DESIGN OF A SOFTWARE BASED PLATFORM OF AN INTELLIGENT, AUTONOMOUS, MULTISENSOR ROBOCAR

By

Muhammad Ossama Khalid

Hassan Zafar Gill

Muhammad Faisal Elahi

Muhammad Taimur Anjum

Submitted to the Faculty of Electrical Engineering, Military College of Signals, National University of Sciences and Technology, Rawalpindi in partial fulfillment for the requirements of a B.E Degree in Telecom Engineering

AUGUST 2010

ABSTRACT

The Robocar is an autonomous car that combines obstacle avoidance with path tracking capability all in one package. The Robocar patrols a given target track while avoiding obstacles on its way. Once an obstacle is detected, the Robocar is reversed for a specific time and changes its direction and moves forward .All the processing is done in real time on a remote computer using OPENCV.

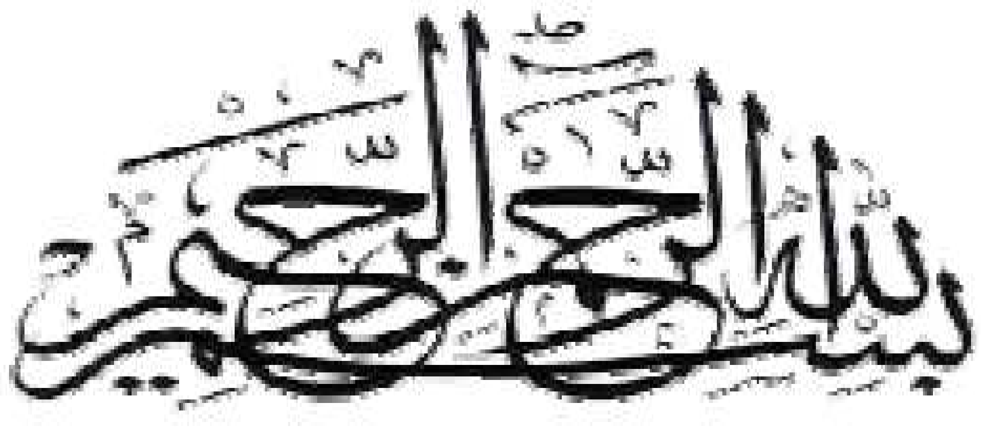
The vehicle is equipped with ultra-sonic sensor that scans the path for the obstacles on its way and a wireless CCTV camera for path tracking .The on-board electronics are controlled by a PIC microcontroller .The Robocar utilizes sonar for measuring the distance of obstacle thus providing obstacle avoidance. It includes the design and development of a cost-effective, reliable, un-manned and intelligent obstacle detection system that detects dangers on the road and suggests a course of action. The system qualifies path-tracking, obstacle detection and classification and in-time decision making.

CERTIFICATE

Certified that the contents and form of project report entitled **“Design of a software based platform of an intelligent, autonomous, multisensor Robocar”** submitted by 1) Muhammad Faisal Elahi, 2) Muhammad Taimur Anjum, 3) Muhammad Ossama Khalid and 4) Hassan Zafar Gill have been found satisfactory for the requirement of the degree.

Supervisor: ____________________

Lecturer Safwat Irteza Butt



DEDICATION

All our efforts are dedicated to our beloved parents and teachers who have been a constant source of encouragement for us throughout our lives. May Allah Almighty bless them with long lives and always provide us their loving and sincere guidance.

ACKNOWLEDGEMENTS

We would like to take this opportunity to pay our humble gratitude to Almighty Allah (S.W.A.T), the most merciful, who blessed us with His divine help in the completion of our project. We are profoundly thankful to our project supervisor, Lecturer Safwat Irteza Butt, for his unparalleled help and motivation, , the college staff for their assistance, and our friends and families for their support and patience throughout the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW

Automotive safety has taken a back seat recently due to the rise in fuel prices focusing more attention on economy. But safety remains an important factor for both consumers and suppliers around the world. Vehicle crashes continue to account for a large number of lives lost each year, despite considerable improvements in occupant protection. Nowadays automotive owners are planning to inculcate new features of auto park and auto driver in the vehicles to facilitate the drivers to drive in an unfavorable atmosphere like fog, low light conditions etc, thus avoiding the risk of accidents. In robotics, obstacle avoidance is the task of satisfying some control objective subject to non-intersection or non-collision position constraints. Normally obstacle avoidance is considered to be distinct from path planning as it is usually implemented as a reactive control law while the other involves the pre-computation of an obstacle-free path on which a controller will then guide a robot along. The Robocar uses multiple sensors for path detection and obstacle avoidance thus making it an intelligent and an autonomous car.

1.2 PROBLEM STATEMENT

No single technique is accurate enough to carry out the functionalities of path tracking and obstacle avoidance. These functionalities can only be attained through an amalgam of multiple techniques. Here the system demands a cost-effective, reliable, un-manned and intelligent obstacle detection system that will detect dangers on the road and suggest a course of action. For the said purpose, it should qualify path-tracking, obstacle detection, relative motion of the obstruction and in-time decision making. We will employ the use of various sensors which will locate the obstacle like ultrasonic sensor (SONAR) and camera-based path tracking. Furthermore, it will involve the application of signal processing and its fast implementation.

1.3 OBJECTIVES

- The aim of the project is to use multiple techniques so as to formulate an effective obstacle detection system that can detect various road obstructions and thus, can avoid them efficiently.
- The system will analyze the data from the sensors used and determine a safe path of travel to follow avoiding collisions.

1.4 PROJECT STAGES

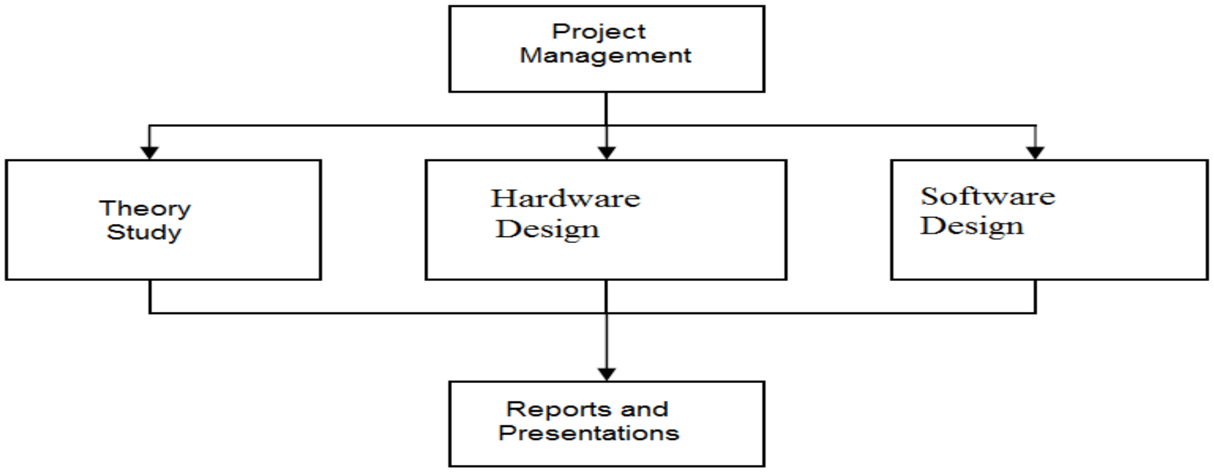


Figure 1.4 Project breakdown structure

CHAPTER-2

PROJECT BACKGROUND

2.1 MOTIVATION

Much interest has been devoted to environment sensors in the field of autonomous driving systems e.g. RADAR, SONAR, LIDAR or video based sensors. Following several decades of research and development in autonomous vehicle guidance, we are now witnessing introduction of driver assistance functions into passenger cars among which most of the functions are based upon sensors measuring the status of the vehicle itself.

There is no single sensor system available which has the capability to fulfill all given requirements. Therefore, a combination of very capable sensor systems which should detect all obstacles surrounding the vehicle is considered. The obstacle detection has to be performed with high reliability to avoid any potential collision in the longitudinal (forward) direction. For this reason, a multi-sensor system containing technologies mentioned above are under investigation for this challenging application of making an autonomous driving system.

The ability to acquire, integrate, and interpret multi-sensory information to generate appropriate actions in the performance of a given task is essential for any realistic autonomous robotic application. Here the key to effective performance of such an application is the design of the sensor system. Thus, in a nutshell, a system is required that adds independence to the driver-assisted vehicle movement, therefore rendering it autonomous and self sufficient.

2.2 THE ORIGIN OF ROBOTS

The word robot has its origins in a Czech word “robotnik” meaning worker or serf. The history of robots can be traced back to AL. 770, when the Swiss clock maker Pierre Jacquet-Droz created three mechanical dolls that could play music on an organ, draw simple pictures, and write. Another name that cannot be omitted from the history of robotics is Nicola Tesla, who built a radio-controlled submarine. Mechanization and automation can be traced back to the industrial revolution.

The theme of robots was first introduced by the playwright Karel Capek in a 1920 play. It was popularized by Isaac Asimov in science fiction in the late 1940’s and the early 1950’s.

In December 1996, Honda demonstrated the Honda Humanoid, a robot with two legs and two arms that is designed for use in a typical domestic environment. The 210 kilograms prototype has 30 degrees of freedom. It is able to walk around, climb a flight of stairs, sit down on a chair, stand up from a sitting position and lift payloads of 10 lbs.

2.3 SENSORS

A sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. For example, a mercury-in-glass thermometer converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube.

A thermocouple converts temperature to an output voltage which can be read by a voltmeter. For accuracy, all sensors need to be calibrated against known standards. There exist a wide variety of sensors that a robot could use. Some are simple, such as bump sensors, while others are very complex, such as vision sensors. The type of sensors used on a robot depends on tasks the robot must perform, available electrical and computational power, and carrying capacity. In robotics we have different types of sensors like

- Infrared sensors
- Ultrasonic sensors
- LADAR
- RADAR
- LIDAR
- Optical sensors

2.3.1 INFRARED SENSORS

Infrared sensors are very common, and can be configured in numerous ways. One common setup is a break beam sensor, where an IR emitter and sensor face each other separated by a gap. When an object comes between the sensor and emitter, a signal is produced. Another typical setup has the sensor and emitter facing the same direction. The IR produced from the emitter bounces off objects and is detected by the sensor. The time difference between emission and detection is used to calculate the distance to the object that caused the reflection.

2.3.2 ULTARSONIC SENSORS

Ultrasonic sensors or SONARs are another type of sensor commonly found on robots. The same principles used in an infrared (IR) range sensor apply to a sonar sensor. Sound waves are emitted from the sonar, and then detected after they bounce off objects in the environment. The time difference between emission and detection is used to determine the distance to an object. Sonar sensors typically have a longer range and wider field of view than IR, but suffer from specular reflection.

2.3.3 LADAR

Laser Radar (LADAR) is employed similar to millimeter wave radar, but uses laser beams to scan and process the signal echoed from targets in order to create a virtual picture of the area. The LADAR processor looks for familiar patterns in the scenes. The processor continuously compares these patterns with 3D target files stored in the weapon's memory. Due to its capability to scan large areas with very high precision and its ability to gradually build a detailed picture of the area under surveillance, LADAR sensors are usually employed on loitering systems, which can look at the target from different angles, verify the target identification and select the best attack position for the desired effect.

2.3.4 RADAR

A radar system has a transmitter that emits radio waves. When they come into contact with an object they are scattered in all directions. The signal is thus partly reflected back and it has a slight change of wavelength (and thus frequency as well) if the target is moving. The receiver is usually, but not always, in the same location as the transmitter. Although the signal returned is usually very weak, the signal can be amplified through use of electronic techniques in the receiver and in the antenna configuration. This enables radar to detect objects at ranges where other emissions, such as sound or visible light, would be too weak to detect. Radar uses include meteorological detection of precipitation, measuring ocean surface waves, air traffic control, police detection of speeding traffic, military applications, or to simply determine the speed of a baseball.

2.3.5 LIDAR

LIDAR (Light Detection and Ranging) is an optical remote sensing technology that measures properties of scattered light to find range and other information of a distant target. The prevalent method to determine distance to an object or surface is to use laser pulses. Like the similar RADAR technology, which uses radio waves, the range to an object is determined by measuring the time delay between transmission of a pulse and detection of the reflected signal. The primary difference between LIDAR and radar is that LIDAR uses much shorter wavelengths of the electromagnetic spectrum, typically in the ultraviolet, visible, or near infrared range. In general it is possible to image a feature or object only about the same size as the wavelength, or larger. Thus LIDAR is highly sensitive to aerosols and cloud particles and has many applications in atmospheric research and meteorology.

2.3.6 OPTICAL SENSORS

Cameras offer a significant source of data for any robot. Images are filled with useful information such as color, texture, movement and 3D data when using stereo vision. However extracting this information is computationally expensive. The camera sends images to computer which performs the computations and provides the robot with information for which it is designed.

2.4 THE ROBOCAR

The Robocar (Figure 2.4) is an intelligent car that can avoid obstacles while following its path autonomously. It transmits a video signal in a manner that suits the transmission over a wireless connection to a computer port, enabling the computer to receive and display this video signal using a software program. In addition, some control signals are transferred from the computer to the robot to control its motion and perform tasks. The characteristics of a robot vary with the source referenced. Generally, robots are intelligent and obedient but impersonal machines. A robot is any machine that does work on its own automatically after it is programmed by humans.

