Navigation of Autonomous Vehicle in Indoor Environment

HAKAN YILMAZ
B.S., Computer Engineering, Işık University, 2008

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY
2011

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

NAVIGATION OF AUTONOMOUS VEHICLE IN INDOOR ENVIRONMENT

HAKAN YILMAZ

APPROVED BY:

Assist. Prof. Emine Ekin (Thesis Supervisor)	Işık University	_____
Prof. Yorgo İstefanopulos	Işık University	_____
Assist. Prof. F. Boray Tek	Işık University	_____

APPROVAL DATE: ..../..../....

Navigation of Autonomous Vehicle in Indoor Environment

Abstract

This study investigates transforming a remote controlled vehicle into an autonomous vehicle which explores the environment and is able to find its own location in this unknown static indoor environment. It also constructs the map of this environment simultaneously. The study consists of mainly three parts which are (i) the hardware of autonomous vehicle; (ii) while exploring the environment furnished with landmarks to detect the landmarks, and to control the vehicle so that it can move towards these landmarks and/or it can avoid them; (iii) once having the ability of moving in the environment safely, locating itself and constructing the map of environment. The hardware of the vehicle has mostly designed and installed by Robotics and Autonomous Vehicles Laboratory (RAVLAB) members as it is a part of one of the RAVLAB's projects[1]. Moving safely in an unknown environment is a requirement for the car to locate itself and to extract the map of the environment, which are known as localization and mapping problems respectively. While traveling, at every time step, the car estimates its new position by computing the displacement using its instantaneous speed value. The positions of landmarks in its visible area are also estimated. However, real positions of both the car and the landmarks are usually different then the estimated ones because of several factors including sensor noises, car's voltage regulations etc. We have performed a series of experiments in laboratory[2]. The first set of experiments focused on making the vehicle to navigate towards the closest object. Second set of experiments focused on making the vehicle to stop in the presence of an object in front of the car. Finally, simultaneous localization and mapping the environment of the car has been tested. The experiments have revealed encouraging results in the sense that the vehicle can traverse the running environment safely. Although, localization and mapping results of the vehicle were not free of error, we were able to get the expected results at the end.

[1]TOTAY:Tekerlekli Otonom Araba Yarışları. Supported by the Işık University Scientific Research Projects Fund, BAP...

[2]Robotic and Autonomous Vehicles Laboratory of Işık University

OTONOM ARACIN KAPALI ALANDA DOLAŞMASI

Özet

Bu çalışma, uzaktan kumandalı bir aracın, çevresini araştıran ve bu bilinmeyen kapalı çevrede kendi yerini tayin edebilen otonom bir araca dönüşümünü incelemektedir. Çalışma aynı zamanda bu çevrenin haritasını oluşturmaktadır. Çalışma, üç temel bölümden oluşmaktadır: (i) otonom aracın donanımı, (ii) işaretleri bulmak için işaretlerle donatılmış ortamı araştırırken, ve bu işaretlere doğru hareket etmesi ve/veya onlara çarpmaması için aracı kontrol etme, (iii) ortamda güvenli şekilde hareket kabiliyetini edindikten sonra, ortamdaki yerini tayin etmesi ve haritasını çıkarması. Otonom aracın tasarımı ve kurulumu Robotbilim ve Otonom Araçlar Laboratuarında (RAVLAB) ve RAVLAB çalışanları tarafından yapılmıştır. Bilinmeyen bir ortamda güvenli bir şekilde haraket etmek, sırasıyla yer bulma ve harita çıkarma problemleri olarak bilinen arabanın kendi yerini bulma ve ortamın haritasını çıkarma işlemidir. Otonom araç seyir halindeyken, bir sonraki pozisyonunu tahmin etmek için, anlık hızı ile geçen zamanın hesaplanması sonucunda aracın yeni yeri tahmin edilir. Aynı zamanda kameradan görülen işaretlerinde yeri tahmin edilir. Ama genellikle aracın gerçek pozisyonuyla, aracın tahmin pozisyonları farklı çıkar. Farklı çıkmasının birçok nedeni vardır örneğin ortamdaki gürültü, sensor gürültüsü, aracın gerilim regülasyonu, v.b.. Laboratuvarda birçok deney gerçekleştirdik. Birinci deney grubu, aracın en yakın objeye ilerlemesi üzerinde yoğunlaştı. Sonraki deney grubu, önünde bir obje bulunması durumunda aracın durmasına odaklandı. Son olarak, araç ortamının eş zamanlı olarak yer tayin edilip haritasının çıkarılması test edildi. Deneyler, aracın çalıştığımız ortam içinde güvenli bir şekilde her yöne haraket etmesi açısından cesaretlendirici sonuçlar verdi. Otonom aracın yer bulma ve harita çıkarma sonuçlarının hatasız olmamasına rağmen, sonunda beklediğimiz sonuçları elde edebildik.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my thesis advisor Assist. Prof. Dr. Emine Ekin without whose guidance and support I would never have been able to write this thesis. Her constant contribution, understanding and patience enabled me to complete this work successfully. She listened to my half-cooked ideas and helped me to develop them. I am indebted to her for all.

I would like to thank Assist. Prof. Dr. Boray Tek for sharing his time and knowledge with me. His constructive criticism and useful comments have been an invaluable contribution to this study. I also wish to thank Prof. Dr. Yorgo Istefanopulos for sharing his valuable time with me to improve this study and for his helpful comments.

Many thanks to my friends Koray Ak, Mehmet Güneş, Doğan Kırcalı, Îbrahim Îyidir, Pasindu Abeysundera and Mehmet Alkan for their friendship, support and confidence in me.

I am also grateful to one special person, Yasemin Kesen, whose patience and support have been a great contribution to this study. She listened to me patiently and cheered me up when I am down. Also she was always there when I got stuck.

Finally, I would like to express my deepest gratitude to my family for their endless love, support and belief. My parents, Erdem Yılmaz and Vicdan Yılmaz, my sisters Tuğba Yılmaz, and Seda Yılmaz were very supportive when I needed them. I am very lucky to have such a family. This thesis is dedicated to them.

Table of Contents

List of Tables

List of Figures

List of Symbols

B	Control matrix
F	State transition matrix
GB	Gigabayt
H	Measurement matrix
i, j	Pixel index
K	Kalman Gain
n	The index of landmark
MHz	Mega Hertz
MM	MilliMeter
P	State variance matrix
Q	Process variance matrix
R	Measurement variance matrix
s	State
t	Time
x, y	The coordinate of x,y in the plane
u	Controls of vehicle
V	Volt
v	Vector
z	Measurement range
θ	All Landmarks
β	Angle between two vectors
$\oplus$	Dilation Operation
$\ominus$	Erosion Operation
$\widehat{x}$	Estimated state
μ	Mean value of a block
σ	Standard deviation of a block
σ^2	Variance of a block

List of Abbreviations

A	**A**mper
AGVs	**A**utonomously **G**uided **V**ehicles
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
EEPROM	**E**lectrically **E**rasable **P**rogrammable **R**ead **O**nly **M**emory
EGNOS	**E**uropean **G**eostationary **N**avigation **O**verlay **S**ervice
EMI	**E**lectro **M**agnetic **I**nterference
GPS	**G**lobal **P**ositioning **S**ystem
HDMI	**H**igh **D**efinition **M**ultimedia**I**nterface
KF	**K**alman **F**ilter
KG	**K**alman **G**ain
LCC	**L**eadless **C**eramic **C**arrier
Li-Po	**Li**thium-ion **Po**lymer
ME	**M**otion **E**stimation
MHz	**M**ega **H**ertz
MSAS	**M**ulti-functional **S**atellite **A**ugmentation **S**ystem
Ni-Cd	**Ni**ckel **Cd**dmium
Ni-MH	**Ni**ckel **M**etal **H**ydride
OEM	**O**riginal **E**quipment **M**anufacturer
OS	**O**perating **S**ystem
PC	**P**ersonal **C**omputer
PCBs	**P**oly **C**hlorinated **B**iphenyls
PCI	**P**eripheral **C**omponent **I**nterconnect
PWM	**P**ulse **W**idth **M**odulation
RGB	**R**ed **G**reen **B**lue

SBAS	**S**atellite **B**ased **A**ugmentation **S**ystem
SD	**S**ecure **D**igital
SDMC	**S**ecure **D**igital **M**emory **C**ard
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SMAL	**S**imultaneous **M**apping **A**nd **L**ocalization
SRAM	**S**tatic **R**andom **A**ccess **M**emory
Thr	**Thr**eshold
TTFF	**T**ime **T**o **F**irst **F**ix
US	**U**nited **S**tates
WAAS	**W**ide **A**rea **A**ugmentation **S**ystem
WiFi	**Wi**reless **Fi**delity
3-D	**3** **D**imension

Chapter 1

Introduction

Autonomy is the ability of making decisions on your own to achieve a goal. An autonomous vehicle must make decisions to increase its expected gain to maximum level, so it needs to make rational decision. To achieve that, there has to be an agent who has a description of its world, a set of sensors to get information about its present condition, and a set of actuators to execute its decisions.

The description of the world is called an environment model which gives the agent a basis to make rational decisions. To decide on the best order of actions to take place, a rational agent may stimulate its environment model to do an analysis of 'what would happen if', or it may do a backward analysis from the target to the current state. There are two different ways for a rational agent to learn its environment: it may have a built-in model or it may explore its surroundings.

Autonomous vehicles are driverless, so they have to be rational agents in terms of sensing their environment, using their internal model of the environment so as to make rational decisions and using actuators (i.e. gas, brake, steering wheel, turn signals) to execute their decisions.

In this thesis, a small sized autonomous vehicle has been designed and constructed. What this vehicle is supposed to do is to locate itself and to construct the map of its running environment, which are known as robotic localization and mapping problems respectively.

1.1 Simultaneous Localization and Mapping (SLAM)

SLAM is known to be the set of algorithms to solve robotics localization and/or mapping problem. Localization means that the robot estimates its own position in the environment during its movement, where mapping means either constructing the map of an unknown environment or updating a known map.

Though the hardware of the robots might be vastly different, as well as the algorithms to solve both problems of localization and mapping; there are mainly two different data sets required: the odometry data which estimates the robots position using the instructions sent to robot's actuators, and distances to the visible objects. Then, basically, what SLAM techniques achieve can be given as:

Being in a position p, and knowing distances to each object d[i]

At each time step t:

1. Move robot
2. Estimate robot's and objects new positions using odometry data
3. Measure the distance to the objects
4. Using the real and estimated distances to the objects, update the robot's position

The last step in above algorithm is the heart of SLAM process, where mostly either Kalman Filter or Extended Kalman Filter [1, 2] is being employed.

The solutions offered for localization and mapping problem strongly depends on environment, i.e., whether or not the robot has priori knowledge about the environment; whether or not the positions of the objects are changing, appearing or disappearing in time.

We have worked in laboratory, that is indoor, environment which is furnished with a number of landmarks. Thus, the environment of the car is described as static

environment. As it is aimed to construct the map of the environment, no priori information has provided to the car, which means that the running environment for the car is unknown, static, and indoor.

1.2 Thesis Outline

The first part in this thesis investigates how autonomous car navigates towards a target object in indoor environment. After determining random start and end points for the vehicle, it has seen that it arrived at the end point by avoiding obstacles. Later, the vehicle has aimed to going to the nearest object. After a series of experiments, it has been observed that it succeeded in going to the nearest object.

In the second part of this thesis, simultaneous localization and mapping of the environment were done.

The applications have been developed on Windows environment with Visual Studio 2008, C++ with OpenCV computer vision library installed. The sensor data has been collected, and the instructions have been sent to the car via Arduino software. Furthermore, the connection between Arduino software and Visual C++ software is completed via Serial communication library. Matlab has been used in distance measurement task.

In Chapter 2, some background information was provided on autonomous vehicles. The background information comprises the concepts of the domain of intelligent vehicles, their history, and information on construction of an autonomous ground vehicle. The background information also includes the list of components that are used in autonomous vehicle.

Chapter 3 explains two subgoals of this thesis, avoiding the obstacles and moving to the closest object, which requires identifying the closest object first. Both objectives are achieved using image data taken via web cam, and distance data

measured with ultrasonic sensors. Thus, the chapter mainly contains image processing algorithms employed. According to the decision made, how the motor control is achieved has also been included in the chapter.

Chapter 4 presents SLAM algorithms. It further introduces distance measurement techniques which we have used to extract the positions of the landmarks . Then, this chapter provides a summary of the studies concerning Kalman Filtering.

This thesis finally discusses the construction of an autonomous vehicle and SLAM [3] being used for indoor navigation.

Chapter 2

Background Information

This chapter investigates previous studies on autonomous robot vehicle projects and materials that are used in this project.

2.1 Early Work on Autonomous Robot Vehicles

An intelligent autonomous vehicle should navigate by itself while sensing its environment and internal states. It is further expected to avoid obstacles by interpreting the information that comes from sensors. Guiding and controlling commands depending on spatial and temporal variations in the environment are final duties of this vehicle [4].

In the past, there have been many attempts for autonomous vehicles including the prominent ones EUREKA Prometheus Project [5] and the Defense Advanced Research Projects Agency (DARPA) Grand Challenge [6]. The EUREKA Prometheus Project is considered as the largest research development undertaken in the area of driverless cars so far. The DARPA Grand Challenge is also regarded as the most recent major advancement in the field of autonomous vehicles.

There are a number of research programs on autonomously guided vehicles (AGVs) those had been initiated for several years before 1992 [4].These research programs focused on sensor management, multi-sensor data fusion and machine perception

[7, 8], goal specification and planning [9, 10], navigation [11, 12], guidance and motion control [13–15], which also cover the programs being worked on nowadays.

Lorfield and Harris [16] argue that autonomous vehicles have the ability of decreasing the robot complexity, widening the market for subsystems by using control protocols, supplying portability, inter-operability, and modularity in software, minimizing time, cost, risk and initial investment, and yielding technology transfer between application areas. There are levels and sublevels of sensory acquisition and processing, world modeling, and task decomposition in the architecture of the vehicle.

