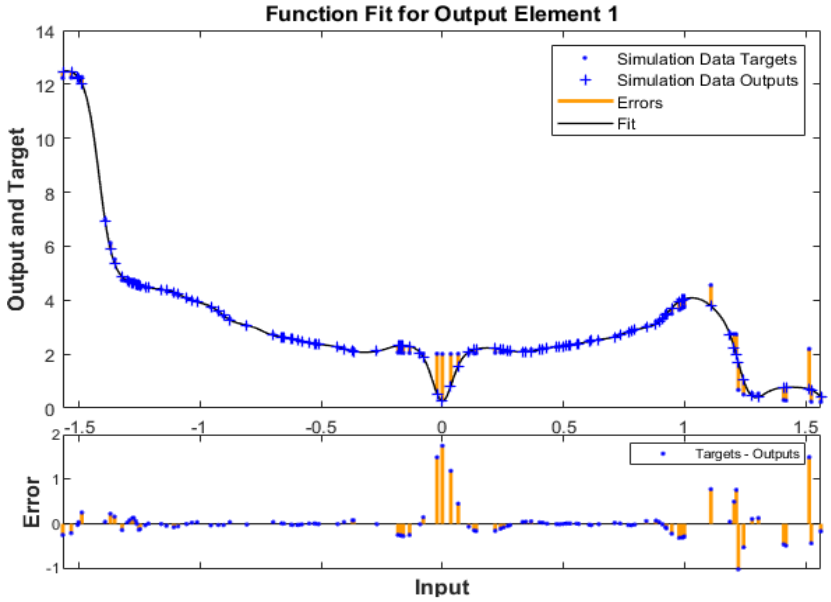


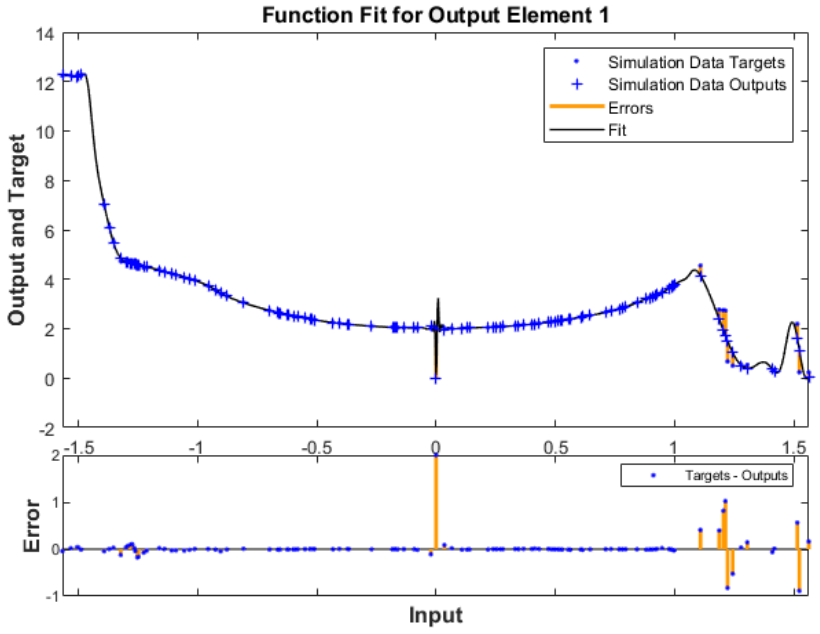
Figure 6-18. Histogram graph of the trained algorithm for the prediction a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

As seen from Figure 6-19, the results of predicted and exact values for the new testing set of data were presented with their error. Regarding these figures, the predicted values were the same as those exact values obtained by the LiDAR sensor in terms of angle and distance for the new set of results. In addition, in the testing step, all algorithms played an effective role to predict the object distance with the highest error of 2%. In this among, the minimum error was achieved when