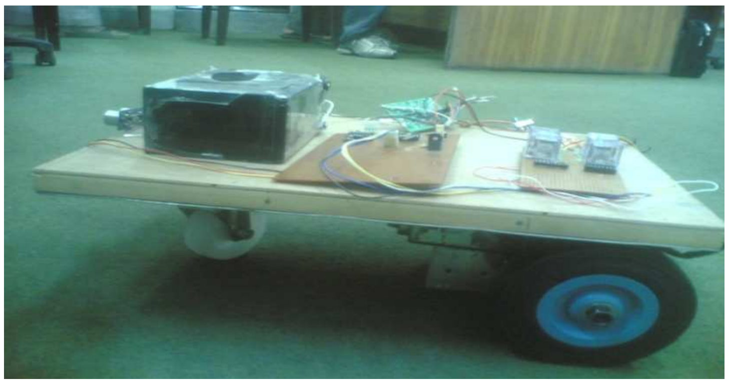


Figure 2.4 The Robocar

CHAPTER-3

OBSTACLE DETECTION

3.1 SONAR

Sonar (originally an acronym for Sound Navigation and Ranging) is a technique that uses sound propagation (usually underwater, as in Submarine navigation) to navigate, communicate with or detect other vessels. Two types of technologies share the name "sonar": passive sonar is essentially listening for the sound made by vessels; active sonar is emitting pulses of sounds and listening for echoes. Sonar may be used as a means of acoustic location and of measurement of the echo characteristics of targets in the water. Acoustic location in air was used before the introduction of radar. Sonar may also be used in air for robot navigation, and SODAR (an upward looking in-air sonar) is used for atmospheric investigations. The term sonar is also used for the equipment used to generate and receive the sound. The study of underwater sound is known as underwater acoustics or hydro acoustics. The acoustic frequencies used in sonar systems vary from very low (infrasonic) to extremely high (ultrasonic).

3.2 HISTORY

Although some animals like dolphins and bats have used sound for communication and object detection for millions of years, use by humans in the water is initially recorded by Leonardo Da Vinci in 1490: a tube inserted into the water was said to be used to detect vessels by placing an ear to the tube. In the 19th century an underwater bell was used as an ancillary to lighthouses to provide warning of hazards.

Canadian Reginald Fessenden, while working for the Submarine Signal Company in Boston, built an experimental system beginning in 1912. In that test he demonstrated depth sounding, underwater communications (Morse code) and echo ranging (detecting an iceberg at two miles range. During World War I, the need to detect submarines prompted more research into the use of sound.

The British made early use of underwater hydrophones, while the French worked on the development of active sound devices for detecting submarines in 1915 using quartz. Lightweight sound-sensitive plastic film and fibre optics have been used for hydrophones (acousto-electric transducers for in-water use), while Terfenol-D and PMN (lead magnesium niobate) have been developed for projectors.

3.3 TYPES

A major step in the development of sonar systems was the invention of the acoustic transducer and the design of efficient acoustic projectors. Sonar systems may be broadly classified into three groups.

- Active sonar systems.
- Passive sonar systems.
- Acoustic communication systems.

3.3.1 ACTIVE SONAR SYSTEMS

In active sonar systems an acoustic projector generates a sound wave that spreads outward and is reflected back by a target object. A receiver picks up and analyzes the received signal and may determine the range. Active sonar uses a sound transmitter and a receiver. When the two are in the same place it is mono-static operation. When the transmitter and receiver are separated it is bi-static operation. When more transmitters or more receivers are used, again spatially separated, it is multi-static operation. Most SONARs are used mono-statically with the same array often being used for transmission and reception. Active sonobuoy fields may be operated multi-statically. Active sonar creates a pulse of sound, often called a "ping", and then listens for reflections (echo) of the pulse. This pulse of sound is generally created electronically using a Sonar Projector consisting of a signal generator, power amplifier and electro-acoustic transducer array, possibly with a beam former. To measure the distance to an object the time from transmission of a pulse to reception is measured and converted into a range by knowing the speed of sound. To measure the bearing several hydrophones are used and the set measures the relative arrival time to each, or with an array of hydrophones, by measuring the relative amplitude in beams formed through beam forming.

3.3.2 PASSIVE SONAR SYSTEMS

Passive systems consist simply of receiving sensors that pick up the noise produced by the target (such as a ship, submarine, or torpedo). Waveforms thus detected may be analyzed for identifying characteristics as well as direction and distance. Passive sonar listens without transmitting. So a switch from active to passive SONARs that detected and tracked submarines by mere listening was made. Passive sonar has a wide variety of techniques for identifying the source of a detected sound. For example, U.S.vessels usually operates 60 Hz alternating current power systems. If transformers or generators are mounted without proper vibration insulation from the hull, the 60 Hz sound from the windings can be emitted from the submarine or ship. This can help to identify its nationality, as most European submarines have 50Hz power systems. Intermittent sound sources (such as a wrench being dropped) may also be detectable to passive sonar. Until fairly recently, an experienced trained operator identified signals, but now computers may do this. The elimination of noises on board of submarines made them so silent that passive detection became unlikely before the submarine opened fire with its torpedoes.

3.3.3 ACOUSTIC COMMUNICATION SYSTEMS

The third category of sonar devices is acoustic communication systems, which require a projector and receiver at both ends of the acoustic path. Acoustic communication provides an attractive solution for continuous service of wireless sensor networks. Current prevailing sensor boards in use provide the basic hardware support for acoustic communication. For instance, the Mica sensor board has a sounder and a tone detector. The sounder is capable of making sound of certain pitch, which can be detected by the tone detector.

3.4 PERFORMANCE FACTORS

The detection, classification and localization performance of a sonar depends on the environment and the receiving equipment, as well as the transmitting equipment in an active sonar or the target radiated noise in a passive sonar.

3.4.1 SOUND PROPAGATION

Sonar operation is affected by variations in sound speed, particularly in the vertical plane. Sound travels more slowly in fresh water than in sea water, though the difference is small. The speed is determined by the water's bulk modulus and mass density. The bulk modulus is affected by temperature, dissolved impurities (usually salinity), and pressure. The density effect is small. The speed of sound (in feet per second) is approximately:

v= 4388 + (11.25 × temperature (in °F)) + (0.0182 × depth (in feet)) + salinity (in parts-per-thousand).

This empirically derived approximation equation is reasonably accurate for normal temperatures, concentrations of salinity and the range of most ocean depths. Water pressure also affects sound propagation: higher pressure increases the sound speed, which causes the sound waves to refract away from the area of higher sound speed. The mathematical model of refraction is called Snell's law. If the sound source is deep and the conditions are right, propagation may occur in the 'deep sound channel'. This provides extremely low propagation loss to a receiver in the channel. This is because of sound trapping in the channel with no losses at the boundaries. Sound propagation is affected by absorption in the water itself as well as at the surface and bottom. This absorption depends upon frequency, with several different mechanisms in sea water. Long-range sonar uses low frequencies to minimize absorption effects. The main noise sources are waves and shipping. The motion of the receiver through the water can also cause speed-dependent low frequency noise.

3.4.2 SCATTERING

When active sonar is used, scattering occurs from small objects in the sea as well as from the bottom and surface. This can be a major source of interference. This acoustic scattering is analogous to the scattering of the light from a car's headlights in fog: a high-intensity pencil beam will penetrate the fog to some extent, but broader-beam headlights emit much light in unwanted directions, much of which is scattered back to the observer, overwhelming that reflected from the target ("white-out"). For analogous reasons active sonar needs to transmit in a narrow beam to minimize scattering.

3.4.3 TARGET CHARACTERISTICS

The sound reflection characteristics of the target of the active sonar, such as a submarine, are known as its target strength. A complication is that echoes are also obtained from other objects in the sea such as whales, wakes, schools of fish and rocks. Passive sonar detects the target's radiated noise characteristics. The radiated spectrum comprises a continuous spectrum of noise with peaks at certain frequencies which can be used for classification.

3.5 ROBOCAR SONAR – OPERATION AND PROCESSING

The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, SX or Propeller chip, requiring only one I/O pin. The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated as shown in Figure 3.5

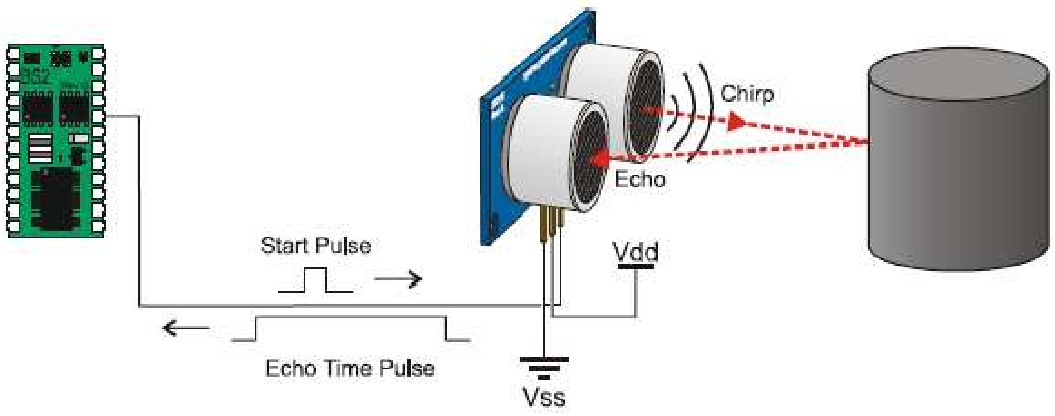


Figure 3.5 Working of Sonar

3.5.1 FEATURES

- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5V
- TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2µs Min, 5µs typ.
- Echo pulse: positive TTL pulse, 115µs Minimum to 18.5 ms maximum.

3.5.2 KEY SPECIFICATIONS

- Supply voltage: +5 VDC
- Supply current: 30mA type; 35mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1” spacing(ground, power, signal)
- Operating temperature: 0 – 70° C.
- Size: 22 mm H x 46 mm W x 16 mm D(0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

3.5.2 PIN DEFINITIONS

- GND Ground (Vss)
- 5 V 5 VDC (Vdd)
- SIG Signal (I/O pin)

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged directly into solder-less breadboard, or into a standard 3-wire extension cable. The dimensions of the sensor are shown in Figure 3.5.2

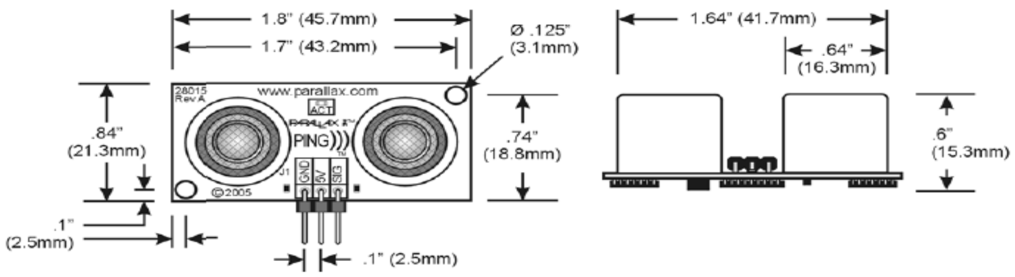


Figure 3.5.2 Dimensions of ping sensor

3.5.3 COMMUNICATION PROTOCOL

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected; hence the width of this pulse corresponds to the distance to the target. The echo sending and listening is shown in Figure 3.5.3

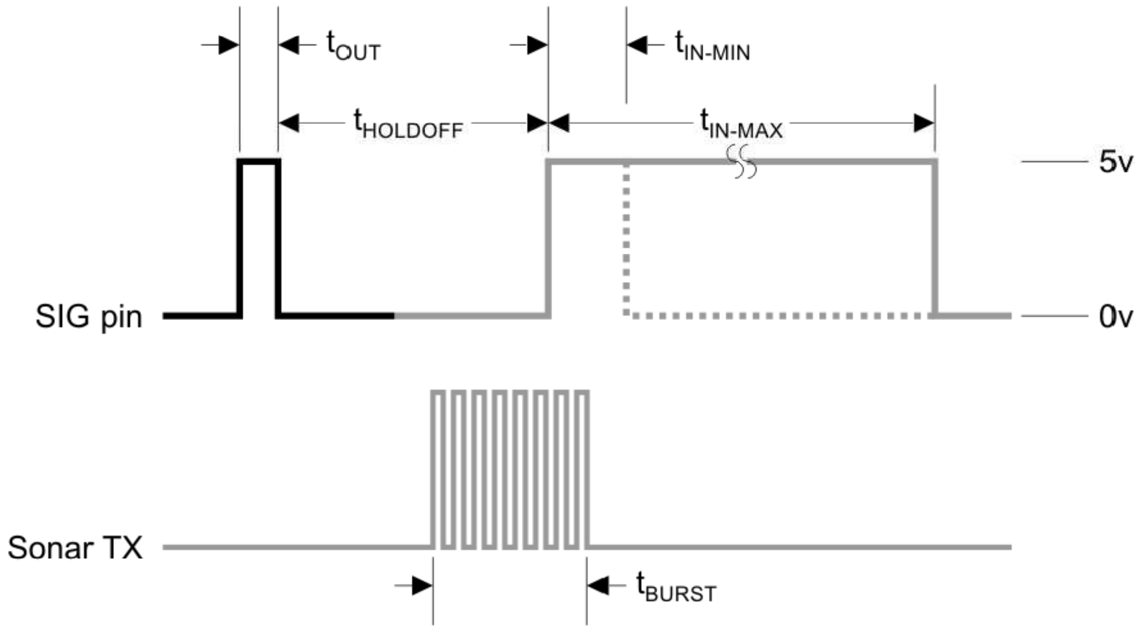


Figure 3.5.3 Echo sending and listening

CHAPTER-4

PATH DETECTION

4.1 OPEN CV

OpenCV stands for “open source computer vision”. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages. OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV is written in optimized C and can take advantage of multi-core processors. OpenCV automatically uses the appropriate Integrated Performance Primitives (IPP) library at runtime if that library is installed. All such transformations are done for achieving some particular goal. The input data may include some contextual information such as “the camera is mounted in a car” or “laser range finder indicates an object is 1 meter away”. The decision might be “there is a person in this scene” or “there are 14 tumor cells on this slide”.Typically, after the development of the first-draft solution, one can see where the solution has weaknesses and then fix those weaknesses using your own code and cleverness better known as solve solution as a benchmark to assess the improvements you have made. From that point, whatever weaknesses remain can be tackled by exploiting the context of the larger system in which your problem solution is embedded.