World modeling, landmark recognition, and learning the landmarks are requirements needed in this thesis for navigation. In Harris and Channley's [4] study, two stages are argued for the path planning problem. These stages are developing a global strategy in terms of immediate goals and generating commands at a low level for the achievement of these goals. At the lowest level, which is the servo level, vehicle dynamics needs to be handled by the motion controller.

In recent years, path planning algorithms, which are responsible for maintaining many essential aspects of plausible agent behavior, including collision avoidance and goal satisfaction, are observed to be more efficient. The problem of path planning is studied for the case of a mobile robot moving in an environment filled with obstacles whose shape and positions are not known [17]. The A* algorithm is a path planing algorithm that uses Dijkstra algorithm [18] to obtain the optimum result for the robot. The disadvantage of the A* is that A* uses uniformed grids representation which must allocate large amounts of memory [19]. The D* algorithm is another path planing algorithm.The D* algorithm helps the robot to find the optimal path in a dynamic environment in which fixed obstacles in the environment are changed to moving obstacles.

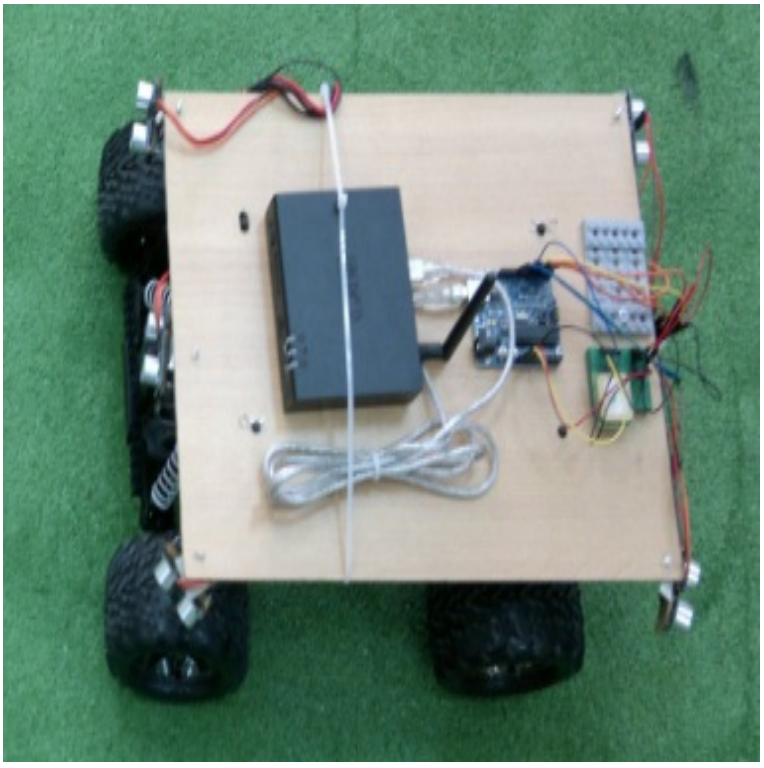

Figure 2.1: Autonomous Car

2.2 Designing an Autonomous Vehicle

An autonomous car is expected to navigate in the absence of an outside control by avoiding the obstacles in the environment. The autonomous car in Figure 2.1 is designed for this thesis with five sensors, one fit-PC, one camera, one arduino, one GPS device and one compass.

2.2.1 Fit-PC

Fit-PC is a computer that has 1.6GHz Atom Z530 processor, 1GB ram, 160GB hard disk, HDMI output which provides 1920x1080 resolution support. Both Gigabit Ethernet and 802.11g Wi-Fi are present in Fit-PC shown in Figure 2.2. It further supplies six USB 2.0 port, a mini PCI Express slot and a microSD card.

2.2.1.1 Why Fit-PC?

The reason why Fit-PC is used in this thesis is that it is small. Not only Fit-PC is one of the smallest computer in the world but also it is very light. Another reason for using it is that it saves energy compared to other notebooks.

Figure 2.2: Fit-PC

2.2.2 Arduino

Arduino, shown in Figure 2.3, is a general purpose device which enables computers to control the physical world from the desktop computers. It consists of a microcontroller board and provides an open source environment for software development. Arduino is used to develop interactive objects by taking inputs from various switches or sensors, and it controls motors and physical outputs. Arduino projects can be used alone or with a software program (i.e. C, C++, Matlab and Java) which is already running in the computer.

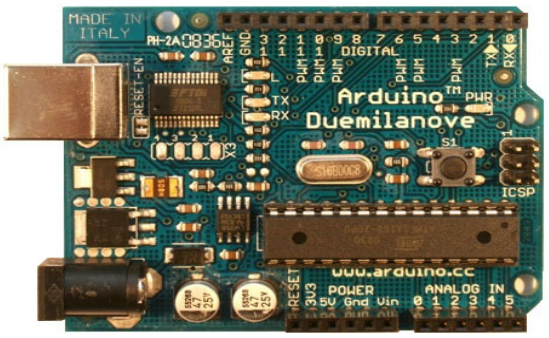

Figure 2.3: Arduino

2.2.2.1 Why Arduino?

First reason why the arduino used is that sensors and some other device controls such as GPS and compass cannot be directly connected to the computer and arduino serves as a bridge between these parts and the computer. Another reason is that its an open source development environment and also the community working with arduino is quite big.

Although there are many other microcontrollers and microcontroller platforms for physical computing, arduino differs from others in terms of simplifying the process of working with microcontrollers. This tool is practical so it is easy-to-use both by professionals and beginners. It is also relatively inexpensive and compatible with a number of operating systems such as Windows, Linux and Macintosh OSX.

2.2.2.2 Features of Arduino

- Microcontroller ATmega168
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V
- Digital I/O Pins 14 (of which 6 provide PWM output)

- Analog Input Pins 6
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
- SRAM 1 KB (ATmega168) or 2 KB (ATmega328)
- EEPROM 512 bytes (ATmega168) or 1 KB (ATmega328)
- Clock Speed 16 MHz

2.2.2.3 Software of Arduino

Arduino software runs on Windows on which Java Virtual Machine(JVM) is installed. And also arduino software is compatible with Windows, Macintosh OSX and Linux.

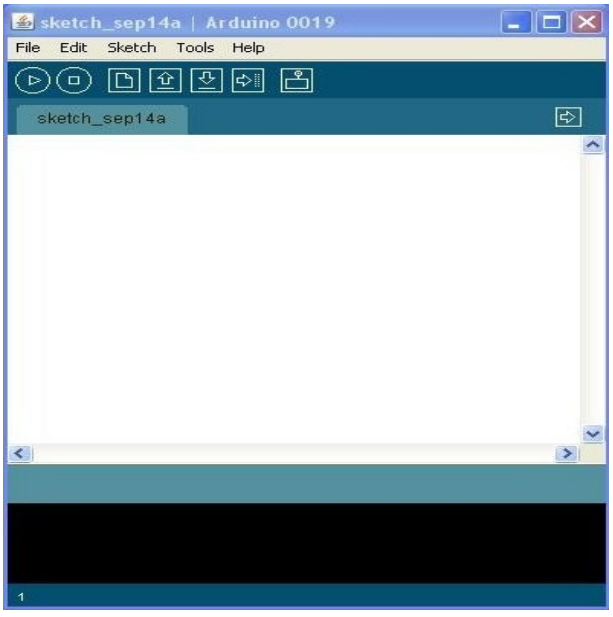

Figure 2.4: Arduino Interface

Figure 2.4 illustrates Arduino software Interface.

2.2.3 Ultrasonic Sensor

Ultrasonic sensors, given in Figure 2.5, can detect object using high frequency waves without actually contacting with the objects. The Parallax Ping))) sensor, that is what we use in this project, measures distance using sonar, in which an ultrasonic pulse is transmitted from the unit and distance-to-target is determined by measuring the time necessary for the echo return. It provides low-cost and it is practical for measuring the distance from any object whether it is moving or stationary. In this project, five Parallax Ping))) sensors are used.

Parallax Ping))) sensors are not affected by the color and brightness of objects and they are better than other sensors in terms of being affected by the surface and material of objects. Furthermore, they do not require maintenance and it can identify small objects from long distance. The last advantage of an ultrasonic sensor is that it is resistant to vibration, noise and EMI radiation.

On the other hand, there are a few disadvantages of ultrasonic sensors that needs to be presented at this point. To begin with, ultrasonic sensors can not give an exact value of the distance between the object and the car because of the noise around. This can be listed as the biggest disadvantage of the ultrasonic sensors. Secondly, as we worked in real time, it takes some time for ultrasonic sensor to measure the distance between the vehicle and the object.

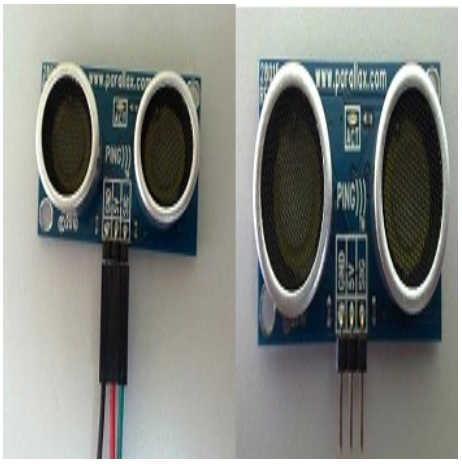

Figure 2.5: Ultrasonic Sensor

2.2.3.1 Features of Ultrasonic Sensor

- Provides precise, non-contact distance measurements within a 2 cm to 3 m range
- Simple pulse in/pulse out communication
- Burst indicator LED shows measurement in progress
- 20 mA power consumption
- Narrow acceptance angle
- 3-pin header makes it easy to connect using a servo extension cable, no soldering required

2.2.4 Restricted Digital Video Camera (Webcam)

Digital Video Camera is connected to a computer, and gathers a series of images. In this study, webcam is used to observe the environment. The reason why it is preferred is that it is inexpensive.

2.2.5 Global Positioning System (GPS)

Global Positioning System, as seen in Figure 2.6, is a satellite web which sends regularly coded information, and makes it possible to find and detect a spot on earth by measuring the distance between the satellites and us.This system consists of 24 satellites which belong to US defense department and revolve around the orbit continuously [20].These satellites give off very low level radio signals. The GPS receiver on earth receives these signals. Thus, position detection can be done. In other words, GPS device makes it possible to sign our position and returns that position. GPS is able to work in every place except for closed areas and underwater where it is hard to get signals. In this project, we work in indoor environment, therefore GPS device is not used in our project.

Figure 2.6: GPS

2.2.6 Compass

This magnetic compass, in Figure 2.7, has many advantages including heading information. It is the only device which can provide absolute heading information without any other outside reference for calibration. Today's electronic compass can connect with micro-controls easily, and involve some features such as power save and calibration. However, compass has their own uncertainties and problems. Therefore, one needs to understand how electronic compasses work and the effect of environment conditions on them so as to use this technology in its best way in projects. It is also important to set the appropriate experiment environment for digital compasses as they are sensitive to magnetic areas. For these reasons, compass is integrated in the autonomous vehicle; however, we have not used in this thesis.

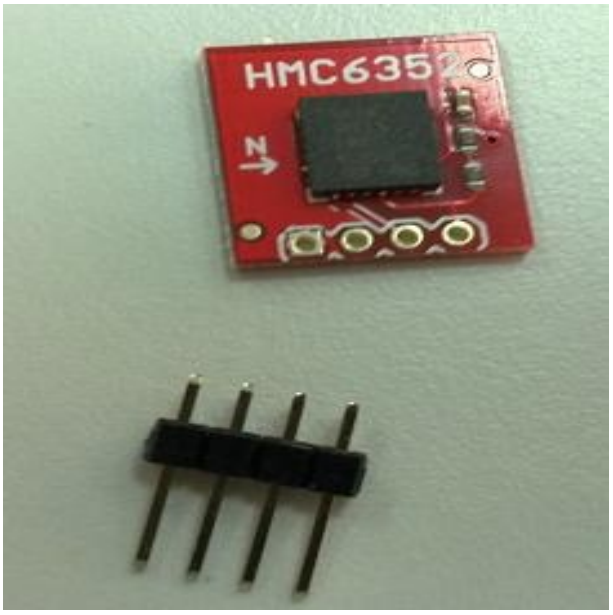

Figure 2.7: Compass

2.2.7 Vehicle (Traxxas)

In this project, we have chosen Traxxas, in Figure 2.8, model which works with electricity. Needless to say, any model car can be used. The reasons why we have chosen Traxxas are that it has a very good suspension, and it comes already installed. The most important reason that it is cheap and spare parts are easy to find. Also note that it carries heavy load which enables us to use some devices such as battery, netbook, arduino and camera on this vehicle.

Figure 2.8: Traxxas

2.2.8 Li-Po (Lithium-ion Polymer)

Model car that is chosen for this project works with Li-Po (Lithium-ion polymer) batteries which provide long trial drives. Besides, it is charged in a short amount of time and it works Fit-pc and car for 6 hours without charging. In addition, while other batteries (Ni-Cd, Ni-Mh, etc) are charged in 16 hours, Li-Po battery, in Figure 2.9 is charged only in an hour, which is time saving. We have chosen 3 cells series connecting Li-Po batteries as the operating voltage of the Fit-pc is between 9V-15V. 3 cells series connected Li-Po battery gives 9.9V as the minimum voltage and 12.4V as the peak voltage. 4.0 Ah is chosen for this project since Fit-pc draws approximately 0.5A. Therefore 4000 mAh gives us 8 hours of Fit-pc operating time. On the other hand, Li-Po batteries have a disadvantage; battery tends to die below 9.0 V. Hence, a simple hardware implemented critical level battery alarm is used. Battery alarm gives warning, if it bellows 9.9V.

Figure 2.9: Lithium-ion Polymer Battery

All equipments of the car, mentioned above, are put together neatly. As shown in Figure 2.10, 3 ultrasound are fixed onto the car while two ultrasound are fixed on the corners which will be explained in detail. As mentioned before, since we work in indoor environment, we did not use GPS and compass but GPS and compass

devices are connected to the autonomous car. By the help of arduino, all sensors, GPS and compass were tied to the Fit-pc.