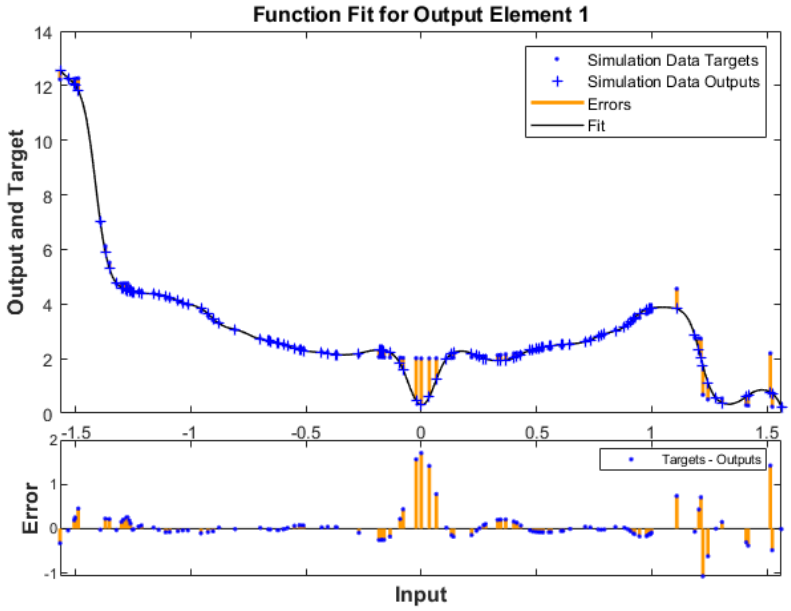
the Bayesian regularization technique was employed. This could be attributed to a high number of iterations performed by this network. Finally, a regression curve fitting was developed between the predicted and detected results for a new set of data to find an R-value. The results are presented in Figure 6-20.



(a)



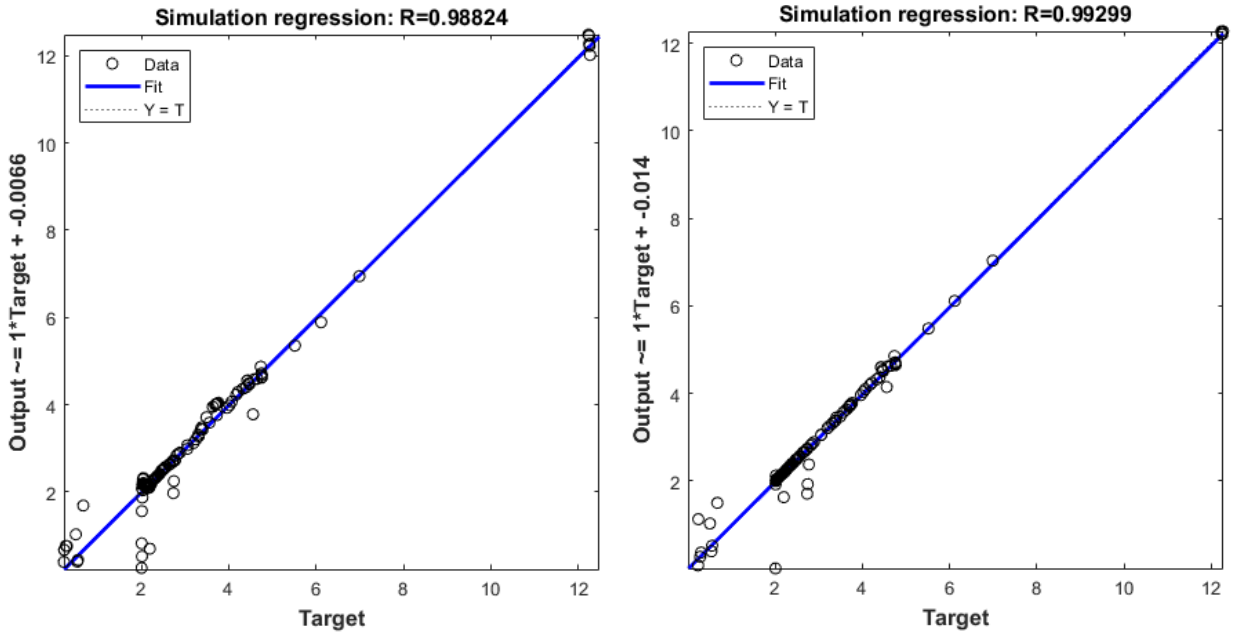
(b)