4.1.1 THE ORIGIN OF OPENCV

OpenCV grew out of an Intel Research initiative to advance CPU-intensive applications. Toward this end, Intel launched many projects including real-time ray tracing and 3D display walls. One of the authors working for Intel at that time was visiting universities and noticed that some top university groups, such as the MIT Media Lab, had well developed and internally open computer vision infrastructures-code that was passed from student to student and that gave each new student a valuable head start in developing his or her own vision application. Instead of reinventing the basic functions from scratch, a new student could begin by building on top of what came before. There were several goals for OpenCV at the outset: Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.

• Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

• Advance vision-based commercial applications by making portable, performance optimized code available for free, with a license that did not require commercial applications to be open or free themselves.

Those goals constitute the "why" of OpenCV. Enabling computer vision applications would increase the need for fast processors. Driving upgrades to faster processors would generate more income for Intel than selling some extra software. Along the way, OpenCV was affected by the dot-com boom and bust and also by numerous changes of management and direction.OpenCV is an active area of development at several institutions, so expect to see many updates in multi camera calibration, depth perception, methods for mixing vision with laser range finders, and better pattern recognition as well as a lot of support for robotic vision needs. Figure 4.1.1 shows the timeline of development of OpenCV at different stages.

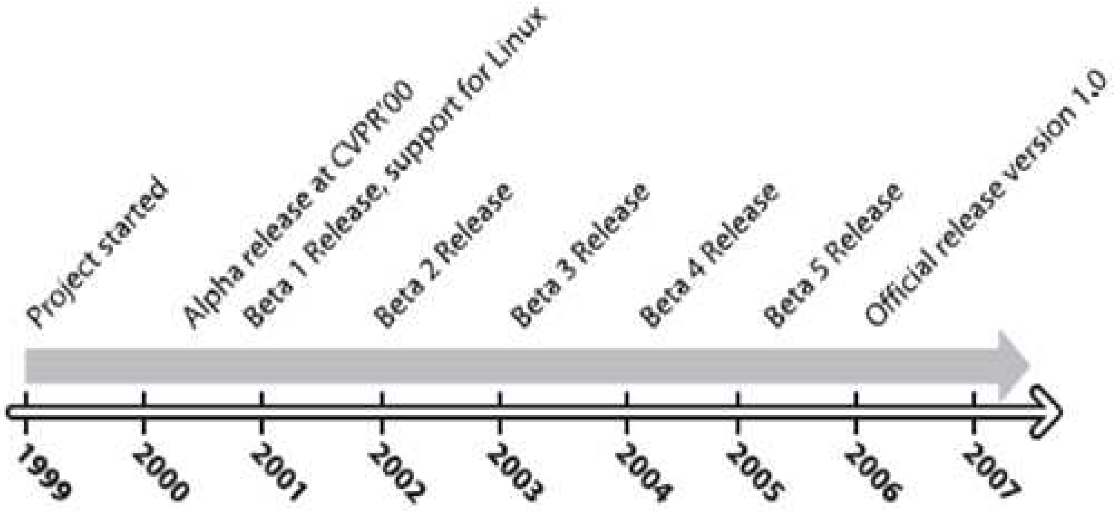


Figure 4.1.1 OpenCV Timeline

4.2 CAPTURING DATA FROM CAMERA:

Vision can mean many things in the world of computers. In some cases we are analyzing still frames loaded from elsewhere. In other cases we are analyzing video that is being read from disk. In still other cases, we want to work with real-time data streaming in from some kind of camera device. The process of acquiring data is explained in further sub-sections.

4.2.1 INPUT FROM A CAMERA

OpenCV, more specifically the High-GUI portion of the OpenCV library, provides us with an easy way to handle this situation. The method is analogous to how we read AVIs. Instead of calling cvCreateFileCapture(), we call cvCreateCameraCapture(). The latter routine does not take a file name but rather a camera ID number as its argument. Of course, this is important only when multiple cameras are available. The default value is –1, which means "just pick one"; naturally, this works quite well when there is only one camera to pick. The cvCreateCameraCapture() function returns the same CvCapture* pointer, which we can hereafter use exactly as we did with the frames grabbed from a video stream. Of course, a lot of work is going on behind the scenes to make a sequence of camera images look like a video, but we are insulated from all of that. We can simply grab images from the camera whenever we are ready for them and proceed as if we did not know the difference. For development reasons, most applications that are intended to operate in real time will have a video-in mode as well, and the universality of the CvCapture structure makes this particularly easy to implement.

Example:

```
CvCapture* capture;
if( argc==1 ) {
capture = cvCreateCameraCapture(0);
} else {
capture = cvCreateFileCapture( argv[1] );
}

assert( capture != NULL );
```

4.2.2 FRAMES

When working with video we must consider several functions, including how to read and write video files. We must also think about how to actually play back such files on the screen. The first thing we need is the CvCapture device. This structure contains the information needed for reading frames from a camera or video file. Depending on the source, we use one of two different calls to create and initialize a CvCapture structure.

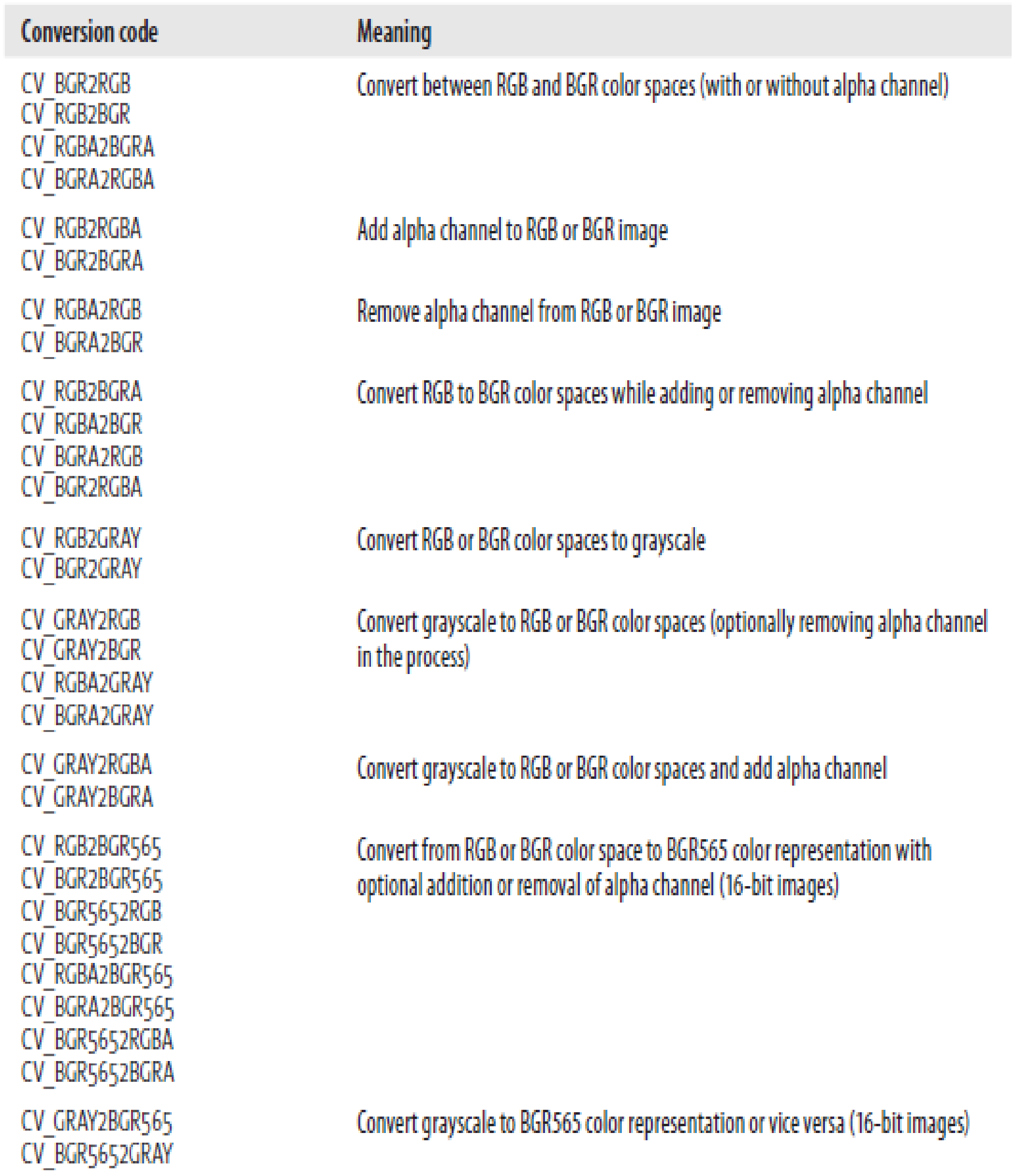
```
CvCapture* cvCreateFileCapture( const char* filename );
```

```
CvCapture* cvCreateCameraCapture( int index );
```

In the case of cvCreateFileCapture(), we can simply give a filename for an MPG or AVI file and OpenCV will open the file and prepare to read it. This is why it is always important to check the return value of cvCreateFileCapture(), because even if it works on one machine (where the needed DLL is available) it might not work on another machine (where that codec DLL is missing). Once we have the CvCapture structure, we can begin reading frames and do a number of other things. But before we get into that, let's take a look at how to capture images from a camera.

4.2.3 CONVERT IMAGE TO BINARY

The color conversion functions were for converting from one data type to another, and they expected the number of channels to be the same in both source and destination images. The complementary function is cvCvtColor(), which converts from one color space (number of channels) to another while expecting the data type to be the same. The exact conversion operation to be done is specified by the argument code, whose possible values are listed in Table 4.2.3 below:



Conversion code	Meaning
CV_BGR2RGB CV_RGB2BGR CV_RGBA2BGRA CV_BGRA2RGBA	Convert between RGB and BGR color spaces (with or without alpha channel)
CV_RGB2RGBA CV_BGR2BGRA	Add alpha channel to RGB or BGR image
CV_RGBA2RGB CV_BGRA2BGR	Remove alpha channel from RGB or BGR image
CV_RGB2BGRA CV_RGBA2BGR CV_BGRA2RGB CV_BGR2RGBA	Convert RGB to BGR color spaces while adding or removing alpha channel
CV_RGB2GRAY CV_BGR2GRAY	Convert RGB or BGR color spaces to grayscale
CV_GRAY2RGB CV_GRAY2BGR CV_RGBA2GRAY CV_BGRA2GRAY	Convert grayscale to RGB or BGR color spaces (optionally removing alpha channel in the process)
CV_GRAY2RGBA CV_GRAY2BGRA	Convert grayscale to RGB or BGR color spaces and add alpha channel
CV_RGB2BGR565 CV_BGR2BGR565 CV_BGR5652RGB CV_BGR5652BGR CV_RGBA2BGR565 CV_BGRA2BGR565 CV_BGR5652RGBA CV_BGR5652BGRA	Convert from RGB or BGR color space to BGR565 color representation with optional addition or removal of alpha channel (16-bit images)
CV_GRAY2BGR565 CV_BGR5652GRAY	Convert grayscale to BGR565 color representation or vice versa (16-bit images)

Table 4.2.3 Argument code for conversion operation in OPEN CV

Table 4.2.3(cont'd)

Conversion code	Meaning
CV_RGB2BGR555 CV_BGR2BGR555 CV_BGR5552RGB CV_BGR5552BGR CV_RGBA2BGR555 CV_BGRA2BGR555 CV_BGR5552RGBA CV_BGR5552BGRA	Convert from RGB or BGR color space to BGR555 color representation with optional addition or removal of alpha channel (16-bit images)
CV_GRAY2BGR555 CV_BGR5552GRAY	Convert grayscale to BGR555 color representation or vice versa (16-bit images)
CV_RGB2XYZ CV_BGR2XYZ CV_XYZ2RGB CV_XYZ2BGR	Convert RGB or BGR image to CIE XYZ representation or vice versa (Rec 709 with D65 white point)
CV_RGB2YCrCb CV_BGR2YCrCb CV_YCrCb2RGB CV_YCrCb2BGR	Convert RGB or BGR image to luma-chroma (aka YCC) color representation
CV_RGB2HSV CV_BGR2HSV CV_HSV2RGB CV_HSV2BGR	Convert RGB or BGR image to HSV (hue saturation value) color representation or vice versa
CV_RGB2HLS CV_BGR2HLS CV_HLS2RGB CV_HLS2BGR	Convert RGB or BGR image to HLS (hue lightness saturation) color representation or vice versa
CV_RGB2Lab CV_BGR2Lab CV_Lab2RGB CV_Lab2BGR	Convert RGB or BGR image to CIE Lab color representation or vice versa
CV_RGB2Luv CV_BGR2Luv CV_Luv2RGB CV_Luv2BGR	Convert RGB or BGR image to CIE Luv color representation
CV_BayerBG2RGB CV_BayerGB2RGB CV_BayerRG2RGB CV_BayerGR2RGB CV_BayerBG2BGR CV_BayerGB2BGR CV_BayerRG2BGR CV_BayerGR2BGR	Convert from Bayer pattern (single-channel) to RGB or BGR image

Example:

```
void cvConvertImage(

const CvArr* src,
CvArr* dst,
int flags = 0 );
```

4.3 EDGE DETECTION

There are two commonly used algorithms for edge detection.