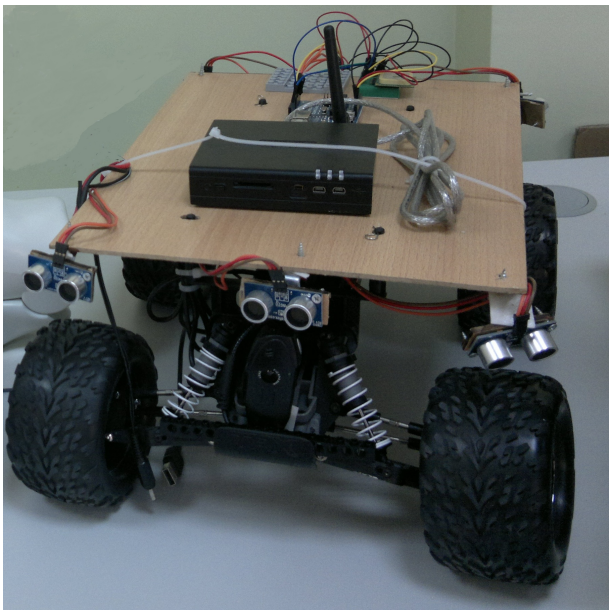

Figure 2.10: Autonomous Car

2.3 Running Environment

Figure 2.11 illustrates the running environment which is covered with a green carpet. Tenpins are placed on the corner of the green carpet. These tenpins are used as landmarks.

Figure 2.11: Running Environment

2.3.1 Landmark

Landmarks are features which can easily be re-observable and distinguished from the environment. The car uses them to find out its location. It is crucial for the landmarks to be re-observable as they should be detected from different positions and angles. Moreover, they should be unique in the sense that they are identified easily without being mixed. The key points of suitable landmarks are:

- Landmarks should be re-observable easily.
- Individual landmarks should be distinguishable from each other so that they are not mixed.
- Landmarks should be plentiful in the environment.
- Landmarks should be stationary.

In this project, tenpins with different colors are used as landmarks. As shown in Figure 2.12, unique tenpins are placed in selected locations of the environment.

Landmarks are characterized by their location in Q_k for k = 1,....,number of landmarks. Landmark is as points x,y in the plane therefore locations are specified by two numerical values which are stored in a landmark array.

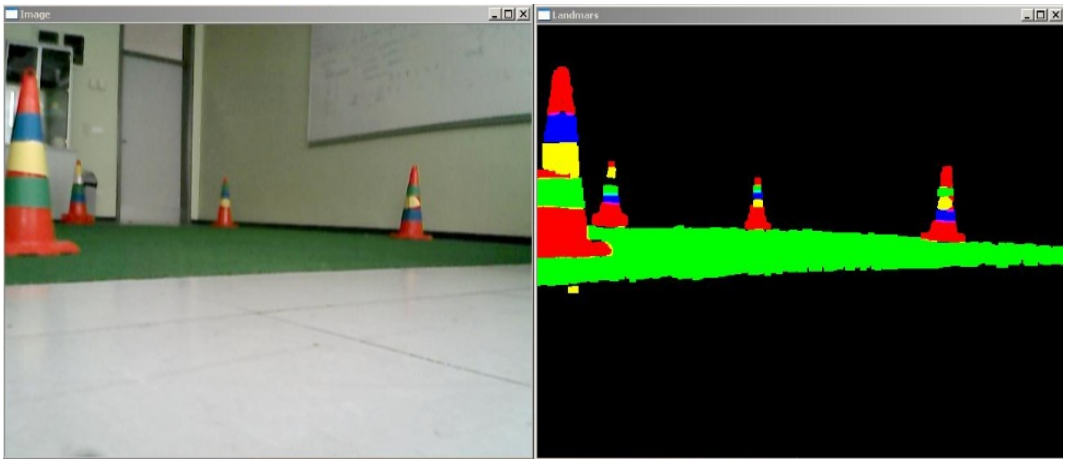

Figure 2.12: Landmarks

Chapter 3

Moving in the Environment

Before dealing with SLAM, the vehicle is required to be safely moving in the environment. As the environment is decorated with a number of landmarks, which are later used in SLAM as well, it is necessary to identify those landmarks, to move towards one of them. It is aimed that the autonomous vehicle finds the closest landmark, which is assumed to be the biggest object in the image taken by web cam. Then the car is commanded to move towards it. The vehicle should also be prevented to crash into the landmarks. The chapter explains how the vehicle has gained these abilities.

3.1 Landmark Detection

Since there are predefined landmarks in the environment, an application has been developed for the vehicle to detect which landmark it encounters. From the image taken with web cam, to extract and identify the landmarks, some image processing algorithms have been employed, which are;

- Pixel Detection Algorithm
- Filtering Algorithm
- Grayscale Morphological Operations

Following subsections introduce what a digital image is, and informal definitions of above algorithms.

3.1.1 Digital Image

Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or, a set of characteristics or parameters related to the image [21]. Image processing utilizes computer algorithms to perform processing on digital images.

Digital image is a numeric representation of a two-dimensional image. A digital image consists of pixels which stand for picture elements. As the image size gets bigger the number of pixels increases. A digital image can be represented as a matrix which contains these pixels where the values of pixels represent intensity values. For example, a common grey level image pixel values are in the range of 0 - 255. The value shows the intensity of pixels. 0 denotes darkest pixel and 255 denotes the brightest pixel [22].

In this project, three types of images are used. These are:

- Binary Image
- Greyscale Image
- 3 Channel RGB Color Image

3.1.1.1 Binary Image

In binary images, pixel values can be either 0 or 1. 0 value shows the black pixels and 1 value shows the white pixels. The binary image has been used to find landmarks on the environment. The binary images in Figure 3.1, 3.2

and 3.3 are obtained by applying, red pixel, green pixel and blue pixel detection algorithms (see section 3.1.2) to the input image, respectively.

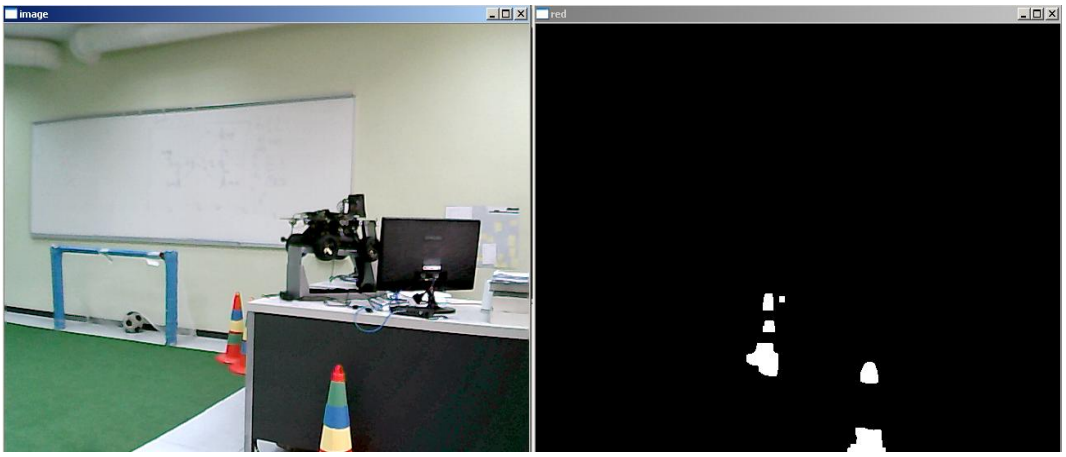

Figure 3.1: Input Image (Original mage) - Binary Image for red

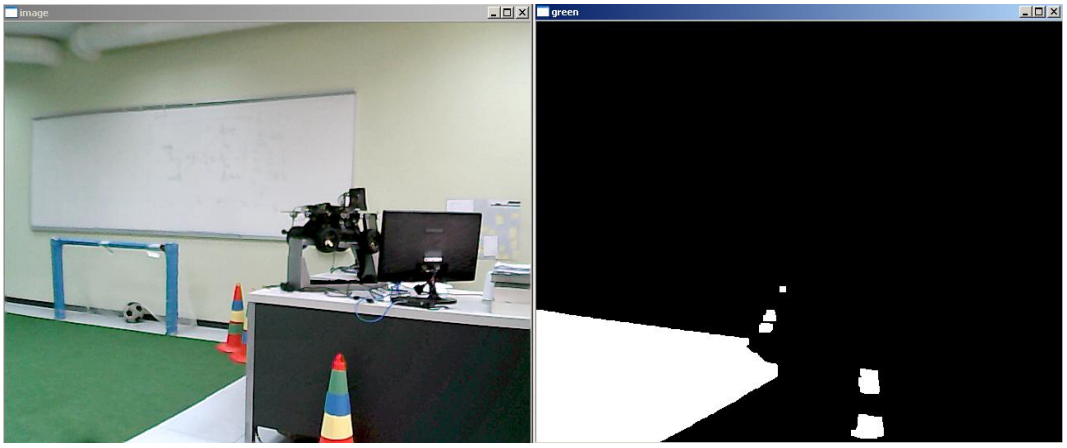

Figure 3.2: Input Image (Original mage) - Binary Image for green

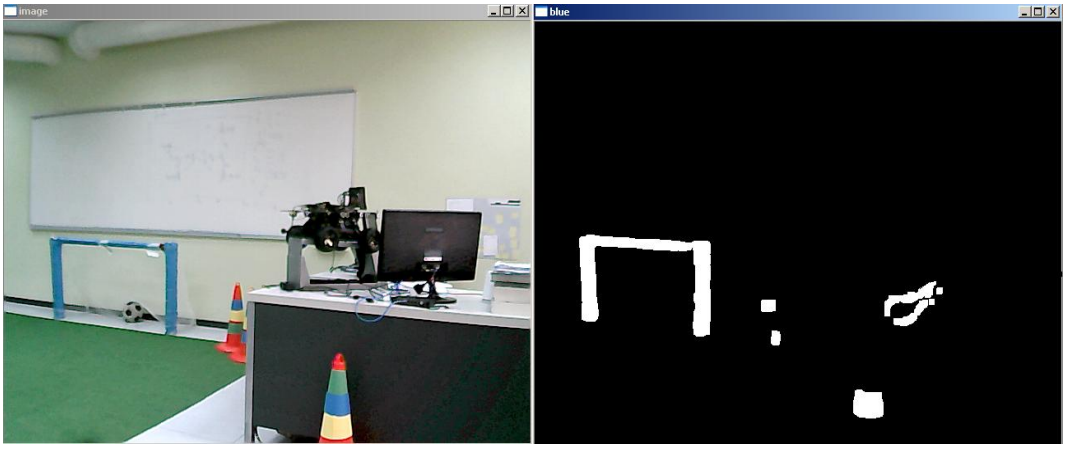

Figure 3.3: Input Image (Original mage) - Binary Image for blue

3.1.1.2 Grayscale Image

In a grayscale (or gray level) image in Figure 3.4, there are only colors and shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a 'gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive gray level is significantly better than the gray level resolving power of the human eye.

Grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images. Grayscale images have been used to find the biggest object.

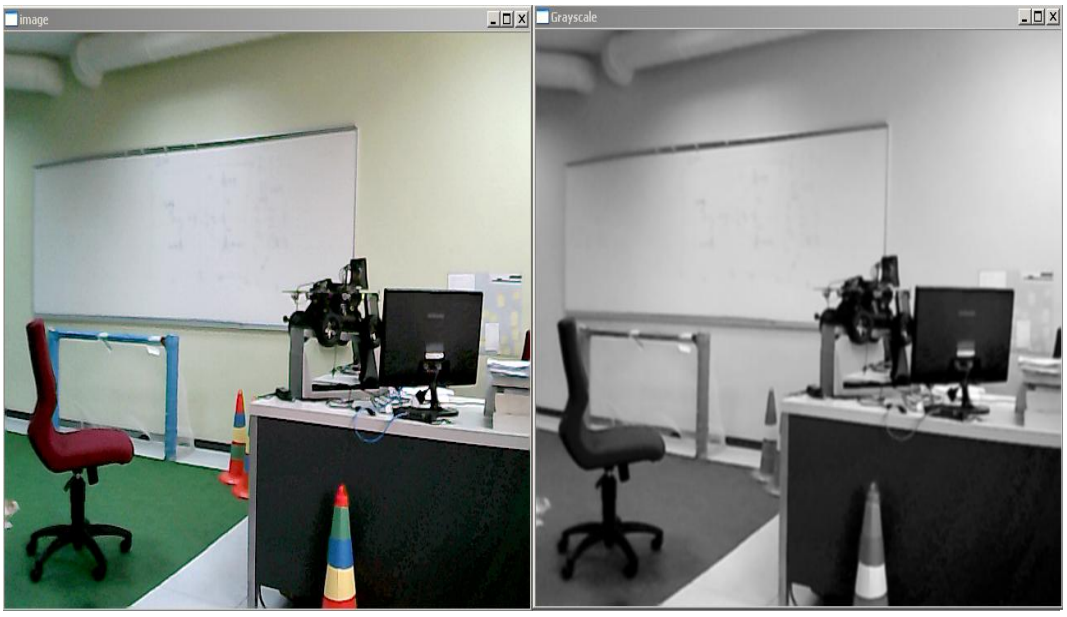

Figure 3.4: Input Image (Original Image) - Grayscale Image

3.1.1.3 3 Channel Color Image (RGB)

An RGB image has three channels which are red, green, and blue. RGB channels roughly follow the color receptors in the human eye, and are used in computer displays. If the RGB image is 24-bit which it is used in this project, each channel has 8 bits, for red, green, and blue. In other words, the image is composed of three images (one for each channel), where each image can store discrete pixels brightness intensities between 0 and 255 [23]. Figure 3.5 shows the RGB channels of the color image.