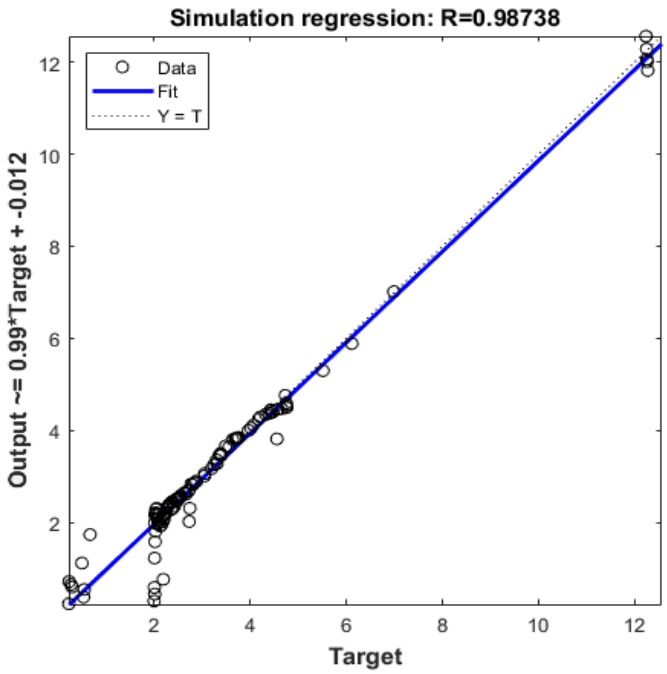
(c)

Figure 6-19. Prediction value and error of predicted a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

According to Figure 6-20, the obtained R-value after a curve fitting between the new set of detected results and those predicted by ANNs based on their trained algorithm was 0.98824, 0.99299 and 0.98738 for Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate ANNs. These results denote that the used algorithms with high accuracy predict the distance of objects from a camera with no need to use of LiDAR sensor. In this among, Bayesian regularization showed the highest efficiency and accuracy.



(a) (b)



(c)

Figure 6-20. Correlation between the predicted and exact values using a) Levenberg-Marquardt, b) Bayesian regularization and c) Scaled Conjugate

CHAPTER 7

CONCLUSION & SUGGESTION

7.1. Conclusion

In this study, the structure, use and performance of the LiDAR sensor have been identified. Then, to identify the accuracy of this sensor, the sensor is installed on a ligament and then the position of objects with different brightness percentages and different sizes are identified. Then, the accuracy of the LiDAR sensor was evaluated by measuring the exact distance of objects from the sensor location. Thus, by programming at the top, the data collected by the sensor is prepared. In the next step, by accurately measuring the detected distances, the accuracy of the sensor is determined. Consequently, by collecting all the data, mapping was performed for three different cases using the SLAM package in ROS and objects location. Finally, the artificial neural network (ANN) was used to propose a new algorithm for object detection and mapping instead of using LiDAR in future studies which helps to reduce costs and time-consuming. According to obtained results, the following conclusion could be drawn:

1- The accuracy of the used sensor was assessed by comparing the exact distances obtained with the use of a tape measure tool. The results showed that the LiDAR sensor, UTM-30LX model, could be accurately utilized for object detection with a low error of 2% in comparison with the exact distance value between the location of the object and sensor. To show the accuracy obtained results, the obtained results of the LiDAR sensor was compared with the exact distance. In this, the modified SSD algorithm in ROS that was used in the current thesis played a crucial role to detect the angle and coordinate of an object from the coordinate of the LiDAR sensor. The obtained low error confirm the accuracy of the SSD algorithm for object detection.

2- SLAM packages were used for mapping in this thesis. The comparison between the obtained maps and the cases study, and also the exact location of the objects and those provided on the maps showed that the used SLAM packages in ROS are a very useful tool for mapping. Therefore, the exact shape of maps including the corners, curve shapes and broken surfaces could be found. In addition, the coordinates and location of objects could be found on maps. This allows finding the location of objects around the car as well as the routs and space;

3- One of the main aims of this study was to develop a prediction algorithm to find a chance to replace the costly LiDAR sensor with a simple camera. To develop an algorithm, the collected data from the LiDAR sensor was employed. Therefore, the proposed work increases the range of the artificial neural network from a single item to a full range of objects detected at a different distance from the vehicle so that the vehicle may plan its route accordingly. The artificial neural network is applied in an interaction-enhancing multi-vehicle environment. More complicated items than artificial neural networks are detected in the algorithm as well. Our effort has enlarged the scope of the artificial neural network. Much has to be done though to ensure that the UGV is ready to be used in reality. The detection of obstacles can be increased such that the UGV can additionally detect and take action on any obstacles right before them. The UGV can also be done to scan its environment to recognize any obstacles that may arise during its selected route;