- Laplace based algorithms
- Canny edge detector

The Canny edge detection is used in the path tracking process and is explained below.

4.3.1 CANNY EDGE DETECTOR

The method for finding edges was further refined by J. Canny in 1986 called the *Canny edge detector.* One of the differences between the Canny algorithm and the simpler, Laplace-based algorithm from the previous section is that in the Canny algorithm, the first derivatives are computed in *x* and *y* and then combined into four directional derivatives. The points where these directional derivatives are local maxima are then candidates for assembling into edges. Figure 4.3.2a shows the Laplace method.

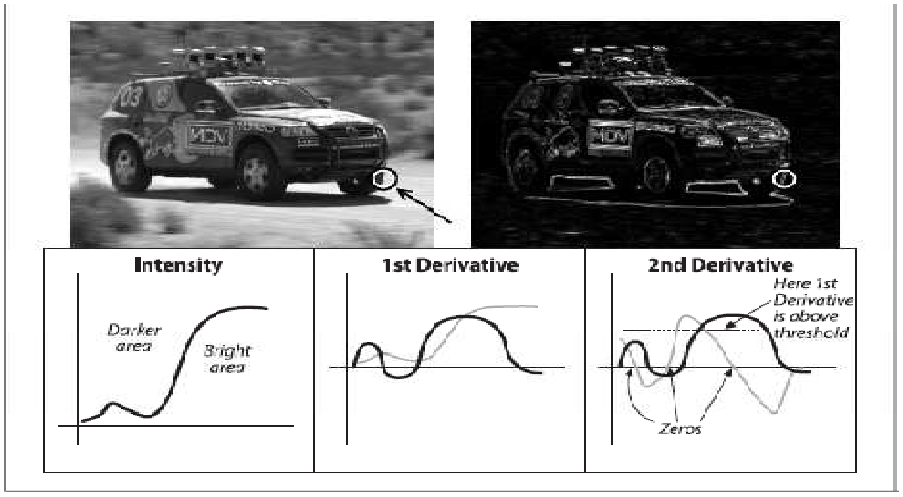


Figure 4.3.1a Laplace method of edge detection

However, the most significant new dimension to the Canny algorithm is that it tries to assemble the individual edge candidate pixels into *contours*. These contours are formed by applying an *hysteresis threshold* to the pixels. This means that there are two thresholds, an upper and a lower. If a pixel has a gradient larger than the upper threshold, then it is accepted as an edge pixel; if a pixel is below the lower threshold, it is rejected. If the pixel's gradient is between the thresholds, then it will be accepted only if it is connected to a pixel that is above the high threshold. Canny recommended a ratio of high:low threshold between 2:1 and 3:1. Figure shows the results of applying cvCanny() to a test pattern and a photograph using high:low hysteresis threshold ratios of 5:1 and 3:2, respectively. Figure 4.3.2b shows Canny edge detection.

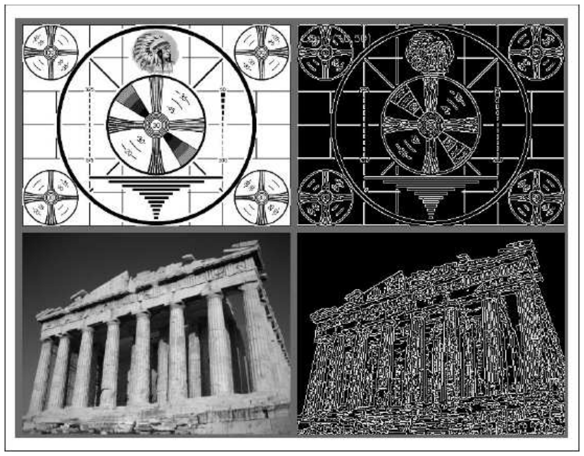


Figure 4.3.1b Canny edge detection

4.4 HOUGH TRANSFORM

The *Hough transform* is a method for finding lines, circles, or other simple forms in an image. The original Hough transform was a line transform, which is a relatively fast way of searching a binary image for straight lines. The transform can be further generalized to cases other than just simple lines.

4.4.1 HOUGH LINE TRANSFORM

The basic theory of the Hough line transform is that any point in a binary image could be part of some set of possible lines. Figure 4.4.1a shows the results of Canny edge detection for two different images for two different values of thresholds.

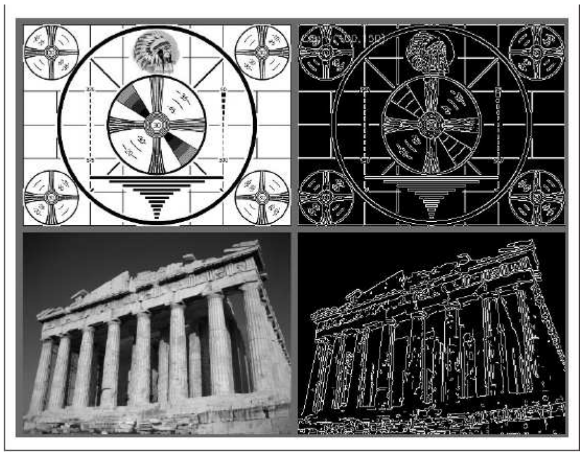


Figure 4.4.1a Results of Canny edge detection for two different images when the high and low thresholds are set to 150 and 100, respectively

If we parameterize each line by, for example, slope a and an intercept b, then a point in the original image is transformed to a locus of points in the (a, b) plane corresponding to all of the lines passing through that point. If we convert every nonzero pixel in the input image into such a set of points in the output image and sum over all such contributions, then lines that appear in the input (i.e., (x, y) plane) image will appear as local maxima in the output (i.e., (a, b) plane) image. Because we are summing the contributions from each point, the (a, b) plane is commonly called the *accumulator plane*. It might occur to you that the slope-intercept form is not really the best way to represent all of the lines passing through a point because of the considerably different density of lines as a function of the slope, and the related fact that the interval of possible slopes goes from $-\infty$ to $+\infty$. It is for this reason that the actual parameterization of the transform image used in numerical computation is somewhat different. The preferred parameterization represents each line as a point in polar coordinates (ρ, θ), with the implied line being the line passing through the indicated point but perpendicular to the radial from the origin to that point. The equation for such a line is:

$$\rho = x \cos\theta + y \sin\theta$$

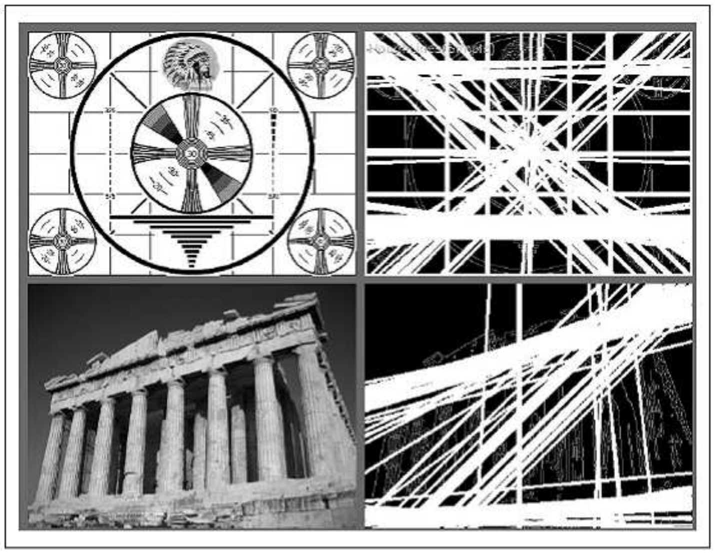


Figure 4.4.1b The Hough line transform finds many lines in the image; some of the lines found are expected but others may not be.

The Hough transform might find lines other than the expected lines in the image. This is shown in Figure 4.4.1b. For a particular point, many lines are parameterized for different values of *ρ and θ, which when taken together form a curve of characteristic shape. This is shown in Figure 4.4.1c as follows.*

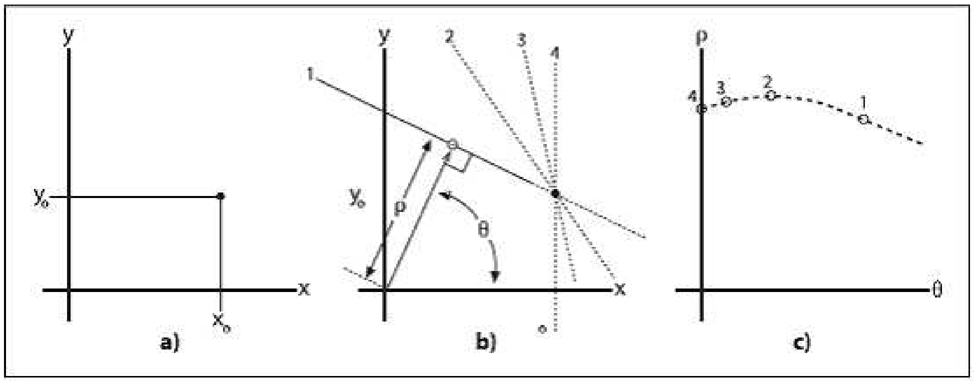


Figure 4.4.1c A point (x0, y0) in the image plane (panel a) implies many lines each parameterized by a different ρ and θ (panel b); these lines each imply points in the (ρ, θ) plane, which taken together form a curve of characteristic shape (panel c)

The OpenCV Hough transform algorithm does not make this computation explicit to the user. Instead, it simply returns the local maxima in the (ρ, θ) plane. However, you will need to understand this process in order to understand the arguments to the OpenCV Hough line transform function.

```
CvSeq* cvHoughLines2(
CvArr* image,
void* line_storage,
int method,
double rho,
double theta,
int threshold,
double param1 = 0,
double param2 = 0
);
```

The first argument is the input image. It must be an 8-bit image, but the input is treated as binary information i.e. all nonzero pixels are considered to be equivalent. The second argument is a pointer to a place where the results can be stored, which can be either a memory storage or a plain N-by-1 matrix array (the number of rows, *N*, will serve to limit the maximum number of lines returned). The next argument, method, can be CV_HOUGH_STANDARD, CV_HOUGH_PROBABILISTIC, or CV_HOUGH_MULTI_SCALE for (respectively) SHT, PPHT, or a multi-scale variant of SHT. The next two arguments, rho and theta, set the resolution desired for the lines (i.e., the resolution of the accumulator plane). The units of rho are pixels and the units of theta are radians; thus, the accumulator plane can be thought of as a two-dimensional histogram with cells of dimension rho pixels by theta radians.This last argument is a bit tricky in practice; it is not normalized, so you should expect to scale it up with the image size for SHT. This argument is, in effect, indicating the number of points in the edge image that must support the line for the line to be returned. Figure 4.4.1d shows the results of Canny Edge detection overlaid with the progressive probabilistic Hough Transform as follows.

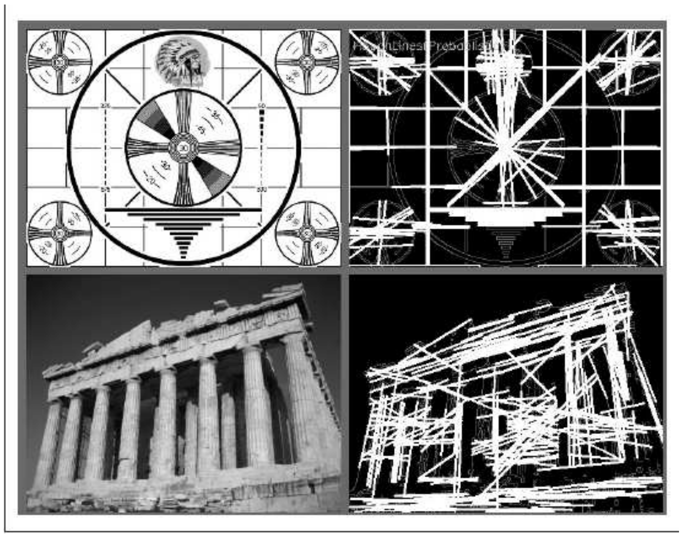


Figure 4.4.1d The Canny edge detector (param1=50, param2=150) is run first, with the results shown in gray, and the progressive probabilistic Hough transform (param1=50, param2=10) is run next, with the results overlaid in white; It is noticeable that the strong lines are generally picked up by the Hough transform

The param1 and param2 arguments are not used by the SHT. For the PPHT, param1 sets the minimum length of a line segment that will be returned, and param2 sets the separation between collinear segments required for the algorithm not to join them into a single longer segment. For the multi-scale HT, the two parameters are used to indicate higher resolutions to which the parameters for the lines should be computed. The multi-scale HT first computes the locations of the lines to the accuracy given by the rho and theta parameters and then goes on to refine those results by a factor of param1 and param2, respectively.