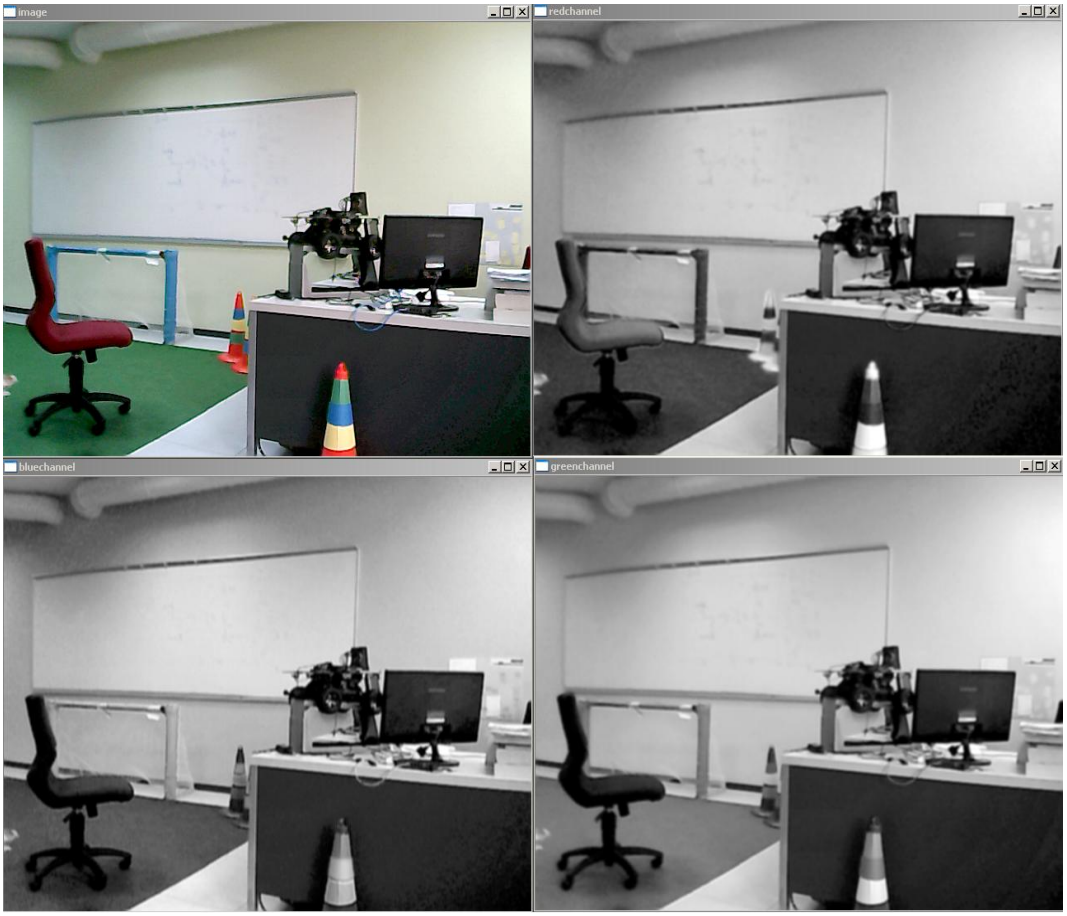

Figure 3.5: RGB channels for the input image

It is possible to find all other colors by the mixture of red, green and blue channels. Figure 3.6 illustrates how other colors are found. For instance, if we are looking for black color, this means that we are detecting the color in which all three channels have 0 value (0, 0, 0). Accordingly, if we are looking for cyan color, this

means that we are detecting the color in which blue and green have 255 value and red has 0 value (Figure 3.6).

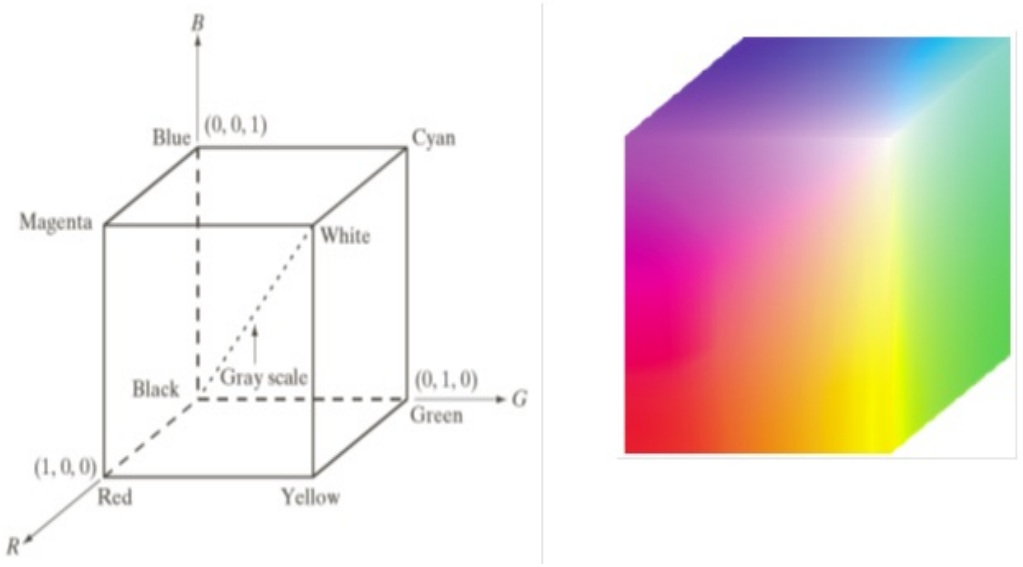

Figure 3.6: RGB Color Cube [24]

3.1.1.4 Conversion From 3 Channel Color Image (RGB) to graylevel Image

It requires a considerable amount of process load to proceed with a three channel image such as biggest component. To decrease computation power, first, the color image is converted to grayscale image. Then some operations (morphological operations, connected component labeling, filtering) are applied to the input image. Under the condition that there is a three channel color image and it is required to convert it to grayscale image in an equal weighted way, each pixel value is recalculated by:

$$\text{grayscale pixel value} = \frac{RED + GREEN + BLUE}{3} \tag{3.1}$$

RED : Red channel pixel value.

GREEN : Green channel pixel value.

BLUE : Blue channel pixel value.

3.1.2 Pixel Detection Algorithm

Pixel detection algorithm has been used to explore the color of a particular pixel. As mentioned before, there exists three channels in RGB image and other colors are made by using red, green and blue channels (See Section 3.1.3). In brief, pixel detection algorithm works in this way. The values in three channels are checked and each of these three channels are filtered one by one. The threshold value, which changes depending on the light condition, is compared to the value in each pixel. If a pixel value, is higher than the threshold value, that pixel is labeled as the one that is being asked for. An example of blue pixel detection algorithm is;

$$\text{if } (2 \times blue[i] - (red[i] + green[i]) > thr) \text{ This pixel is blue} \tag{3.2}$$

thr : Threshold value

blue[i]: i'th pixel's blue channel value

red[i] : i'th pixel's red channel value

green[i] : i'th pixel's green channel value

3.1.3 Filtering

Filtering is used to reduce the noise in the image. Pixel detection can be used as a filtering operation. There are two criteria for choosing the threshold value: the color histogram of the image, and the lighting conditions of the environment.

Besides the simple pixel detection method, there are some more sophisticated filters as well. Among many others, we have used two filters:

- Mean Filter

- Gausssian Filter

3.1.3.1 Mean Filter

The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. Mean filter is used to reduce noise. In other words, mean filter filters the images that are coming from the camera and minimizes the noise. Mean filtering circuits all pixels of an image with a 3x3 sliding window. Then, the average of 9 values is taken and written at the center of window. This applies to all pixels of the image except for first and last pixels. In Figure 3.7, mean filter is applied to input image.

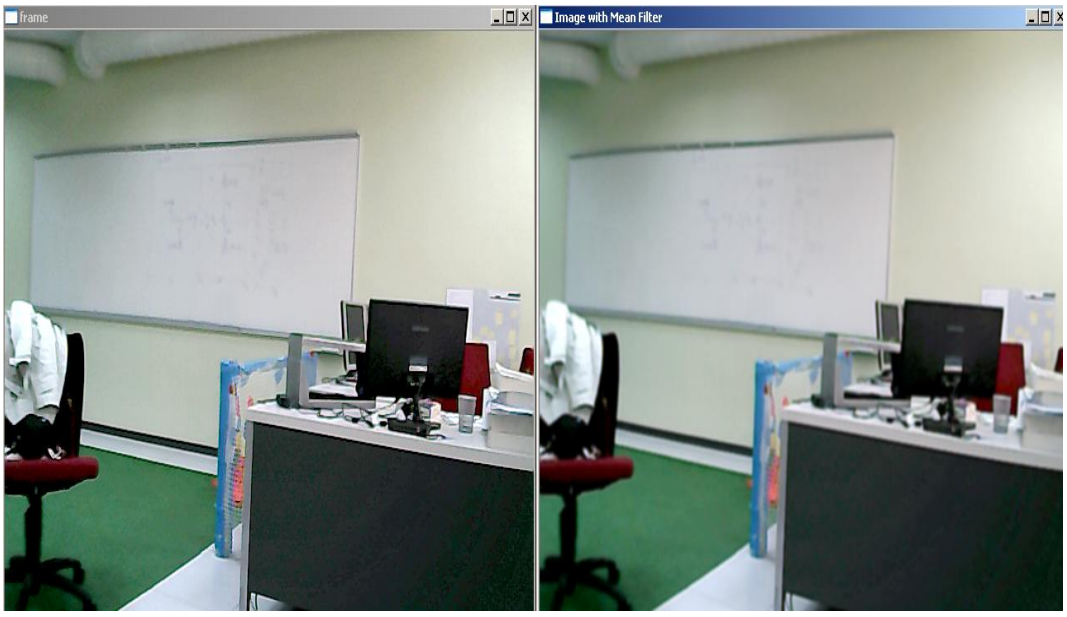

Figure 3.7: Mean Filter

3.1.3.2 Gaussian Filter

The Gaussian filter is a filter that is used to blur images to remove details and noises by calculating weighted averages in a filter box [25]. The equation of Gaussian filter is;

$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}} \tag{3.3}$$

σ : Standard deviation of the Gaussian 2-D distribution x : Distance from the origin in the horizontal axis y: Distance from the origin in the vertical axis

In Figure 3.8 Gaussian filter is applied to input image.

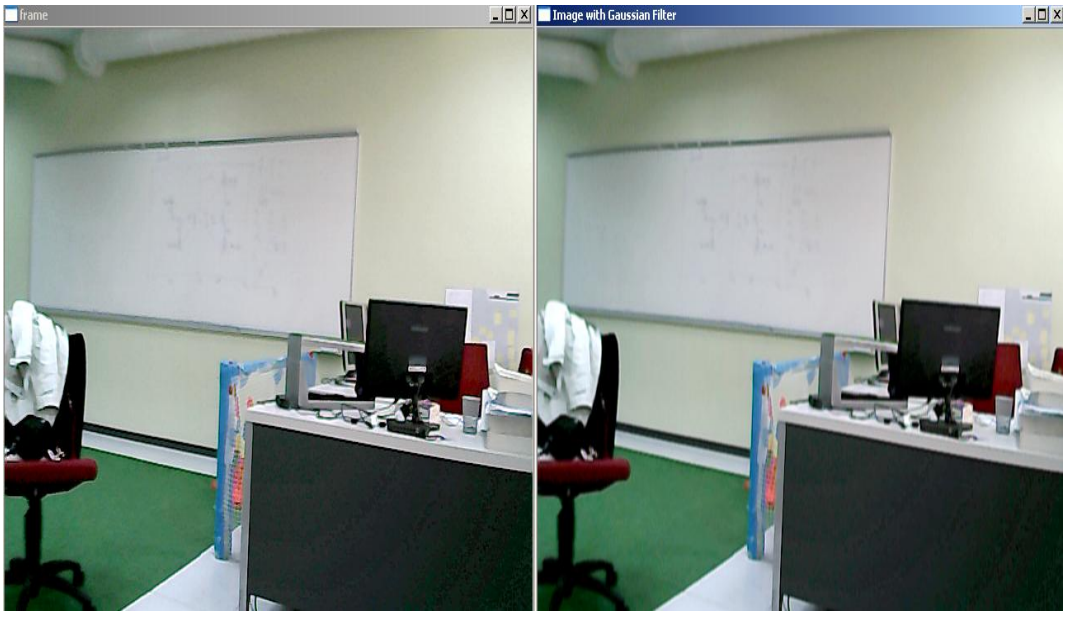

Figure 3.8: Gaussian Filter

3.1.4 Grayscale Morphological Operations

Morphological Operations (MM operation) are based on set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems. Sets in mathematical morphology represent objects in an image. For example, the set of all white pixels in binary image is a complete morphological description of the image [26].

3.1.4.1 Pixel Neighborhood

When dealing with images and applying image morphology functions, not only a single pixel is considered but also the surrounding pixels are taken into account. The pixels which are adjacent to the current pixels are called neighbors of a pixel. There are two common types of neighborhood which are 4 neighborhoods and 8 neighborhoods [27].

Figure 3.9 shows the two types of neighborhood. The letters in the boxes denote the directions (north, east, south, west, northwest, southwest, and northeast, southeast) and the empty box in the middle denotes the current (center) pixel. In this project, 8 neighborhood version is used to get more accurate results.

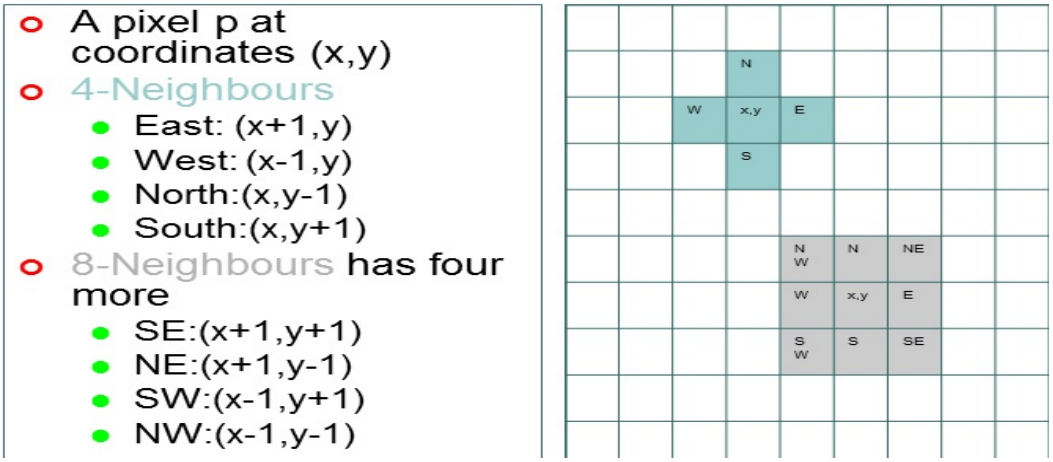

Figure 3.9: Pixel Neighborhood[28]

3.1.4.2 Opening Operation

The opening operator tends to remove some of the foreground (bright) pixels from the edges of regions of foreground pixels [29]. It is derived from the fundamental operators erosion and dilation. Like those operators opening operator is normally applied to binary or graylevel images.