4- The low error obtained with the prediction artificial neural networks in comparison with those values achieved using a LiDAR sensor showed that the artificial neural network techniques are very useful tools for object detection by prediction instead of using LiDAR sensors which is costly equipment. Therefore, using ANNs, a simple camera could be replaced by a LiDAR sensor;
5- In the current thesis, three ANN techniques were utilized, Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate. According to the obtained results, Levenberg-Marquardt showed a fast-tracking performance in object detection. There, Levenberg-Marquardt, Bayesian regularization and Scaled Conjugate detected the distance of objects using 10, 1000 and 88 iteration which show the high performance of Levenberg-Marquardt as a fact object detection, relative to two other algorithms. Therefore, the Levenberg-Marquardt network can detect the distance and angle of objects around a car very fast. Therefore, the results were converged faster in Levenberg-Marquardt and the almost constant high accuracy prediction was stable after 4 iterations which the stable predicted was obtained at 1000 and 81 iterations for Bayesian regularization and Scaled Conjugate, respectively with the error of 1×10^{-1}, 4×10^{-2} and 3×10^{-1};
6- Conversely, by comparison between Levenberg-Marquardt and Bayesian regularization networks, it was found that the performance of Bayesian regularization for object detection prediction was superior. Conversely, when Scaled Conjugate was utilized, the R-value for training, testing and all results performance steps were obtained correspondingly by 0.9715, 0.98554 and 0.9706 which also indicates the accurateness and productivity of this network. Therefore, the accuracy of the Bayesian regularization network in terms of object detection was better than Levenberg-Marquardt and Scaled Conjugate;

7.2. Suggestion for future studies

According to the obtained results and performed evaluations in this thesis, the following suggestions are recommended for further work in the same area by future investigations:

1- A simple camera could be connected to the proposed ANN algorithms instead of using a LiDAR sensor. After the connection to the simple camera, the accuracy of the proposed networks should be evaluated;
2- It is recommended to perform the same procedure to find the accuracy of the used methodology in this thesis when a simple camera and LiDAR sensor are mounted o the rover;
3- It is also recommended to test other mapping and ANN techniques to find the highest accurate and efficient way for object detection during a low reaction time

Appendix I

```
# Single scan from a planar laser range-finder

        Header
        # stamp: The acquisition time of the first ray in the scan.
        # frame_id: The laser is assumed to spin around the positive Z axis
        # (counterclockwise, if Z is up) with the zero angle forward along the x axis

        float32 angle_min # start angle of the scan [rad]
        float32 angle_max # end angle of the scan [rad]
        float32 angle_increment # angular distance between measurements [rad]

        float32 time_increment # time between measurements [seconds] - if your scanner
        # is moving, this will be used in interpolating position of 3d points
        float32 scan_time # time between scans [seconds]

        float32 range_min # minimum range value [m]
        float32 range_max # maximum range value [m]

        float32[] ranges # range data [m] (Note: values < range_min or > range_max should
        be discarded)
        float32[] intensities # intensity data [device-specific units]. If your
# device does not provide intensities, please leave the array empty.
```

Appendix II

header:
seq: 8933
stamp:
secs: 1623766199
nsecs: 464529877
frame_id: “laser”
angle_min: -1.57079637051
angle_max: 1.57079637051
angle_increment: 0.0043632309619
time_increment: 1.73611151695e-05
scan_time: 0.0250000003725
range_min: 0.0230000000447
range_max:60
ranges: [12.230999946594238,
12.230999946586,12.236000061035156,12.2419996261596,12.253000259399414,
12.253999710083008]

In which, the basic extracted information is:

Angle_min : -1.57
Angle_max : 1.57
Angle_increment: 0.0043
Range_min: 0.02
Range max : 60.
Ranges: array of ranges starting from angle_min to angle_max, with a step of angle_increment