If the line storage value was a pointer to a memory store, then the return value will be a pointer to a CvSeq sequence structure. In that case, you can get each line or line segment from the sequence with a command like float* line = (float*) cvGetSeqElem(lines , I); where lines is the return value from cvHoughLines2() and I is index of the line of interest. In this case, line will be a pointer to the data for that line, with line[0] and line[1] being the floating-point values ρ and θ (for SHT and MSHT) or CvPoint structures for the endpoints of the segments (for PPHT).

4.4.2 HOUGH-CIRCLE TRANSFORM

The Hough circle transform works in a manner roughly analogous to the Hough line transforms just described. The reason it is only 'roughly' is that if one were to try doing the exactly analogous thing, the accumulator plane would have to be replaced with an accumulator volume with three dimensions: one for x, one for y, and another for the circle radius r. This would mean far greater memory requirements and much slower speed. The implementation of the circle transform in OpenCV avoids this problem by using a somewhat more tricky method called the Hough gradient method.

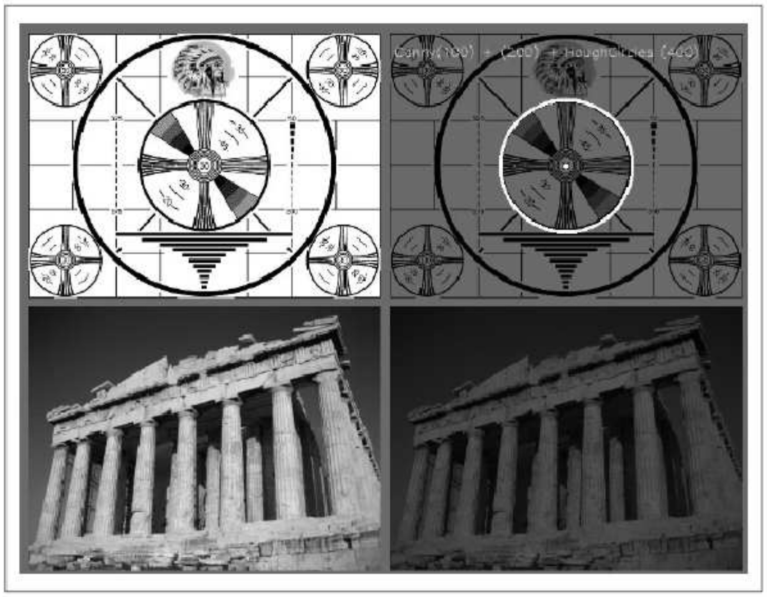


Figure 4.4.2 The Hough circle transform finds some of the circles in the test pattern and (correctly) finds none in the photograph

4.4 ROBOCAR CAMERA: OPERATION AND PROCESSING

The camera is attached to the front end of the 33obocar so that it has a clear view of the road ahead on which it is supposed to travel and avoid any possible collisions with the obstacles. The camera is constantly transmitting its video by means of an in-built transmitter wirelessly to the computer on which the further processing is supposed to take place.

The first step in processing the video output of the camera is to import frames of the video on which further analysis is to be performed. For the said purpose, the colored frames are captured and imported to the visual studio. The colored frame captured is shown as follows in Figure 4.4a :

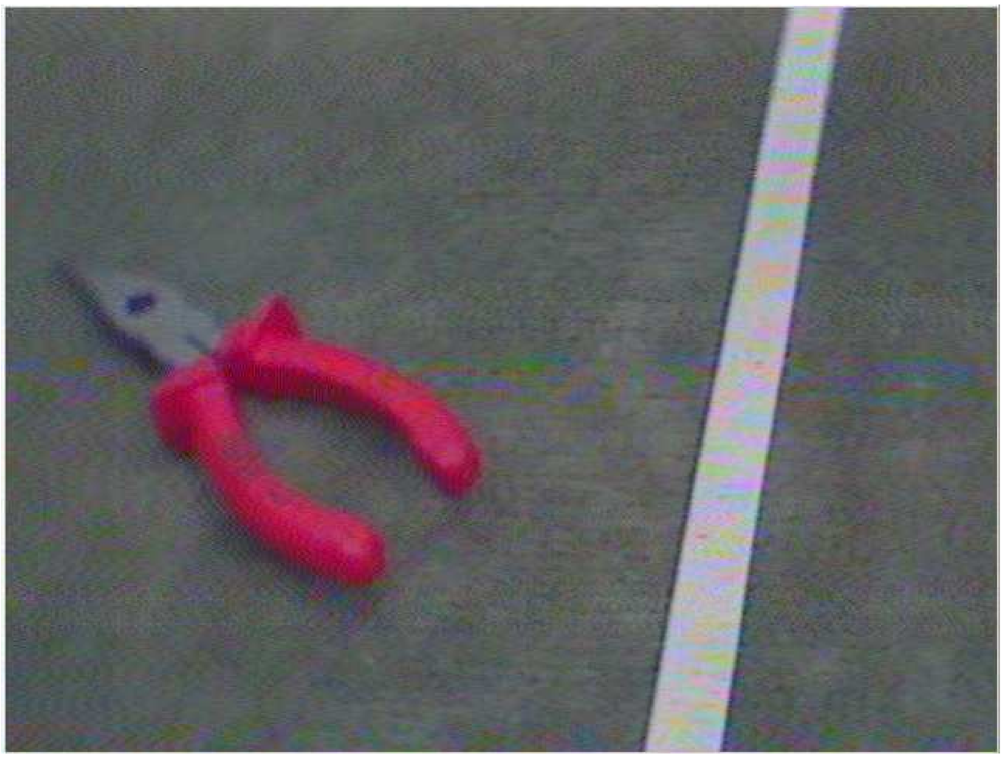


Figure 4.4a Colored frame captured from the camera

The next step shown in Figure 4.4b is to convert every frame into a gray-scale image which actually portrays the original colored image into a combination of gray shades.

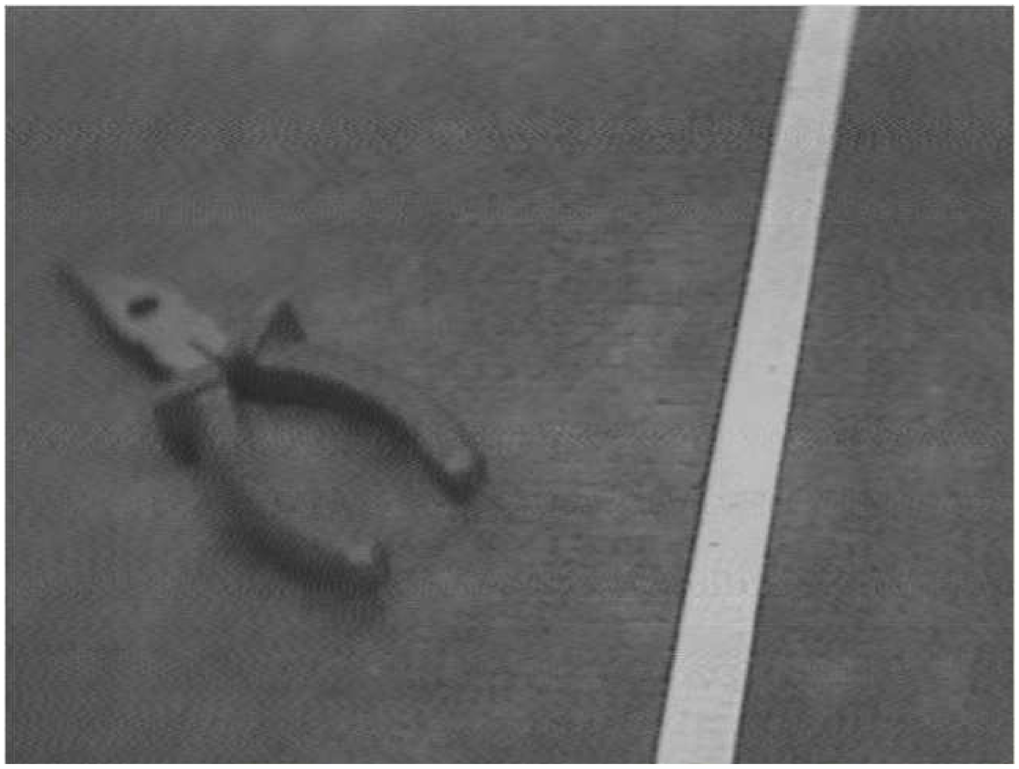


Figure 4.4b Colored image converted into a Grayscale image

Next, a pure black and white image without any shades of gray is achieved. This is done by means of a threshold value which serves as the basis of discrimination of different shades of gray into pure white or black. The shades above the threshold value are categorized into black and those below it are categorized into pure white. This is also called a *binary image* for the fact that it only contains two colors. Hence a frame is achieved which is shown as follows in Figure 4.4c

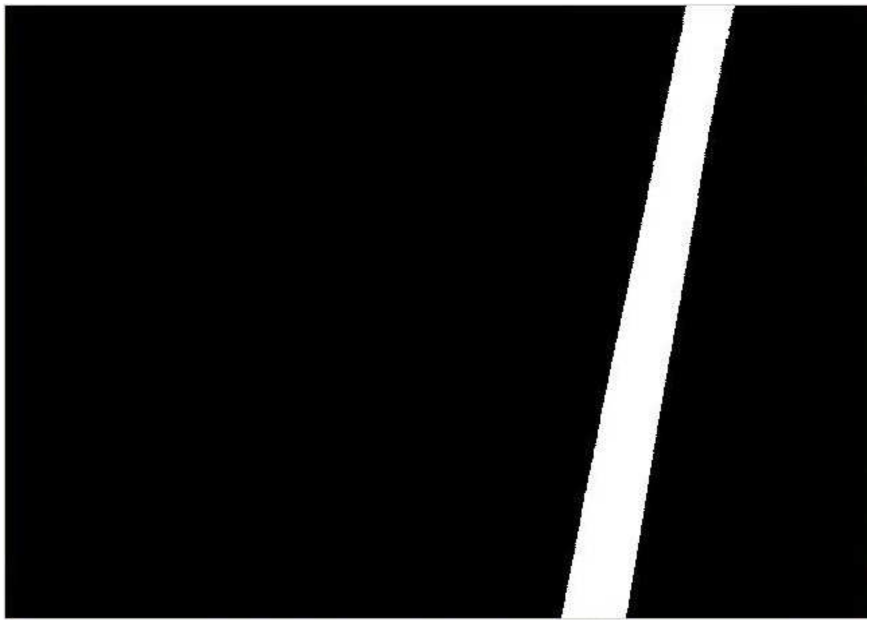


Figure 4.4c Grayscale image converted into a Binary image(Black and White)

Now the Canny Edge Detection procedure is applied to the binary image which extracts the basic borderline of the two colors thus identifying an edge in the frame. Similarly all the edges are computed.

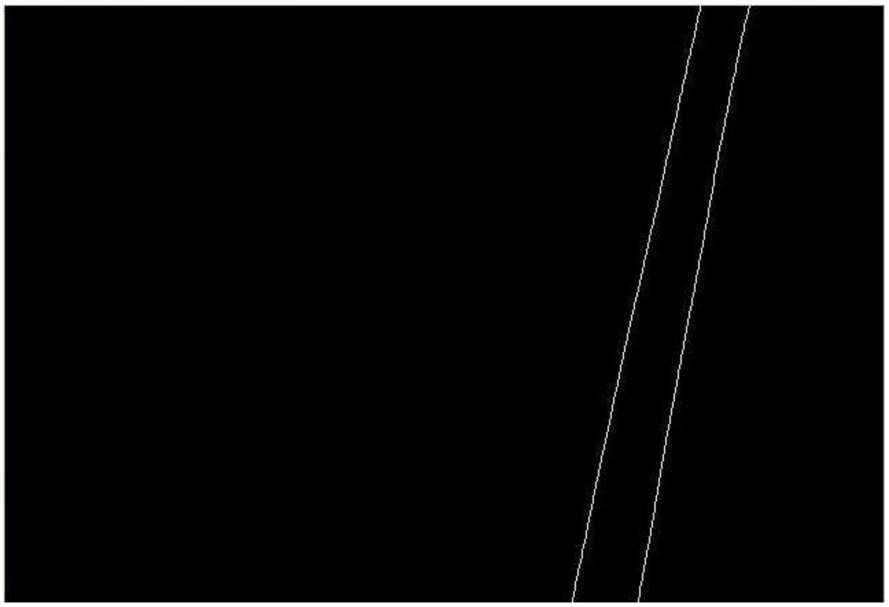


Figure 4.4d Canny Edge Detection applied to the binary frame

The resulting frame shown in Figure 4.4d is achieved. Finally the Hough Line Transform is applied which differentiates the non-linear edges from the linear ones and hence the linear edges are extracted. This is the main characteristic which defines the boundary of any road/path that it has linear edges.