The opening of image f by structuring element b, denoted f ∘ b, is

$$f \circ b = (f \ominus b) \oplus b \tag{3.4}$$

Erosion Operation: The basic effect of the operation on a binary image is to erode away the boundaries of regions of foreground pixels (i.e. white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger.

Morphological Erosion Operation is that, the erosion of binary image A by structuring element B is denoted by A $\ominus$ B and defined by

$$A \ominus B = \{z|(B)z \subseteq A\} \tag{3.5}$$

The structuring element is swept over the image. At each position where every 1-pixel of structuring element covers a 1-pixel of the binary image, the binary image pixel corresponding to the origin of the structuring element is ORed to the input image [30].

Dilation Operation: The basic effect of the operation on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (i.e. white pixels, typically). Thus areas of foreground pixels grow in size while holes within those regions become smaller.

Dilation Morphological Operation is that the dilation of binary image A by structuring element B is denoted by A $\oplus$ B and defined by

$$A \oplus B = \{z|(B)z \cap A \neq \varnothing\} \tag{3.6}$$

As mentioned before, the structuring element is swept over the image. Each time the origin of the structuring element touches a binary 1-pixel.

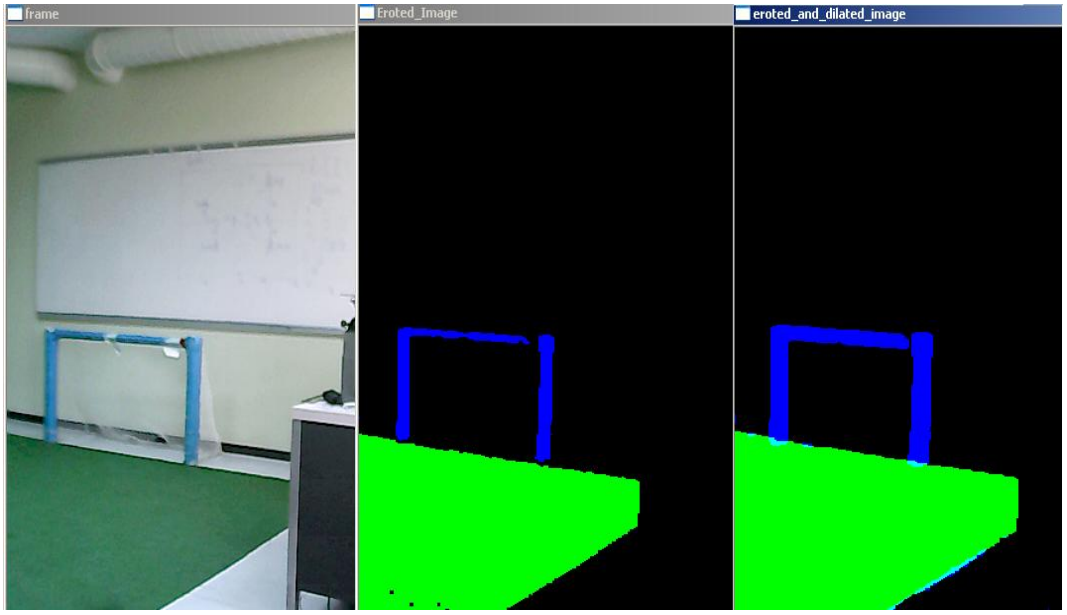

Figure 3.10: Opening Morphological Operation

Opening operation means carrying out dilation operation after erosion operation. Since they are binary operations, erosion and dilation operations are applied to three channels separately in a three channel input image. Afterwards, these three channels are combined. Figure 3.10 is an example of opening operation to the input image.

As mentioned before, opening morphological operation is used in this project. In opening morphological operation, first image erosion and then dilation morphological image is applied. The main reason why it is used is that image erosion operation deletes all noises and dilation operation widens the wanted regions. This makes it easy to find the wanted object.

3.2 Finding the Closest Object

In the image taken by web cam, the closest object is identified as largest object where all the objects are of the same size. As mentioned before, landmarks have been used as standard sized objects.

To find the nearest landmark to the vehicle, the images taken from camera have been filtered to minimize the noise, and opening morphological operator has been applied. The resulting image is a binary image containing only the regions we are interested in. The difficulty here, there might be some objects in the image, which do not correspond to anything in the scene, that is noise in a sense. To overcome this difficulty, connected component labeling has been applied and the regions of image which are supposed to be the landmarks have been marked.

The last step in finding the nearest landmark is to choose the one of the connected components having the biggest pixel count (number of pixel).

Then the center of mass of this object is computed, so that the vehicle can calculate the turning angle and speed to move towards it.

3.2.1 Connected Component Labeling

Connected component labeling is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled. It scans an image, pixel-by-pixel (from first pixel to last pixel). In an image, there may be more than one wanted images. Thus, it labels all objects one by one to find the nearest object. Connected component labeling works on binary or graylevel images, so it converts the image that we get from the camera to binary image (See Section 3.1.4), then it gives values starting from 0 to the wanted pixel and look at its 8 neighbors. If the neighbors have the wanted value, the same value is also given to them. If not, counter increases the value only one and proceeds till the end of the image. For instance; in Figure 3.12, blue objects are detected and found by doing connected component labeling. The pseudocode of connected component labeling algorithm is;

```
1  label=0
2  for i=1 to height-1
3      for j=1 to width-1
4          if image[i,j] or one of the 8 neighborhoods is labeled which is
small labeled to the connectedimage[i,j]
5              temp=label
6              label= connectedimage[i,j]
7          end
8          else
9              label++
10         end
11         if image[i,j] and its 8 neighborhoods are equal to 1
12             connectedimage[i,j]=label
13         end
14     label=temp
15     end
16 end
17 Again this code to start from last pixel to first pixel
```

Figure 3.11: Pseudo code for Connected Component Labeling Algorithm

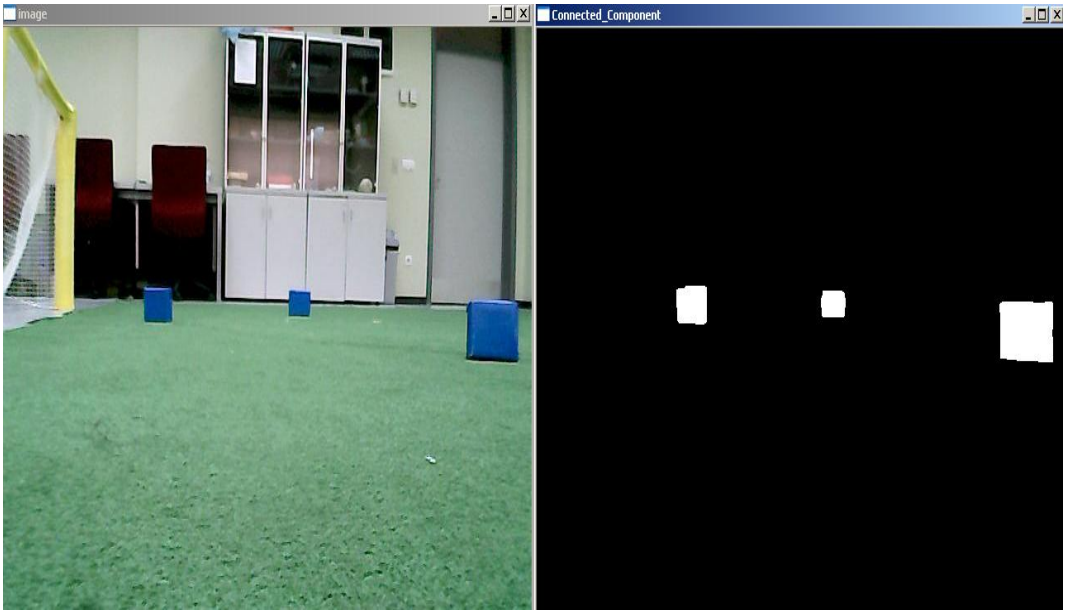

Figure 3.12: Connected Components Labeling

After the application of connected component labeling to the image in figure 3.12, the nearest object is found in Figure 3.13.

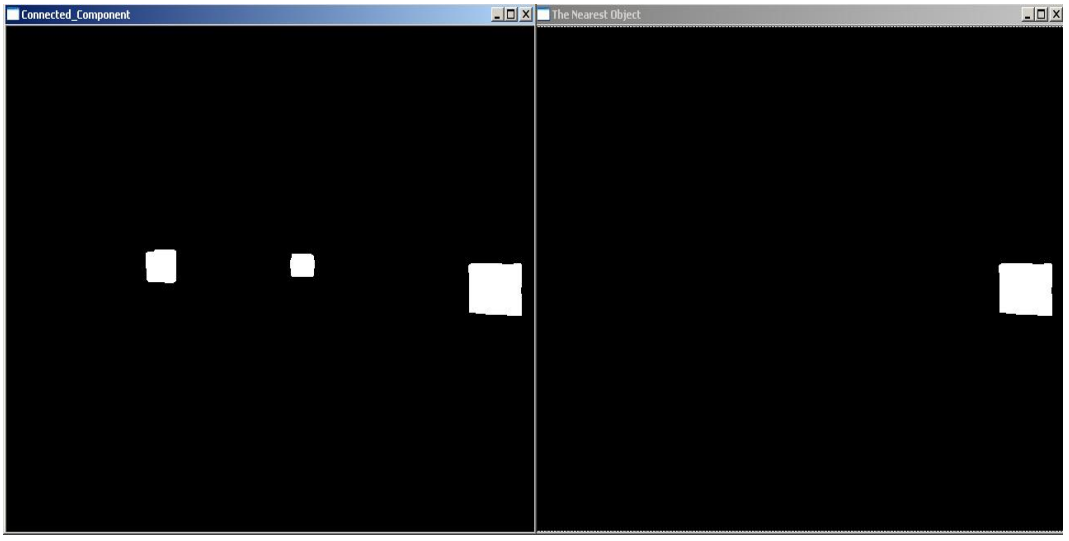

Figure 3.13: The Nearest Object

3.2.2 Center of Mass

The center of mass is found for the biggest connected component. The center of mass equation is in equation 3.7 and equation 3.8.

$$\text{center of mass x} = \sum rowindex \div allindex \tag{3.7}$$

$$\text{center of mass y} = \sum columnindex \div allindex \tag{3.8}$$

rowindex : Row index value of each 1 columnindex : Column index value of each 1 allindex : Number of total active pixels (1's)

In these equations, the center of mass x coordinate finds the angle between the object and autonomous vehicle(see section 3.3); while, center of mass y coordinate gives the nearest object(see section 3.2).

To find the center of mass coordinates, algorithm is

```
1  total=0
2  massx=0
3  massy=0
4  for x =1:height-1
5      for y =1:width-1

6          if(connectedimage[x,y] =labelnumber)
7              massx+= x
8              massy+=y
9              total++
10         end
11     end
12 end
```

Figure 3.14: Pseudo code for Center of Mass Algorithm

3.3 Controlling the Vehicle

As mentioned before, there are five ultrasound sensors, 4 on the corners and one in front of the car, and also one camera is in front of the car. After the car runs, data are gathered through sensors and camera. There is no need to do extra work as the camera is directly connected to the fit-pc; however, data from sensors first come to arduino before fit-pc as sensors are connected to arduino. Therefore, there is more amount of process load when the data is gathered from the sensors than from the camera. Using the data coming from ultrasound sensors and camera the speed and direction of the vehicle are adjusted.

3.3.1 Servo and Speed Control

Servos are small devices which have smart and programmable miles. It is possible to change the mile's position in any angle that we want by sending specific codes to the servo. The angular position of the mile changes as the codes change. That is to say, servo is responsible for the vehicle to gravitate to the asked direction. The servo values are calibrated to the angle (degree). Table 3.1 illustrates the angle (degree) which corresponds to the servo value. In this project, it is determined

as 100 value for the car to be straight, which means that the wheels will have the same direction with the car if 100 value is sent to the servo. If the servo value is increased, for instance 110, the vehicle gravitates to the left. Similarly, it gravitates to the right side if the degree is decreased. It is assumed that the degree for maximum left direction is 145 value. While it is 55 value for maximum right direction. Figure 3.15 represents the turning angle of the car. 100 value is the neutral position for the car. The maximum turning angles are 55 and 145 values. Moreover Figure 3.16 illustrates the calibration from servo values to angle (degree) for all servo values.