Appendix III

Header	# header for time/frame information
string ns	# Namespace to place this object in... used in conjunction with id to create a unique name for the object
int32 id	# object ID useful in conjunction with the namespace for manipulating and deleting the object later
int32 type	# Type of object
int32 action	# 0 add/modify an object, 1 (deprecated), 2 deletes an object, 3 deletes all objects
geometry_msgs/Pose pose	# Pose of the object
geometry_msgs/Vector3 scale	# Scale of the object 1,1,1 means default (usually 1 meter square)
std_msgs/ColorRGBA color	# Color [0.0-1.0]
duration lifetime	# How long the object should last before being automatically deleted. 0 means forever
bool frame_locked	# If this marker should be frame-locked, i.e. retransformed into its frame every timestep
geometry_msgs/Point[] points	
std_msgs/ColorRGBA[] colors	
string text	# Text displayed in rvis

The set of cluster information is saved in a list and published as Visualisation_msgs/markerArray under the /obstacle_list topic

Appendix IV

```
<launch>
  <param name="use_sim_time" value="true"/>
  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <param name="map_update_interval" value="5.0"/>

    <param name="maxUrange" value="16.0"/>
    <param name="sigma" value="0.05"/>
    <param name="odom_frame" value="laser"/>
    <param name="base_frame" value="laser"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="srr" value="0.1"/>
    <param name="srt" value="0.2"/>
    <param name="str" value="0.1"/>
    <param name="stt" value="0.2"/>
    <param name="linearUpdate" value="1.0"/>
    <param name="angularUpdate" value="0.5"/>
    <param name="temporalUpdate" value="3.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="30"/>
    <param name="xmin" value="-50.0"/>
    <param name="ymin" value="-50.0"/>
    <param name="xmax" value="50.0"/>
    <param name="ymax" value="50.0"/>
    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
  </node>
</launch>
```

Appendix V

```
clc
clear all
close all

% CHOOSE which training you'd like to use
% Enter 1 for Levenberg-Marquardt, 2 for Bayesian regularization, 3 for Scaled Conjugate
Gradient
val  = 1 ;

% Data Loading
load data
x= data(:,1)';
t= data(:,2)';

size(x)
size(t)

setdemorandstream(10008)

% Now divide data randomly
net.divideParam.trainRatio = 60/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 20/100;

% Create a Fitting Neural Network
switch val
   case 1

   %PART 1, Levenberg Marquardt

trainFcn = 'trainlm';
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize, trainFcn);

view(net)

%Now, Train the Network
[net,tr] = train(net,x,t);
nntraintool

figure(1)
plotperform(tr)
```

```
y = net(x);
e = t-y;

trainX = x(:,tr.trainInd);
testX = x(:,tr.testInd);
validX = x(:,tr.valInd);

trainT = t(:,tr.trainInd);
testT = t(:,tr.testInd);
validT = t(:,tr.valInd);

netTrain = net(trainX);
netTest = net(testX);
netValid = net(validX);
netAll = net(x);

figure(2)
plotfit(net,trainX,trainT,'Train Data',testX,testT,'Test Data',validX,validT,'Validation Data')

figure(3)
ploterrhist(e,'bins',30)

figure(4)
plotregression(trainT,netTrain,'Train Data',testT,netTest,'Test
Data',validT,netValid,'Validation Data',t,netAll,'All Data')

   case 2

   %PART 2, Bayesian Regulation

trainFcn = 'trainbr';
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize, trainFcn);

%Now, Train the Network
[net,tr] = train(net,x,t);
nntraintool

figure(5)
plotperform(tr)

y = net(x);
e = t-y;

trainX = x(:,tr.trainInd);
testX = x(:,tr.testInd);
```

```
trainT = t(:,tr.trainInd);
testT = t(:,tr.testInd);

netTrain = net(trainX);
netTest = net(testX);

netAll = net(x);

figure(6)
plotfit(net,trainX,trainT,'Train Data',testX,testT,'Test Data')

figure(7)
ploterrhist(e,'bins',30)

figure(8)
plotregression(trainT,netTrain,'Train Data',testT,netTest,'Test Data',t,netAll,'All Data')

   case 3

   %PART 3, Scaled Conjugate Gradient
trainFcn = 'trainscg';
hiddenLayerSize = 20;
net = fitnet(hiddenLayerSize, trainFcn);

%Now, Train the Network
[net,tr] = train(net,x,t);
nntraintool

figure(9)
plotperform(tr)

y = net(x);
e = t-y;

trainX = x(:,tr.trainInd);
testX = x(:,tr.testInd);

trainT = t(:,tr.trainInd);
testT = t(:,tr.testInd);

netTrain = net(trainX);
netTest = net(testX);

netAll = net(x);
```

```
figure(10)
plotfit(net,trainX,trainT,'Train Data',testX,testT,'Test Data')

figure(11)
ploterrhist(e,'bins',30)

figure(12)
plotregression(trainT,netTrain,'Train Data',testT,netTest,'Test Data',t,netAll,'All Data')

end

%Simulation
%Uncomment below and provide your arbitrary data between the brackets

load datasim;
xtry=datasim(:,1)';
ytry=datasim(:,2)';
ynet=net(xtry);
esim=ynet-ytry;
figure(13)
ploterrhist(esim,'bins',30)
figure(14)
plotfit(net,xtry,ytry,'Simulation Data')
figure(15)
plotregression(ytry,ynet,'Simulation regression')
```