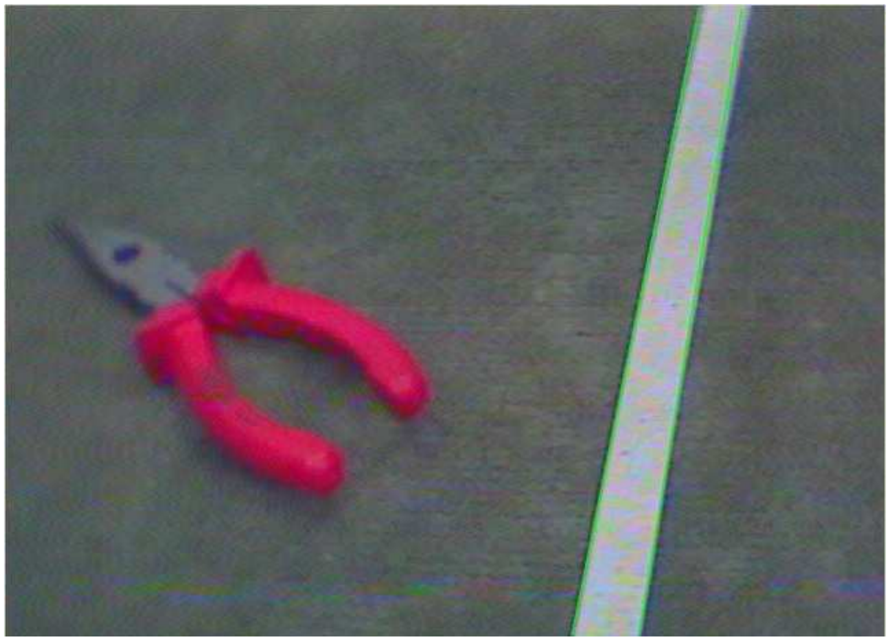


Figure 4.4e Hough Line Transform applied to the binary frame. The edges are marked by the green lines.

Figure 4.4e shows the lines in green which are extracted from the basic frame of the camera and are considered as the boundary of the road. Now the lines extracted give a certain value of slope. If the path is straight, then the value of slope/gradient remains within certain limits. But if the path takes a turn towards either side the value of the slope/gradient changes. The r37obocar is instructed to change its course with respect to the observed change in direction of path calculated through the change in gradient.

CHAPTER-5

DESIGN STRATEGY AND HARDWARE SPECIFICATIONS

5.1 INTRODUCTION

In robotics, obstacle avoidance is the task of satisfying some control objective subject to non-intersection or non-collision position constraints. Normally obstacle avoidance is considered to be distinct from path planning in that one is usually implemented as a reactive control law while the other involves the pre-computation of an obstacle-free path which a controller will then guide a robot along .The design of the car is kept simple so a three wheeler is designed having capability to turn around itself at its place.

Images taken from the wireless camera are received by the camera receiver and transmitted to computer through USB interface .The processing of the images is done in real time and decision is constantly transmitted to the ROBOCAR through 49MHZ RC transmitter which is connected to the computer through serial interface. The decisions received by the car through RC receiver mounted on it .The obstacle detection task is performed with the sonar which is mounted on the car .The whole project schematic is shown in below in Figure 5.1.

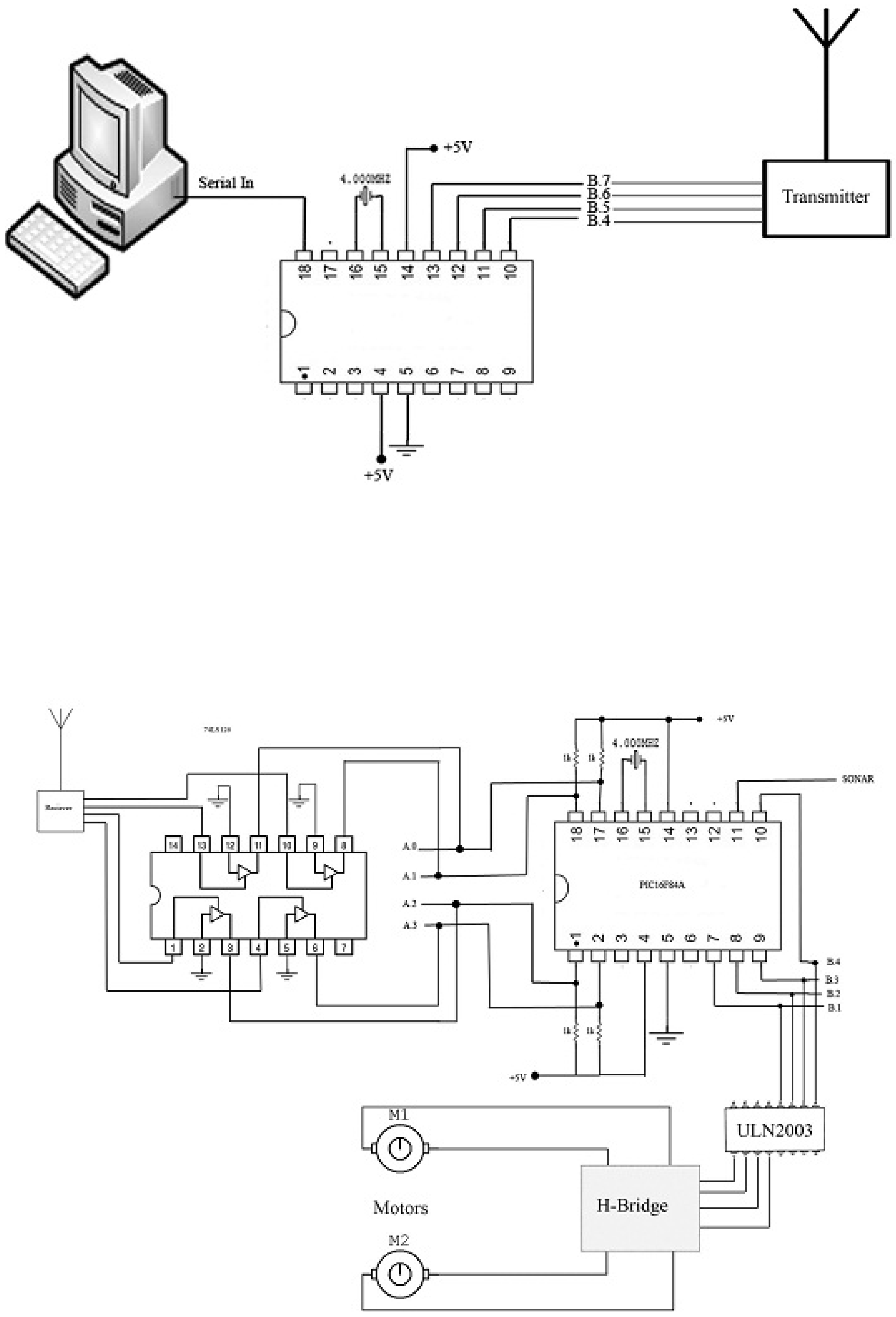


Figure 5.1 Scheme of the operation of Robocar

5.2 RC TRANCEIVER

RC toys typically have a small handheld device that includes some type of controls and the radio transmitter. The transmitter sends a signal over a frequency to the receiver in the toy. The transmitter as shown in Figure 5.2a has a power source, usually a 9-volt battery that provides the power for the controls and transmission of the signal. The key difference between radio controlled and remote controlled toys is that remote controlled toys have a wire connecting the controller and the toy, while radio controlled toys are always wire-less.

Most RC toys operate at either **27 MHz** or **49 MHz.** This pair of frequencies has been allocated by the FCC for basic consumer items, such as garage door openers, walkie-talkies and RC toys. Advanced RC models, such as the more sophisticated RC airplanes, use 72-MHz or 75-MHz frequencies. The majority of RC toys are labeled with the frequency range they operate in.

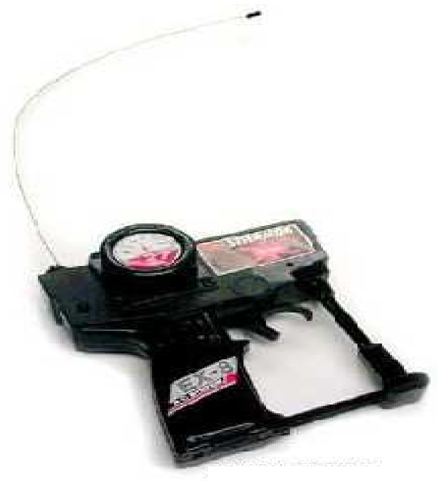


Figure 5.2a RC transmitter

Most RC toy manufacturers make versions of each model for both frequency ranges (27 MHz and 49 MHz). That way, you can operate two of the same model simultaneously, for racing or playing together, without having to deal with interference between the two transmitters. Some manufacturers also provide more specific information about the exact portion of the frequency band that the toy operates in.

Transmitters range from single-function simple controllers to full-function controllers with a wide range of options. An example of a **single-function controller** is one that makes the toy go forward when the trigger is pressed and backward when it is released. To stop the toy, you have to actually turn it off. Figure 5.2.b shows the basic circuit layout of an RC transmitter.

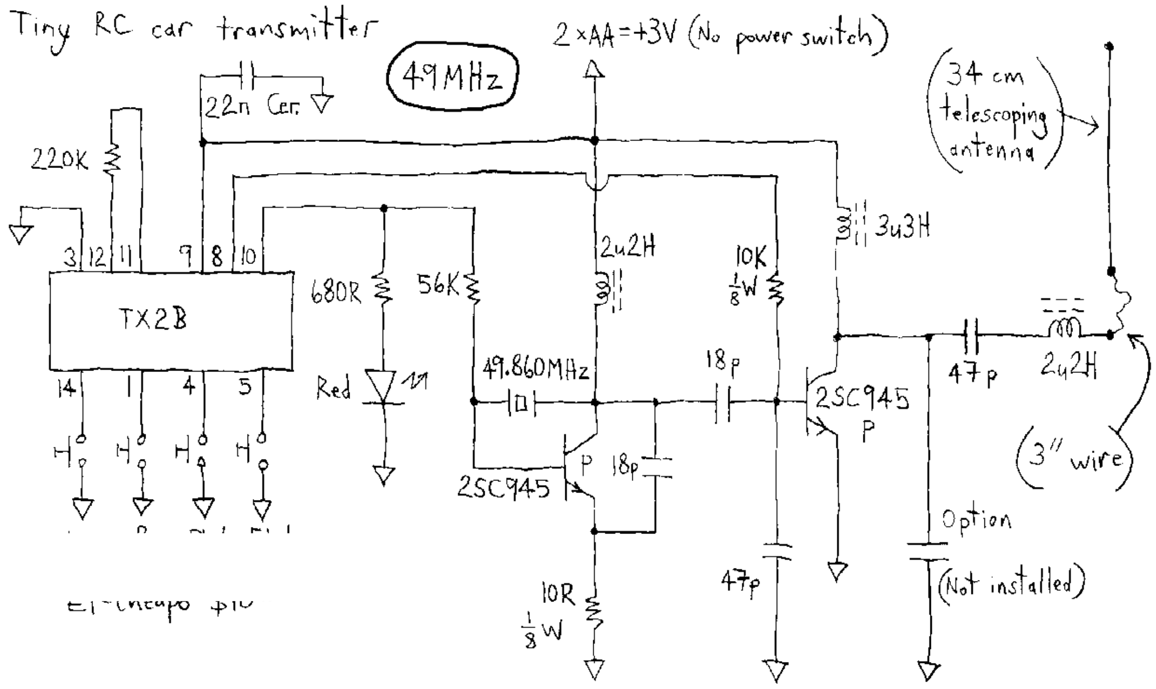


Figure 5.2b Basic circuit diagram of an RC car transmitter

Most full-function controllers have six controls:

- Forward
- Reverse
- Forward and Left
- Forward and Right
- Reverse and Left
- Reverse and Right

In most full-function controllers, not pressing any buttons or turning any knobs causes the toy to stop and await further commands. Controllers for more advanced RC systems often use **dual joysticks** with several levels of response for precise control. The transmitter runs on only 3V whereas most others run on more voltage.

5.3 74LS126

It forwards the desired voltage to the output pin according to the condition of input pin. It is shown in Figure 5.3 as follows.

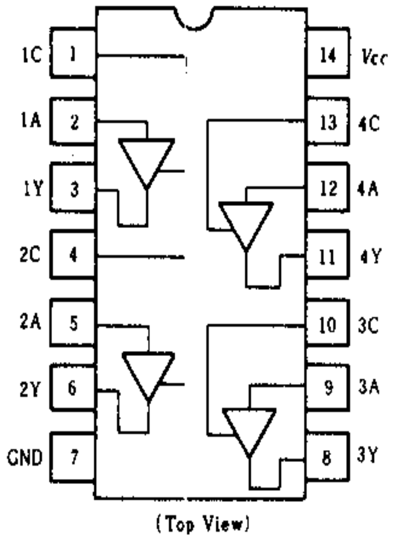


Figure 5.3 74LS126

5.4 RELAY

A relay as shown in Figure 5.4, quite simply, is a small machine consisting of an electromagnet (coil), a switch, and a spring. The spring holds the switch in one position, until a current is passed through the coil. The coil generates a magnetic field which moves the switch. It has single control circuit, but two separate current paths for the switch: One when the relay is off (no current through the control coil) and other when the relay is on, current is flowing through the control coil. When the 5 pin relay is off pins 4 and 5 have continuity and when its on 3 and 5 have continuity.