Table 3.1: Calibration of Servo Value and Angle

Servo (value)	Turning Angle (degree)
55	45
60	40
65	35
70	30
75	25
80	20
85	15
90	10
95	5
100	0
105	5
110	10
115	15
120	20
125	25
130	30
135	35
140	40
145	45

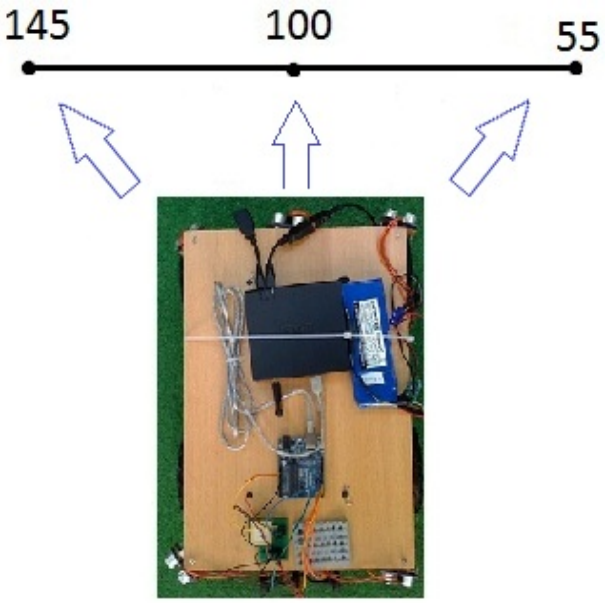

Figure 3.15: Turning Angle of the car

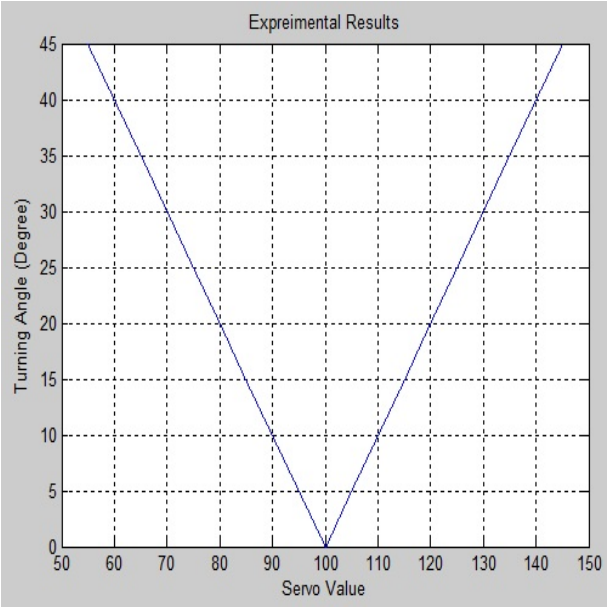

Figure 3.16: The graph of calibration from Servo value to Angle(degree)

This is how the car follows an object. The car is asked to find to go the nearest object which is blue. First, the location of the nearest blue object is identified by using the camera. Later, the center of gravity is found for the object in Figure 3.17. As the turning angle of the car and the angle of the object in the image to the car are different, these two angles are scaled to each other. The formula of this scaling is:

$$\beta = k + \left(\frac{diff(resolx - x)}{resolx} \right) \tag{3.9}$$

k : minimum servo turning value which is 55

Diff : The space between initial and last servo value which is 90

Resolx : The resolution of x coordinate which is 640

β : The turning angle of the car

x : The x coordinate of the center of gravity of the object.

In Figure 3.17, in which there is 640x480 resolution, the x and y coordinates of the vehicle are assumed the center of an image which is 320,480. Taking the formula into consideration, if the mass center coordinates of the object are 400,200, turning value of the servo will be 88. The coordinate of y shows how the vehicle is close to the object.

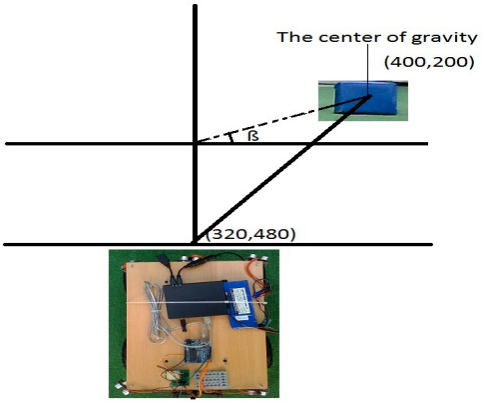

Figure 3.17: Locations of Vehicle and Object

The speed of the car changes according to the data coming from sensors. The vehicle slows down and stops in case of an obstacle. If there is no obstacle, it gathers speed in every 't' time. The value of 90 is given to the vehicle when it is in stop position. A value more than 90 is given to make it gain speed, and 135 value is given to make it go the correct direction with maximum speed. Similarly, a value smaller than 90 is given so that it goes backwards. The value of is sent to the motor to make it go backwards with maximum speed. Table 3.2 illustrates the speed value which corresponds to speed(m/sec) and velocity(v).

Table 3.2: Calibration of vehicle speed value, speed (m/sec), and velocity

Motor Speed (value)	Speed (m/sec)	Velocity (V)
75	10.8	-10.8
80	4.5	4.5
85	1.1	1.1
90	0	0
95	0.9	0.9
100	4.2	4.2
105	10.6	10.6

In Figure 3.18, the graphic shows the car speed value which corresponds to speed (m/sec).

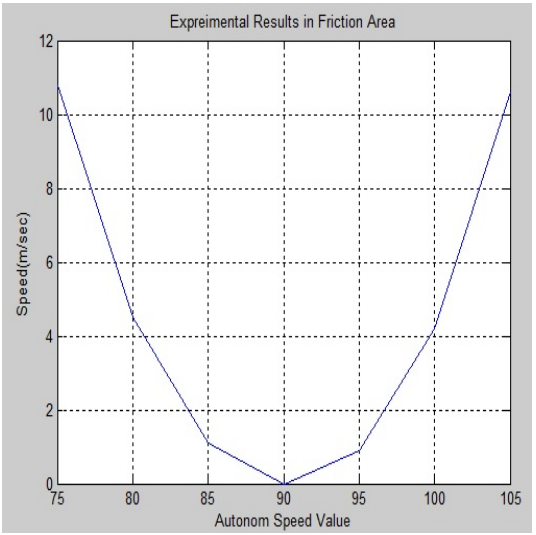

Figure 3.18: The graph of calibration from car speed value to speed(m/sec)

Chapter 4

Simultaneous Localization and Mapping

Hugh Durrant-Whyte and John J. Leonard developed Simultaneous Localization and Mapping (SLAM) inspiring by the work of Smith [31, 32]. While Durrant-Whyte and Leonard originally named it as SMAL, it was later changed so as to entail a positive effect. SLAM is a technique which is a related to the concern of developing a map of an unknown environment by using a mobile robot and navigating the environment using that map.

While SLAM localizes the positions of the landmarks by using vision measurement and ultrasonic sensor, it also computes how much the vehicle moved by using odometry data. Then SLAM computes where the landmarks are supposed to be by looking at the odometry data and it re-extracts the landmarks. Finally, using Kalman filtering, odometry data and landmark positions are filtered, and the position of the vehicle is computed. SLAM consists several subsystems such as odometry data association, state estimation, state update and landmark position update. The outline of the SLAM is shown in Figure 4.1.

SLAM is more like a concept than a single algorithm. There are many steps involved in SLAM and these different steps can be implemented using a number of different algorithms. In this thesis, camera and sensors are used to find the localization and mapping simultaneously of the autonomous car.

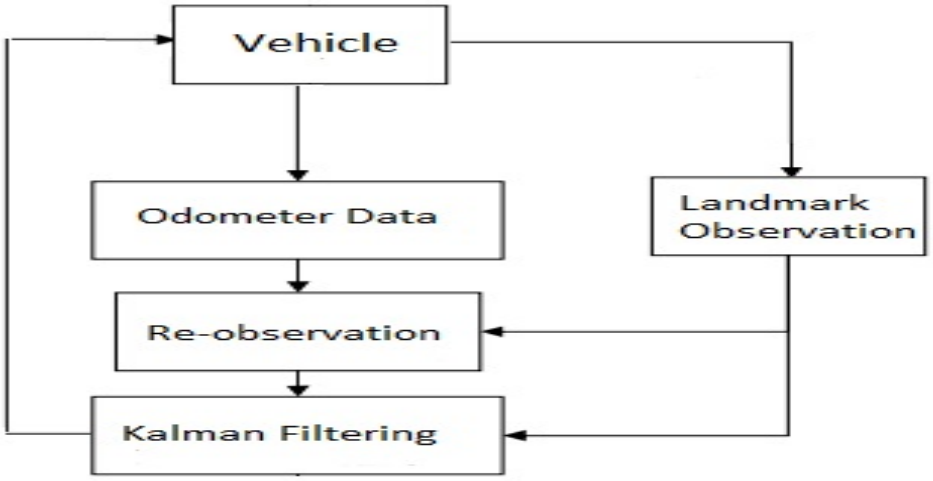

Figure 4.1: Outline of SLAM

The autonomous vehicle states s is referred to as the motion model, and given as the following probability distribution.

$$p(s_t|u_t, s_{t-1}) \tag{4.1}$$

s_t : The autonomous vehicle state s at time t

s_{t-1} : Previous state

u_t : The control of the vehicle at time t

The SLAM process, which handles the (4.1) can be summarized as:

Being in a position p, and knowing distances to each landmark d[i]

At each time step t:

1. Move robot
2. Estimate robot's and landmarks' new positions using odometry data
3. Measure the distance to the landmarks

4. Using the real and estimated distances to the landmarks, update the robot's position

Car's movement has been explained in Chapter3. Before going into explaining the rest of the process, it is worth to mention here that; once moving the car, it may not be possible to observe the landmark which has seen in previous time step, or vice versa. For instance, assume that at time step t, there are k landmarks in the scene; and at t+1, there are k-2 landmarks. Or, there are still k landmarks, but some are different then the previous one. This problem is known as data association problem.

In order to overcome the problem, we have removed the landmark from the environment that is already visited.

4.1 Odometry Data

The odometry data is basically car's instantaneous speed, and rotation of the wheels and is an important aspect of SLAM as it is used to estimate the car position. What is difficult for the odometry data is getting the timing right.

Table 4.1 illustrates the example of odometry data.

Table 4.1: Odometer Data

Time (sec.)	Speed (Traxxas Speed Value)	Turning Angle (Servo Value)
0.0	90	100
0.112	97	100
0.534	97	120
1.234	90	120
1.379	83	120
1.912	83	77
2.112	90	77
2.244	97	77
2.489	97	100

To determine the next state in the mapping process the distance traveled is calculated using the following basic formula 4.2.

$$Displacement = Velocity \times time \tag{4.2}$$

However this is not as simple as it seems due to the role played by time in the calculation process. The measurement errors are statistically dependent since errors in control accumulate over time. Figure 4.2 illustrates an example of the odometry data error. A robot's path, as obtained by its odometry is relative to a given map. The thick line represents the path followed, where the thin one is odometry data. Small odometry errors can have large effects on later position estimates.Also as the voltage drops the distance traveled decreases due to the lesser power produced by the battery.

Decreasing this error is the task of Kalman Filtering.

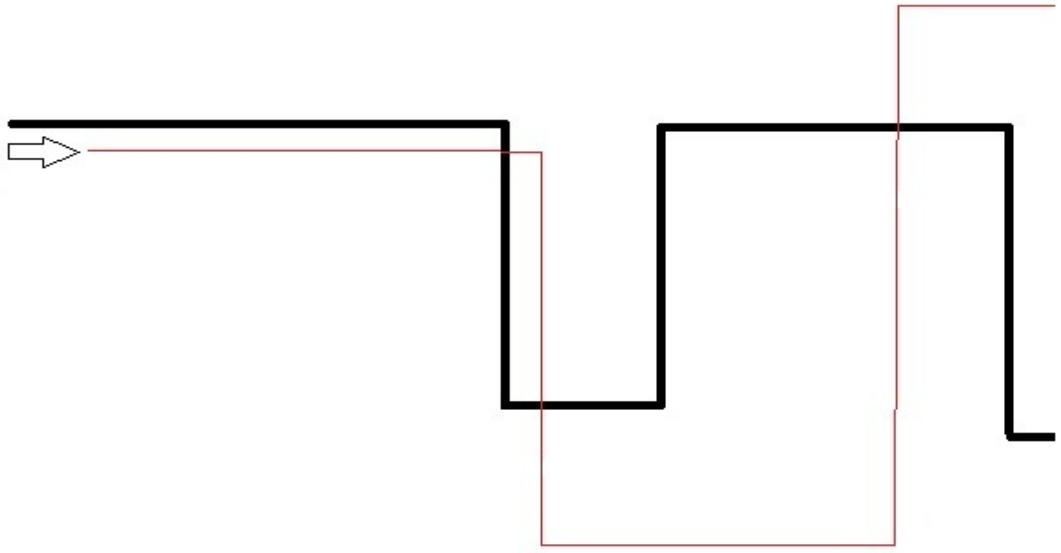

Figure 4.2: Odometer Data Error

4.2 Distance Measurement

In this project,sensors and a camera have been employed as the distance measurement device. Sensors are used that the car does not crash anywhere while navigating in indoor environment (see section 2.2.3). In other words Sensors are employed in detection of obstacles and to determine the location of the vehicle. The map is updated when there is an obstacle in front of it. The output coming from the sensor tells the range from any object in terms of meters. However, all ultrasonic sensors are subject to errors, often referred to as measurement noise. These sensors have strict range limitations (see section 2.2.3.1). Sensors does not provide the distance for a particular region hence it can not be exploited acquiring the distance to a certain landmark location. Therefore vision measurement has been exploited by means of a camera to determine the distance. Three methods are tested for vision measurement which are:

1. Flood Fill algorithm

2. Laser rangefinder

3. Finding the distance depending on the size of the object

4.2.1 Flood Fill Algorithm

After the color image is smoothed using average filtering, equations are calculated on the image. Distance measurement can be calculated with the following equations [33].

$$\alpha + \beta + \delta = 90^o \tag{4.3}$$

$$\tan(\beta) = \frac{b}{h} \tag{4.4}$$

$$\tan(\alpha + \beta) = \frac{b + d}{h} \tag{4.5}$$

$$\tan(\gamma) = \frac{w}{b+d} \tag{4.6}$$

$$y = h \times \tan(\beta + \frac{2\alpha(n-1-v)}{n-1}) \tag{4.7}$$

$$x = y \times \tan(\frac{\gamma(2u-m+1)}{m-1}) \tag{4.8}$$

α : One-half of the vertical field of view

γ : One-half of the horizontal

δ : Tilt angle of the camera

β : Angle of blind area

b : Blind area

h : Height of the camera from the floor

d : Distance the visible area

w : Visible area of the horizontal image

The values of b, d, h and w are calculated only once. In Figure 4.3 perpendicular and side view of the camera located in the robot is shown.