References

[1] A. A. Pourezzat, A. A. Sadabadi, A. Khalouei, G. T. Attar, and R. Kalhorian, “Object Tracking: A Survey,” *Adv. Environ. Biol.*, vol. 7, no. 2, pp. 253–259, 2013.

[2] D. Prajapati and H. J. Galiyawala, “A Review on Moving Object Detection and Tracking,” *Int. J. Comput. Appl.*, vol. 5, no. 3, pp. 168–175, 2015.

[3] K. Schindler, A. Ess, B. Leibe, and L. Van Gool, “Automatic detection and tracking of pedestrians from a moving stereo rig,” *ISPRS J. Photogramm. Remote Sens.*, vol. 65, no. 6, pp. 523–537, 2010.

[4] Y. Cao, D. Guan, Y. Wu, J. Yang, Y. Cao, and M. Y. Yang, “Box-level segmentation supervised deep neural networks for accurate and real-time multispectral pedestrian detection,” *ISPRS J. Photogramm. Remote Sens.*, vol. 150, pp. 70–79, 2019.

[5] H. Luo *et al.*, “Vehicle Detection in High-Resolution Aerial Images via Sparse Representation and Superpixels,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 1, pp. 103–116, 2015.

[6] J. M. Kim, Chanho; Li, Fuxin; Ciptadi, Arridhana; Rehg, “Multiple Hypothesis Tracking Revisisted,” *Int. Conf. Comput. Vis.*, 2015.

[7] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 3354–3361.

[8] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, “MOT16: A benchmark for multi-object tracking,” *arXiv Prepr. arXiv1603.00831*, 2016.

[9] L. Patino, T. Cane, A. Vallee, and J. Ferryman, “Pets 2016: Dataset and challenge,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 1–8.

[10] L. Spinello, R. Triebel, and R. Siegwart, “Multimodal People Detection and Tracking in Crowded Scenes.,” in *AAAI*, 2008, pp. 1409–1414.

[11] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, “Generative Object Detection and Tracking in 3D Range Data - Kaestner et al. - Unknown.pdf,” 2012.

[12] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, “Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1710–1715, 2008.

[13] K. O. Arras, Ó. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," *Proc. - IEEE Int. Conf. Robot. Autom.*, no. April, pp. 3402– 3407, 2007.

[14] W. Xiao, B. Vallet, K. Schindler, and N. Paparoditis, "Simultaneous detection and tracking of pedestrian from panoramic laser scanning data," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 3, no. July, pp. 295–302, 2016.

[15] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for human classification in 3D LiDAR-based tracking," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2017–Septe, pp. 864– 871, 2017.

[16] W. Luo *et al.*, "Multiple Object Tracking: A Literature Review," pp. 1–18, 2014.

[17] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury, "A stochastic graph evolution framework for robust multi-target tracking," in *European Conference on Computer Vision*, 2010, pp. 605–619.

[18] M. Yang, T. Yu, and Y. Wu, "Game-theoretic multiple target tracking," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1–8.

[19] Z. Wang, "Laser-based detection and tracking of dynamic objects." University of Oxford, 2014.

[20] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3D LiDAR scans," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2016–June, pp. 4508–4513, 2016.