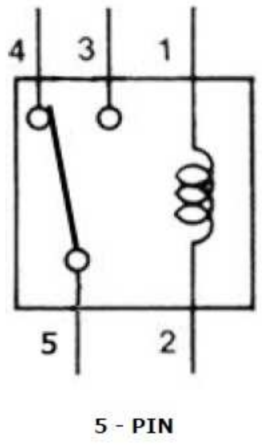


Figure 5.4 5-pin relay

5.5 H-BRIDGE

An H-bridge shown in Figure 5.5 is an electronic circuit which enables a voltage to be applied across a load in either direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards and backwards. The switching logic of H-bridge is shown in Table 5.5. H-bridges are available as integrated circuits, or can be built from discrete components.

- It consists 4 relays i.e. 2 relays drive each motor
- By default all relays are connected to +12 V DC.
- In case of a direction command, one relay out of two gets grounded, hence the circuit is completed and the motor sets into operation.

S1	**S2**	**S3**	**S4**	**Result**
1	0	0	1	Motor moves right
0	1	1	0	Motor moves left
1	1	1	1	No movement
0	1	0	1	Moves reverse
1	0	1	0	Moves forward

Table 5.5 Switching logic of an H-bridge

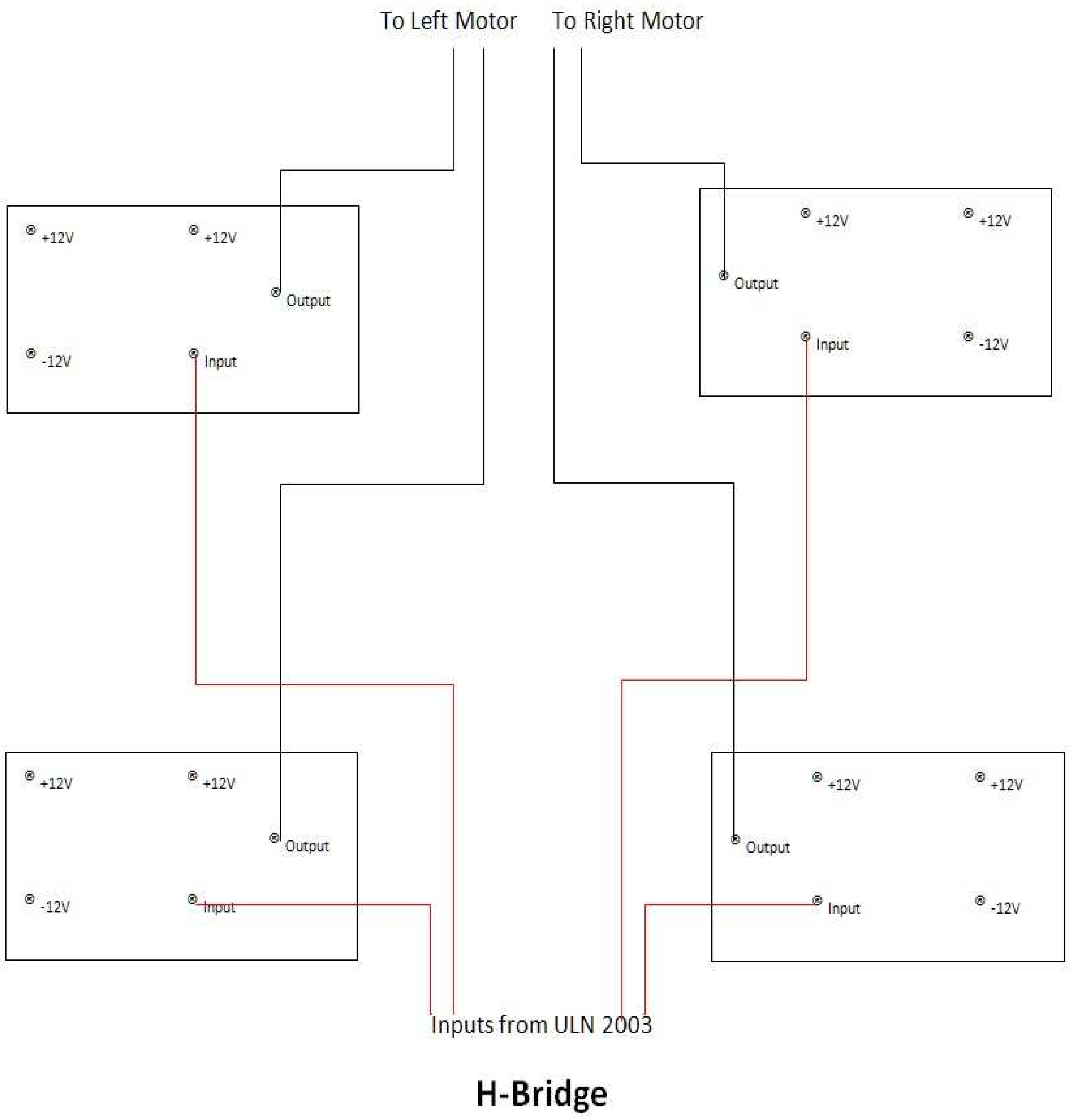


Figure 5.5 Construction of H-bridge

5.6 ULN2003

The ULN2003 consists of high voltage, high current Darlington arrays each containing seven open collector Darlington pairs with common emitters. It is used as a current amplifier. The Darlington transistor (often called a Darlington pair) is a compound structure consisting of two bipolar transistors (either integrated or separated devices) connected in such a way that the current amplified by the first transistor is amplified further by the second one .This configuration gives a much higher current gain. A ULN2003 is shown in Figure 5.6 as follows.

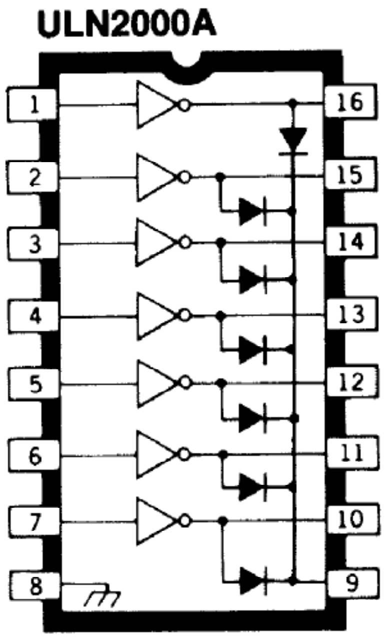


Figure 5.6 ULN2003A

5.7 PIC 16F84 MICROCONTROLLER

The controller used is PIC 16F84 as shown in Figure 5.7. It is a 18 pin controller with 1K flash with1024 programmable words. The program for the controller is written in Microcode Studio. The program needs to verify two conditions for its proper output to the H-bridge:

- Sonar mounted at the front end of the car
- Commands from wireless receiver

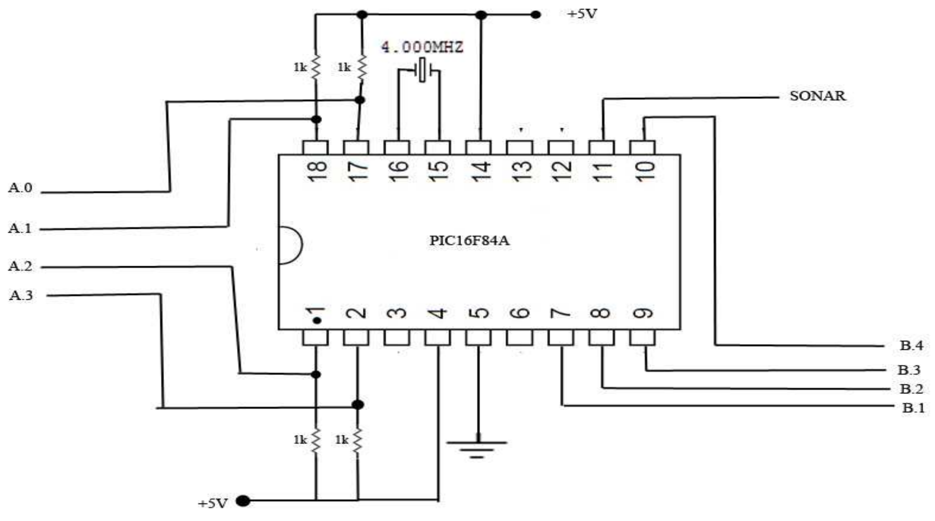


Figure 5.7 PIC16F84

It is powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions). CMOS Flash/EEPROM-based 8-bit microcontroller packs Microchip's powerful PIC architecture into an 18-pin package. The same device can be used for prototyping and production and the end application can be easily updated without removing the device from the end product via the ICSP. Easily adapted for automotive, industrial, appliances low power remote sensors, electronic locks and security applications. Specifications are shown in Table 5.7

Parameter Name	Value
Program Memory Type	Flash
Program Memory (KB)	1.75
CPU Speed (MIPS)	5
RAM Bytes	68
Data EEPROM (bytes)	64
Timers	1 x 8-bit
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	2 to 6
Pin Count	18

Table5.7 Pic specifications

5.8 MECHANICAL MODEL

The mechanical model of the car shown in Figure 5.8a consists of following parts:

- Aluminum Sheet platform reinforced by plywood
- Rear Rubber Wheels
- Front Caster Wheel
- Motor supporting plates
- Motors