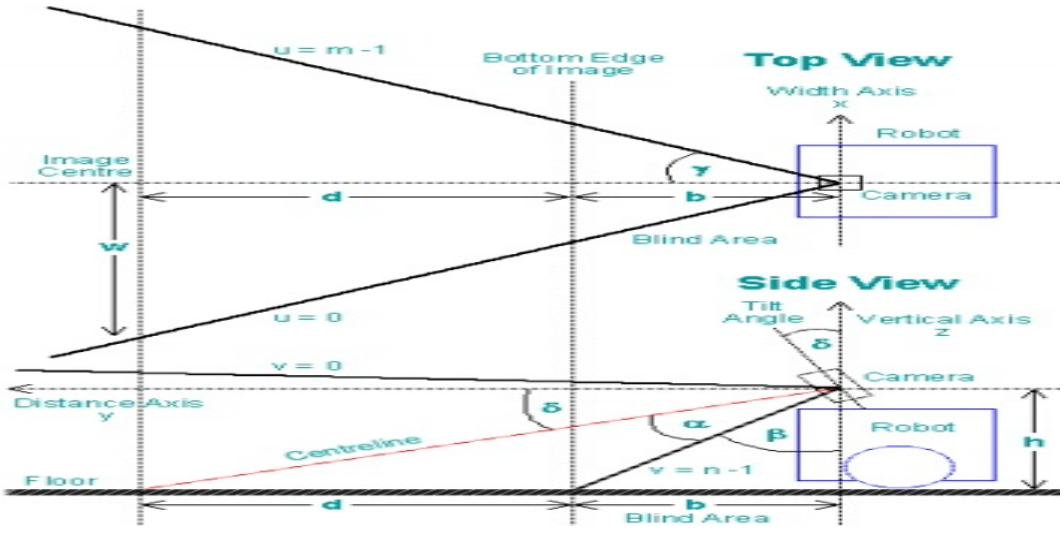

Figure 4.3: Outlook of Camera [33]

4.2.2 Laser Rangefinder

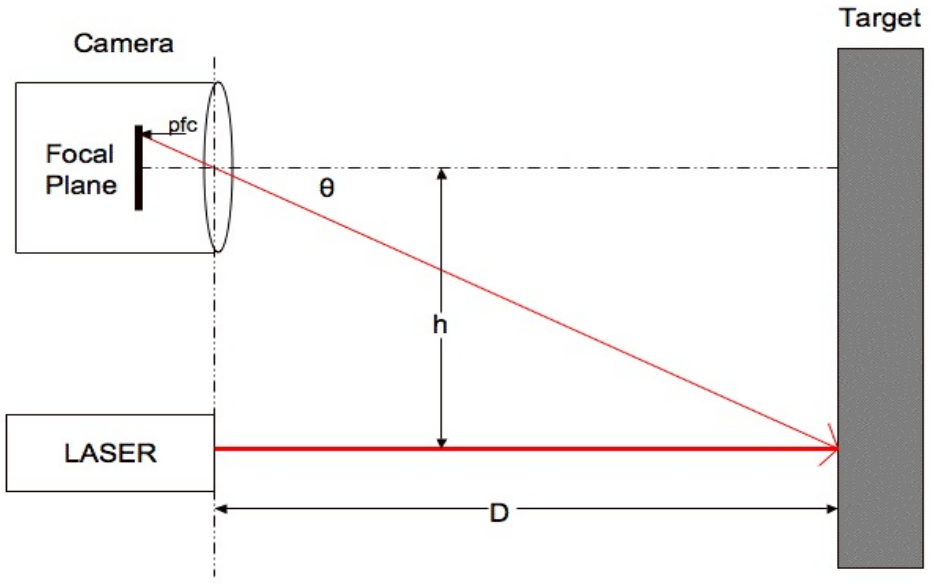

Figure 4.4: Vision Measurement [34]

A laser-beam in figure 4.6 is projected onto an object in the field of view of a camera [34]. This laser beam is ideally parallel to the optical axis of the camera. The dot from the laser is captured along with the rest of the scene by the camera. The dots position in the image frame is known. Then the range to the object is calculated based on where along the y axis of the image this laser dot falls. Distance is calculated with this equation.

$$D = \frac{h}{tan\Theta} \tag{4.9}$$

$$D = \frac{h}{tan(pfc * rpc * ro)} \tag{4.10}$$

pfc : Number of pixels from center of plane

rpc : Radians per pixels pitch

ro : Radian offset

In this thesis, laser rangefinder has been tried but it can not be succeeded. The laser pointer is red but according to the camera view, it is very bright. The pixel where the laser dot in the image seems like white pixel. For running environment has lots of white color, the vision measurements with laser rangefinder have lots of errors.

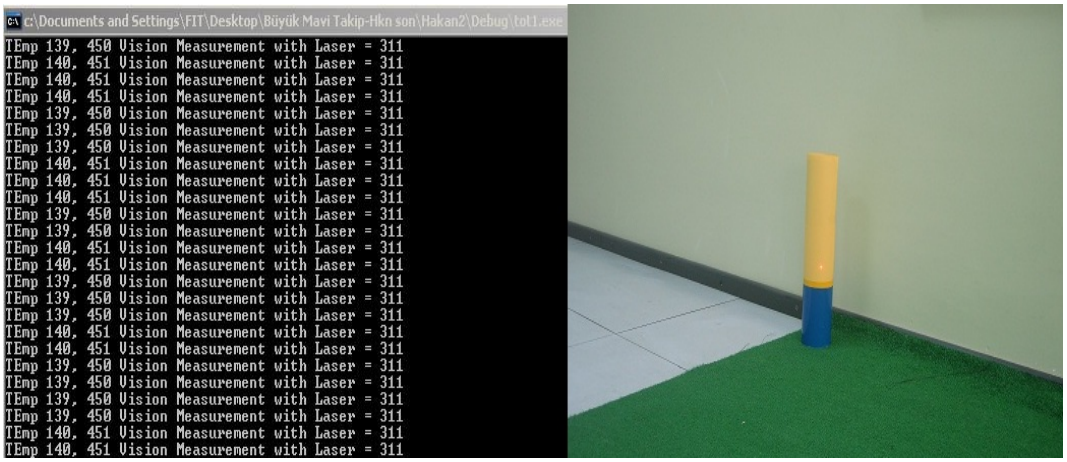

Figure 4.5: Laser Rangefinder

4.2.3 Finding the Distance Depending on the Size of the Object

The distance between the vehicle and object is computed depending on the size of the object. Firstly, the object is detected in an image which is taken from the camera. Then, the object's center of mass is calculated. This is followed by the calculation of the number of pixels stating from the object's center of mass to the top and bottom of the object. As shown in Table 4.2, the exact distance corresponds to this pixel count.

Table 4.2: Calibration between all object pixels and exact distance

Pixel	Exact Measurement(cm)
263	45
250	50
239	55
227	60
217	65
207	70
201	75
194	80
188	85
183	90
136	180
131	200
126	220
121	250

After finding the distances, Matlab Curve Fitting tool the following equation has been obtained. The distance is calculated with the equation of which has been found,

$$distance = 1.662 + 004 * e^{(-0.04082 * pixels)} + 326.5 * e^{(-0.007601 * pixels)} \tag{4.11}$$

distance : distance between the car and object

pixels : Pixel count of the object.

While using this algorithm, the most important point to consider is that the height of the camera needs to be longer than the length of the object. Another important point is that all object must be in the same size and color. Furthermore objects must have the same shape from all the way around. As we have used fixed landmarks, we haven't met any difficulties.

In this project, this algorithm is used because it gives the closest result to the exact-measured- distances. Figure 4.6 illustrates the process of range measurement algorithm.

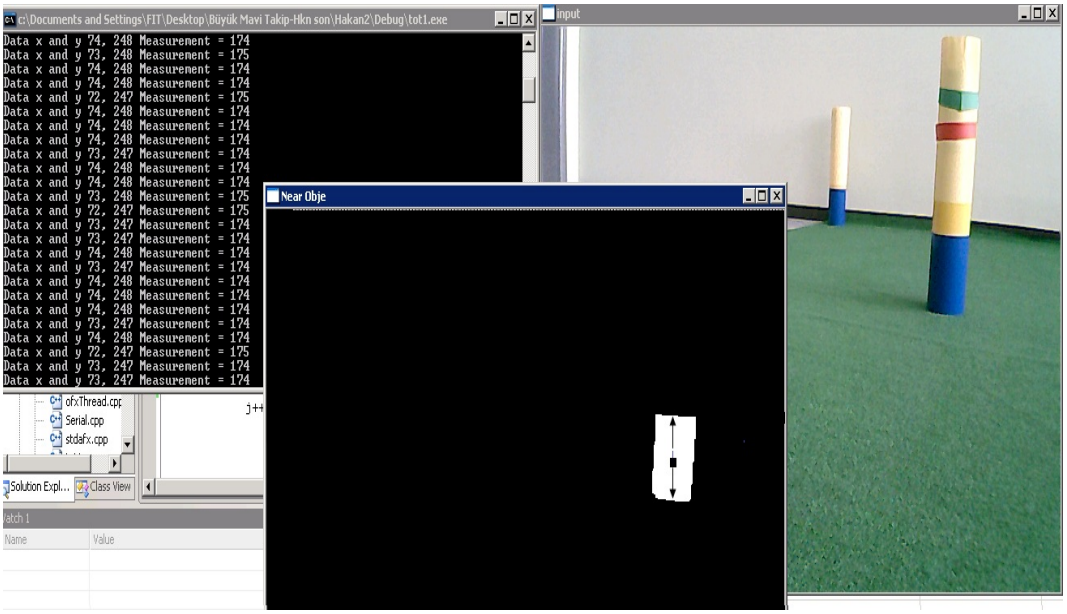

Figure 4.6: Vision Measurement

4.3 Kalman Filter (KF)

A Kalman Filter [1, 2] is simply an optimal data processing algorithm. The Kalman Filter is used to estimate the position of the robot from odometry data and landmark observations, there are three steps in the SLAM process:

1. Update the current state estimate using the odometry data.
2. Update the estimated state from re-observing landmarks.
3. Kalman Gain.

The Kalman filter uses the history of measurements to build a model of the state of the system that maximizes the probability for the position of the target based on the past measurements. First, recall the time discrete Kalman Filter equations and the predict-update equations as below:

Predict:

$$\widehat{x}_{t|t-1} = F_t \widehat{x}_{t-1|t-1} + B_t + u_t \tag{4.12}$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \tag{4.13}$$

Update:

$$x_{t|t} = \widehat{x}_{t|t-1} + K_t(y_t - H_t \widehat{x}_{t|t-1}) \tag{4.14}$$

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \tag{4.15}$$

$$P_{t|t} = (I - K_t + H_t) P_{t|t-1} \tag{4.16}$$

$\widehat{x}$: Estimated state

F : State transition matrix

u : Control variables

B : Control matrix

P : State variance matrix

Q : Process variance matrix

y : Measurement variables

H : Measurement matrix

K : Kalman gain

R : Measurement variance matrix

4.3.1 The Kalman Gain

The Kalman gain (KG) is computed to detect the degree of reliability of the observed landmarks and the new knowledge that we get from them. Under the

condition that the robot should be moved 2 cm to the right, the Kalman Gain is used to learn how much we actually correct the position according to the landmarks. We do not trust the landmarks completely but instead find a compromise between the odometer and the landmark correction. This is done by employing the uncertainty of the observed landmarks accompanied by a measure of the quality of the vision measurement and the odometer data of the car.

Chapter 5

Experiments and Results

The following passage describes the experiments carried out to determine the closest object. This was done under vastly different conditions.

- Three landmarks, all having the same colors
- Three landmarks, each having different colors
- Three objects of any shape and color

For the first two cases, as long as the objects are within the visible range, the car has successfully identified the closest landmark, or object in general and moved towards it. However, for the last case, as the algorithm we have used requires all objects to be of the same size, we could neither detected the closest object nor measured the distance to any object. If the objects are out of scene, the car performs a random walk until there is an object in the visible area.

5.1 Experiments and Results for Distance Measurement

The following graph shows the calibration of the camera, and is used while computing the distances to the landmarks (See Section 4.2.3).

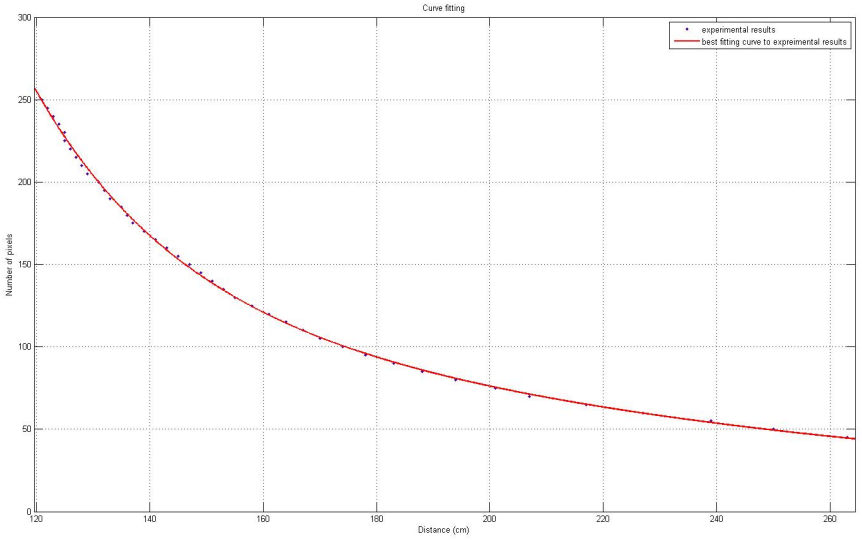

Figure 5.1: Vision Measurement vs. Actual Distance (cm) Graph

5.2 Experiments and Results for SLAM

Tables 5.1 and 5.2 illustrates the results of our study. The results in Table 5.1 shows that there is no rotation of the car. It includes the position of the autonomous vehicle with respect to odometer data and landmark measurement. It also includes the position of the autonomous vehicle after Kalman filtering is applied with its real position.

Table 5.1: SLAM Table without Rotation

Time (milis)	C. P. (y,x)	Velocity (m/sec,degree)	L. M. (cm)	L. Pre. (cm)	R. M. (cm)	R. P. (y,x)	K. Pre. (y,x)
0	(0,0)	0,0	336.54	336	336	(0,0)	(0,0)
187	(19,0)	1.02,0	320.62	317	313	(23,0)	(16.9,0)
391	(59,0)	1.02,0	280.37	277	290	(46,0)	(41.4,0)
390	(99,0)	1.02,0	258.08	237	266	(70,0)	(65.5,0)
390	(139,0)	1.02,0	223.45	197	230	(106,0)	(102.2,0)
375	(177,0)	1.02,0	204.05	159	196	(140,0)	(134,0)
375	(215,0)	1.02,0	136.26	121	153	(183,0)	(178.5,0)
391	(262,0)	97,0	77.04	89	107	(229,0)	(224.7,0)
...	...	...	...	...	...	...	...