[21] J. Cui, H. Zha, H. Zhao, and R. Shibasaki, "Laser-based detection and tracking of multiple people in crowds," *Comput. Vis. Image Underst.*, vol. 106, no. 2–3, pp. 300–312, 2007.

[22] O. Brock and F. Trinkle, J. Ramos, "Model-Based Vehicle Tracking for Autonomous Driving in Urban Environments," *MIT Press*, vol. 1, pp. 175–182, 2009.

[23] H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle," *Rob. Auton. Syst.*, vol. 88, pp. 71–78, 2017.

[24] S. Yang and C. Wang, "Simultaneous ego-motion estimation, segmentation, and moving object detection," *J. F. Robot.*, vol. 28, no. 4, pp. 565–588, 2011.

[25] J. Schauer and A. Nüchter, "Removing non-static objects from 3D laser scan data," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, no. October 2017, pp. 15–38, 2018.

[26] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Comput. Vis. Image Underst.*, vol. 125, pp. 251–264, 2014.

[27] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 1146–1152, 2013.

[28] Y. Yu, J. Li, H. Guan, and C. Wang, "Automated Detection of Three-Dimensional Cars in Mobile Laser Scanning Point Clouds Using DBM-Hough-Forests," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 7, pp. 4130–4142, 2016.

[29] H. Yokoyama, H. Date, S. Kanai, and H. Takeda, "Detection and classification of polelike objects from mobile laser scanning data of urban environments," *Int. J. Cad/Cam*, vol. 13, no. 2, pp. 31–40, 2013.

[30] J. Anderson, R. Massaro, J. Curry, R. Reibel, J. Nelson, and J. Edwards, "LADAR: Frequency-Modulated, Continuous Wave Laser Detection And Ranging," *Photogramm. Eng. Remote Sens.*, vol. 83, no. 11, pp. 721–727, 2017.

[31] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[32] N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imaging*, vol. 16, no. 4, p. 49901, 2007.

[33] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[34] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, vol. 1, no. 14, pp. 281–297.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, 1996, vol. 96, no. 34, pp. 226–231.

[36] A. Sampath and J. Shan, "Segmentation and reconstruction of polyhedral building roofs from aerial lidar point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 3, pp. 1554–1567, 2010.

[37] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[38] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[39] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, Springer, 2005, pp. 321–352.

[40] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," in *ACM Sigmod Record*, 1998, vol. 27, no. 2, pp. 73–84.

[41] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, 1996, vol. 25, no. 2, pp. 103–114.

[42] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *VLDB*, 1997, vol. 97, pp. 186–195.

[43] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *ACM Sigmod record*, 1999, vol. 28, no. 2, pp. 49–60.

[44] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, 1998, vol. 98, pp. 58–65.

[45] P. Konstantinova, A. Udvarev, and T. Semerdjiev, "A study of a target tracking algorithm using the global nearest neighbour approach," in *Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech'03)*, 2003, pp. 290–295.

[46] S. Blackman and R. Popoli, "Design and Analysis of Modern Tracking Systems (Artech House Radar Library)," *Artech House*, 1999.

[47] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," *IEEE J. Ocean. Eng.*, vol. 8, no. 3, pp. 173–184, 1983.

[48] A. Amditis, G. Thomaidis, P. Maroudis, P. Lytrivis, and G. Karaseitanidis, "Multiple Hypothesis Tracking Implementation," *Laser Scanner Technol.*, 2012.

[49] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. Automat. Contr.*, vol. 24, no. 6, pp. 843–854, 1979.

[50] R. Richmond and S. Cain, "Direct-detection LADAR systems," 2010.

[51] J. Shan and C. K. Toth, *Topographic laser ranging and scanning: principles and processing*. CRC Press, 2018.

[52] Z. W. Barber, W. R. Babbitt, B. Kaylor, R. R. Reibel, and P. A. Roos, "Accuracy of active chirp linearization for broadband frequency modulated continuous wave radar," *Appl. Opt.*, vol. 49, no. 2, pp. 213–219, 2010.

[53] M. J. Halmos and D. M. Henderson, "Pulse compression of an FM chirped CO 2 laser," *Appl. Opt.*, vol. 28, no. 17, pp. 3595–3602, 1989.