C. P. : Car Position

L. M. : Landmark Measurement

L. Pre. : Landmark Prediction

R. M. : Real Measurement

R. P. : Real Position

K. Pre. : Kalman Prediction

Figure 5.2 and 5.3 shows the position of the autonomous vehicle with respect to odometer data, its real position and the position of the autonomous vehicle after Kalman filtering is applied by simulating.

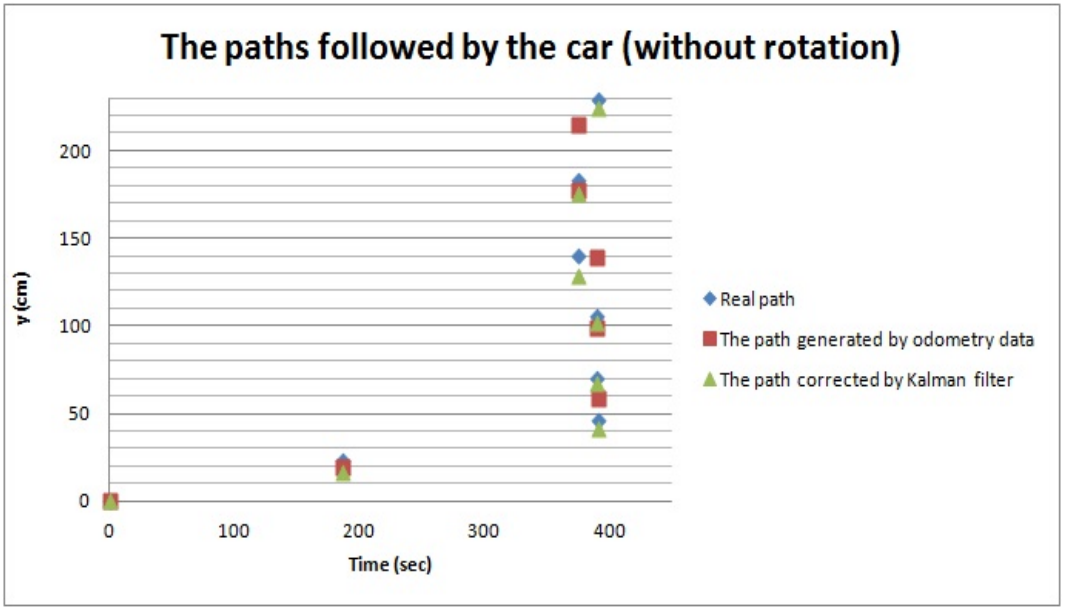

Figure 5.2: The paths followed by the car (without rotation)

In Table 5.2, the car rotates the nearest of the landmark. While the autonomous vehicle is moving towards the nearest landmark, it shows the position of the autonomous vehicle with respect to odometer data and landmark measurement. It also shows the position of the autonomous vehicle after Kalman filtering is applied with its real position.

Table 5.2: SLAM Table with Rotation

Time (milis)	C. P. (y,x)	Velocity (m/sec,deg)	L. M. (y,x)	L. Pre. (y,x)	R. M. (y,x)	R. P. (y,x)	K. Pre. (y,x)
0	(0,0)	0,0	340,28	340,28	340,28	(0,0)	(0,0)
187	(19,1)	1.02,3	318,25	321,27	326,24	(14,4)	(16.9,1.2)
391	(55,17)	1.02,24	293,22	285,11	299,16	(41,12)	(45.2,7.9)
190	(74,20)	1.02,9	274,21	266,8	277,16	(63,12)	(66.7, 8.4)
240	(98,23)	1.02,9	249,16	242,5	256,14	(84,14)	(86.2,9.14)
375	(134,36)	1.02,20	182,7	206,-8	220,9	(120,19)	(123.8,20.1)
495	(175,36)	1.02,0	169,5	165,-8	172,6	(168,22)	(172,23.4)
391	(215,36)	1.02,0	131,2	125,-8	112,4	(228,24)	(225.9,26.1)
...	...	...	...	...	...	...	...

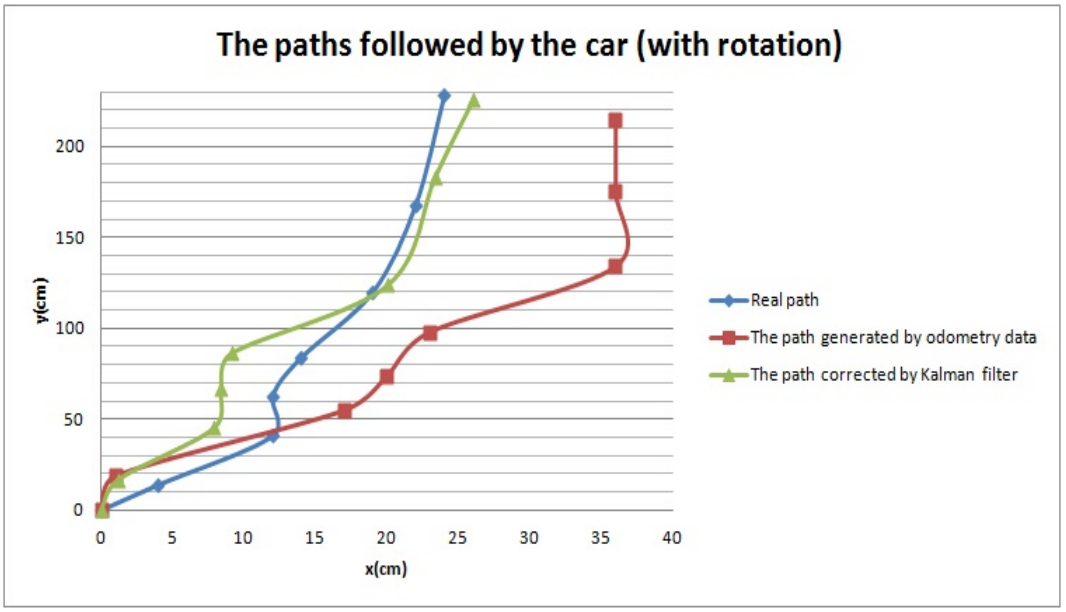

Figure 5.3: The paths followed by the car (with rotation)

Table 5.3 shows the error rate between the real position and the position according to odometer data and the position of the autonomous vehicle after Kalman filtering is applied and its real position.

Table 5.3: SLAM Table Error Rate without Rotation

C. P. (y,x)	R. P. (y,x)	K. P. (y,x)	Error P.o.C and R.P. (cm)	Error R.P. and K. Pre. (cm)
0,0	0,0	0,0	0	0
19,0	23,0	16.9,0	4.0	6.1
59,0	46,0	41.4,0	13.0	4.6
99,0	70,0	65.5,0	29.0	4.5
139,0	106,0	102.2,0	33.0	3.8
177,0	140,0	134,0	37.0	6.0
215,0	183,0	178.5,0	32.0	4.5
262,0	229,0	224.7,0	33.0	4.3
...	...	...	...	...

Table 5.4 shows the error rate between the real position and the position according to odometer data and the position of the autonomous vehicle after Kalman filtering is applied and its real position when the autonomous car moves towards to the closest object.

Table 5.4: SLAM Table Error Rate with Rotation

C. P. (y,x)	R. P. (y,x)	K. P. (y,x)	Error P.o.C and R.P. (cm)	Error R.P. and K. Pre. (cm)
0,0	0,0	0,0	0	0
19,1	14,4	16.9,1.2	4	4.0311
55,17	41,12	45.2,7.9	13	5.8694
74,20	63,12	66.7, 8.4	29	5.5624
98,23	84,14	86.2,9.14	33	5.3348
134,36	120,19	124.8,20.1	37	4.9244
175,36	168,22	172,23.4	32	4.5352
215,36	228,24	225.9,26.1	33	2.9698
...	...	...	...	...

After ten or more step, the error of the ratio will decrease; it means that, the Kalman predicted values get closer to the real measurement.

Conclusion

This study investigates transforming a remote controlled RC car into an autonomous car that is able to find its own location and mapping at the same time. The first part of the thesis explores how autonomous car navigates towards a target object in indoor environment. Setting a start and end point for the vehicle, it is observed to reach at the end point. As the next step, it is aimed to make the vehicle to go to the nearest object. After a number of experiments, it is seen the vehicle could fulfill the task in terms of going to the nearest object. While performing both tasks, if the car perceives an object in front, it goes to the nearest object. Then it reaches at the end point. These encouraging results show that the vehicle in our study could navigate safely in a rough unstructured environment.

One significant issue that needs to be studied in the future is whether the algorithm that is presented in this thesis to find the nearest object can be implemented to the shortest path algorithm. Another issue for further study is to find out whether it would yield more reliable measurements if laser sensor was used instead of ultrasonic sensor.

Concerning the main aim of the study, which was stated as constructing an autonomous vehicle which is able to locate itself and build the map of its unknown indoor environment. It can be concluded that using SLAM techniques we have succeeded in limited lab environment.

The experiment results show the inadequacy of odometer data and landmark measurement while the position of the autonomous vehicle was attempted to be found. Error rate of the calculations made by odometer is increasing in each step since calculations are prepared with respect to the previous erroneous position.

Although Odometer data has lots of noise, Kalman filtering minimize these noises. Therefore, as seen in the experiment results, we could gather very close results to real measurement by using Kalman Filter.

It is seen that the sensors were not sufficient to perceive all environmental features. If they were so, we could have gathered better results. Therefore, laser scan sensor can serve for more efficient and generic algorithms.

For future studies, as image processing filtering algorithms takes a considerable amount of time, there appears a difference between estimated and real distance. To eliminate this problem, laser scan can be used. Moreover, finding nearest object algorithm can be improved to find the shortest path. Finally, the quality of the camera would help us to yield more precise results.

References

[1] Kalman, R., "A New Approach to Linear Filtering and Prediction Problems", *Transactions of the ASME Journal of Basic Engineering*, , No. 82 (Series D), pp. 35–45,, 1960.

[2] Maybeck, P., *Stochastic Models, Estimation and Control. Volume I.*, 1979.

[3] Shangming, W., "Smooth Path Planning and Control for Mobile Robots", 2005.

[4] Harris, C. and D. Charnley, "Intelligent Autonomous Vehicles Recent Progress and Central Research Issues", *Computing & Control Engineering Journal*, pp. 164—171, July 1992.

[5] Dickmanns, E. D., "Vision for Ground Vehicles History and Prospects", *International Journal of Vehicle Autonomous Systems*, August 2003.

[6] DARPA, "What is the Urban Challenge", june 2008.

[7] Hugh, F. and W. Durrant, "Sensor Models and Multisensor Integration", *I. J. Robotic Res,.*

[8] Brady, M., "Special Issue on Sensor Data Fusion", *I. J. Robotic Res.*, December.

[9] Micha, S., "Algorithmic Motion Planning in Robotics", *Computer*, Vol. 22, pp. 9–20,, 1989.

[10] Leonard, A. and J. Ingemar, "Dynamic Map Building for an Autonomous Mobile Robot", *Int. J. Rob. Res.*, pp. 286–298, 1992.

[11] Elfes, A., "Sonar-Based Real-World Mapping and Navigation", 1990.

[12] Cox, I., "Autonomous Robot Vehicles", *Springer-Verlag Inc.*, 1990.

[13] Dickmanns, E. and A. Zapp, "Guiding Land Vehicles along Roadways by Computer Vision", *AFCET Conference*, p. 244, October 1985.

[14] Moore, C. G. and C. J. Harris, "Intelligent identification and control for AGV's Using Adaptive Fuzzy Based Algorithms", *Int. J. of Eng. Applications of AI*, Vol. 2, pp. 267–285, 1992.

[15] Brooks, R. A., "A robust Layered Control System for a Mobile Robot", *International Conference on Robotics and Automation*, 1985.

[16] S. Corfield, R. F. and C. Harris, "Architecture for Real-time Intelligent Control of Autonomous Vehicles", *Computing and Control Engineering Journal*, pp. 254–262, November 1991.

[17] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments", *International Conference on Robotics and Automation*, pp. 3310–3317, 1994.

[18] Fuhao, Z. and L. Jiping, "An Algorithm of Shortest Path Based on Dijkstra for Huge Data", *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 4*, pp. 244–247, IEEE Press, 2009.

[19] Dolgov, D. and J. Diebel, "Practical Search Techniques in Path Planning for Autonomous Driving", November 2007.

[20] Imielinski, T. and C. Navas, "GPS-Based Addressing and Routing", Technical report, 1996.

[21] Pitas, I., *Digital Image Processing Algorithms and Applications*, John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000.

[22] Wang, M., *Concise Introduction to Image Processing Using C++*, crc press edition, 2009.

[23] Gonzalez, R. and R. Woods, *Digital Image Processing (3rd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[24] Tek, B., "CSE 487 Computer Vision Lecture Notes,2", Technical report, Işık University, 2010.

[25] Julier, S. and K. Uhlmann, "A General Method for Approximating Nonlinear Transformations of Probability Distributions", Technical report, 1996.

[26] Gonzalez, R. and R. Woods, *Digital Image Processing*, Prentice-Hall, Inc., 2006.

[27] Shapiro, L. and G. Stockman, *Computer Vision*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2001.

[28] Tek, B., "CSE 487 Computer Vision Lecture Notes,3", Technical report, Işık University, 2010.

[29] Comer, M. and E. Delp, "Morphological Operations for Color Image Processing", *Journal of Electronic Imaging*, , No. 3, pp. 279–289, July 1999.

[30] Aksoy, S., *Computer Vision*, Bilkent University, 2010.

[31] Leonard, J. and W. Durrant, "Mobile Robot Localization by Tracking Geometric Beacons", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 376–382, 1991.

[32] R. Smith, P. C., *Estimating Uncertain Spatial Relationships in Robotics*, pp. 167–193, Springer-Verlag New York, Inc.,, New York, NY, USA,, 1990.

[33] Taylor, T. and W. Boles, "Monocular Vision as a Range Sensor", *CIMCA 2004 Proc*, pp. 566–575, 2004.

[34] Danko, T., "Webcam Based DIY Laser Rangfinder", 2004.

Curriculum Vitae