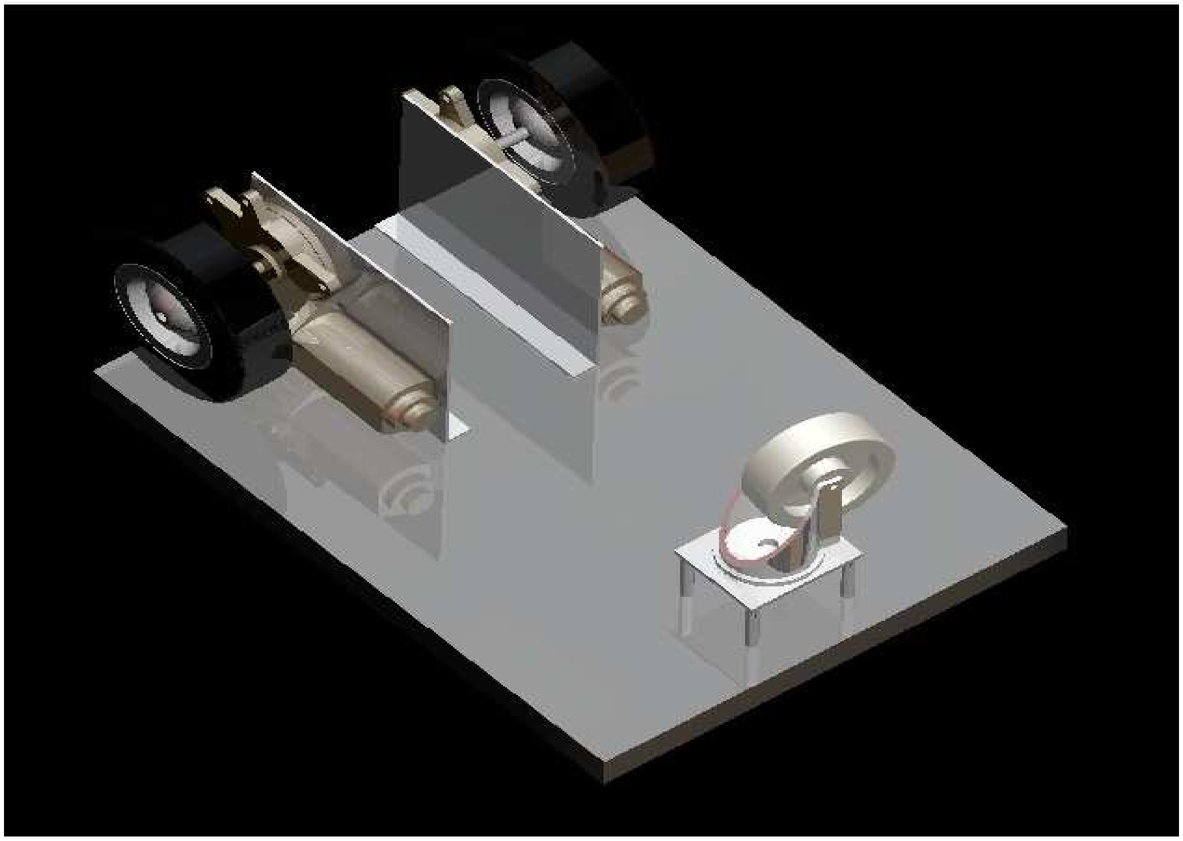


Figure 5.8a The platform

The dimensioning was done in PRO-E wildfire 4.0 and is shown in Figure 5.8b

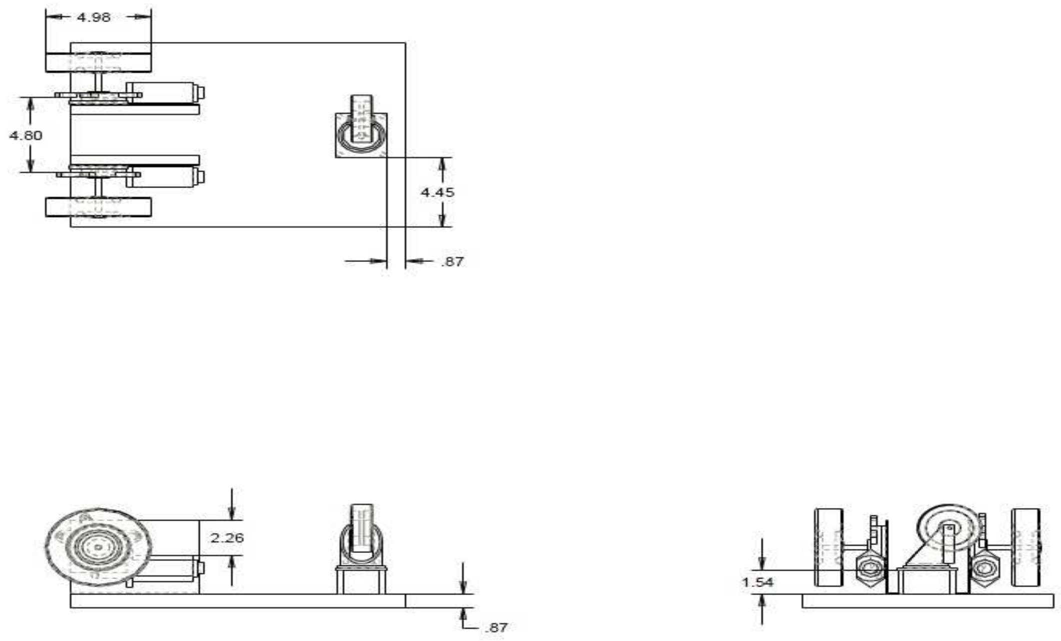


Figure 5.8b Dimensions of Robocar

5.8.1 ALUMINIUM SHEET

The aluminum platform (Figure 5.8.1a) provides basic strength to the platform aiding the movement of the car and accommodating the circuit and battery. A wooden board above this sheet provides insulation and stability to the vehicle.

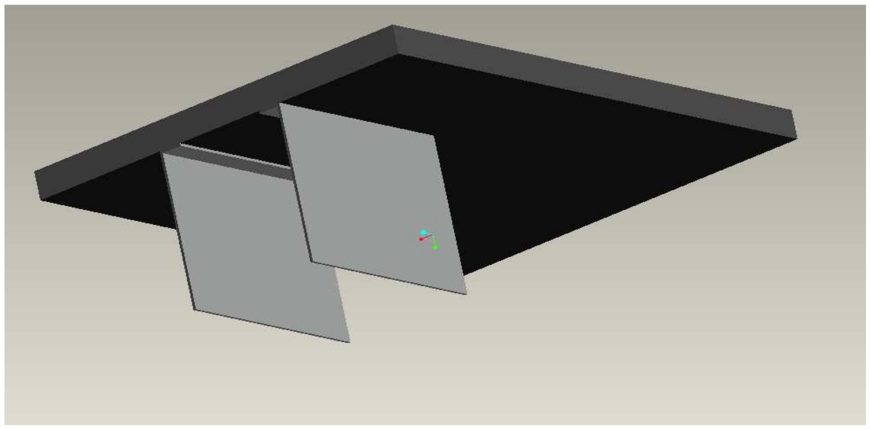


Figure 5.8.1a Aluminium sheet with plywood

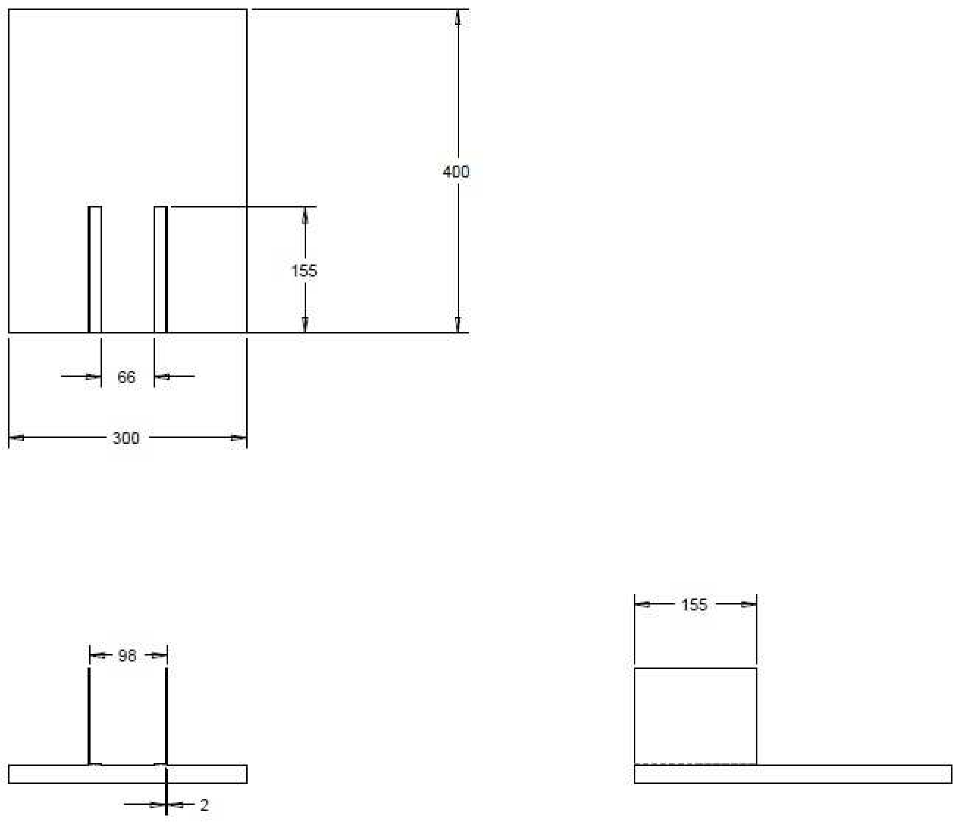


Figure 5.8.1b Dimensions of base

5.8.2 REAR RUBBER WHEELS

The first practical pneumatic tire was made by John Boyd Dunlop, a Scot, in 1887. Dunlop is credited with realizing that rubber could withstand the wear and tear of being a tire while retaining its resilience. Pneumatic tires are made of a flexible elastomeric material, such as rubber, with reinforcing materials such as fabric and wire. The rubber tires are used for the rear tires of the Robocar. These tires are powered by separate motors individually and are the main driving tires that cause the Robocar motion.

5.8.3 FRONT CASTER WHEEL

A caster (or castor) is an un-driven, single, double, or compound wheel that is designed to be mounted to the bottom of a larger object (the vehicle) so as to enable that object to be easily moved. Casters may be fixed to roll along a straight line path, or mounted on a pivot such that the wheel will automatically align itself to the direction of travel. They are available in various sizes, and are commonly made of rubber, plastic, nylon, aluminum, or stainless steel. Casters are found in numerous applications, including shopping carts, office chairs, and material handling equipment. High capacity, heavy duty casters are used in many industrial applications, such as platform trucks, carts, assemblies, and tow lines in plants. Generally, casters operate well on smooth and flat surfaces. Figure 5.8.3a and 5.8.3b show the caster wheel and its dimensions.

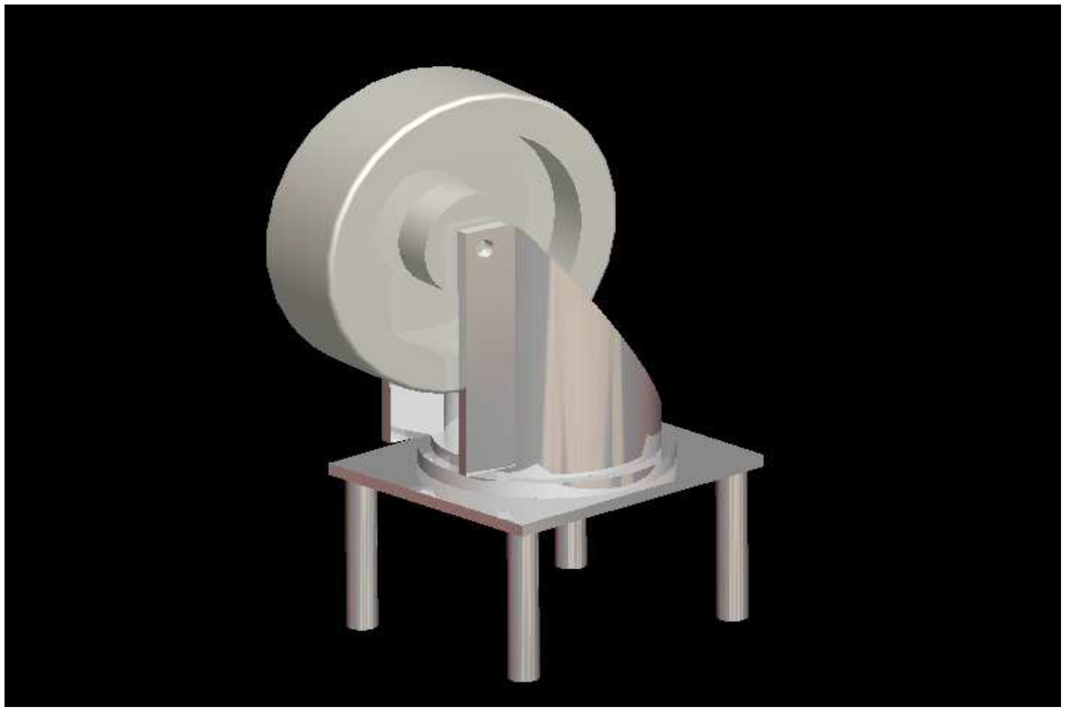


Figure 5.8.3a Castor wheel

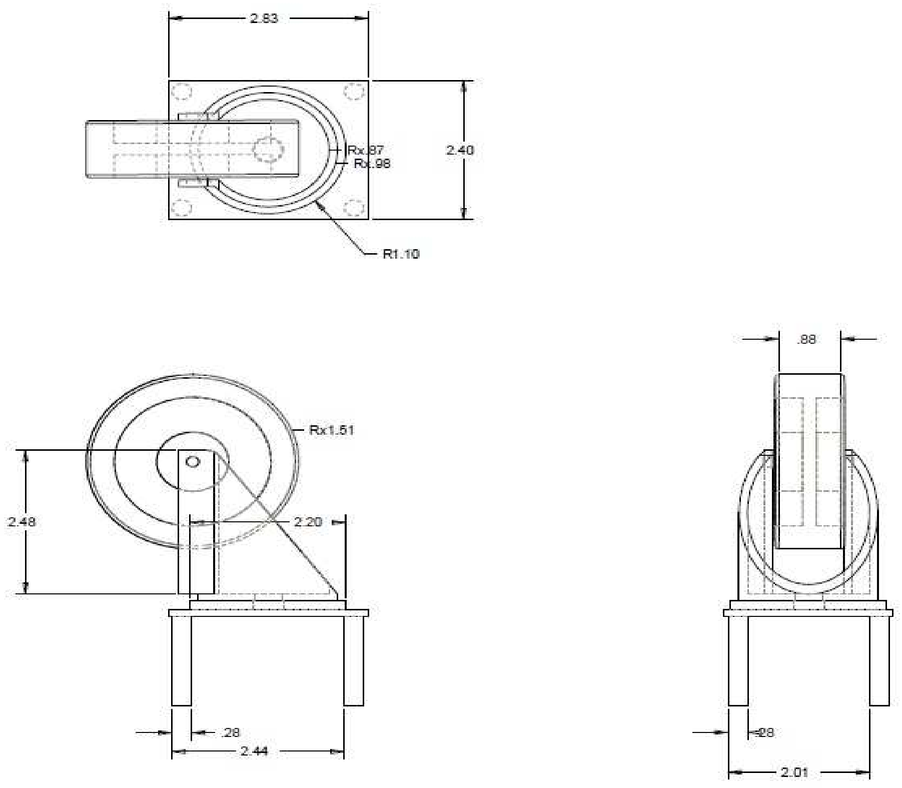


Figure 5.8.3b Dimensions of castor wheel

5.8.3.1 RIGID CASTERS

A basic, rigid caster consists of a wheel mounted to a stationary fork. The orientation of the fork, which is fixed relative to the vehicle, is determined when the caster is mounted to the vehicle. An example of this is the wheels found at the rear of a shopping cart. Rigid casters tend to restrict vehicle motion so that the vehicle travels along a straight line.

5.8.3.2 SWIVEL CASTERS

Like the simpler rigid caster, a swivel caster incorporates a wheel mounted to a fork, but an additional swivel joint above the fork allows the fork to freely rotate about 360°, thus enabling the wheel to roll in any direction. This makes it possible to easily move the vehicle in any direction without changing its orientation. Swivel casters are sometimes attached to handles so that an operator can manually set their orientation. When in motion along a straight line, a swivel caster will tend to automatically align to, and rotate parallel to the direction of travel. This can be seen on a shopping cart when the front casters align parallel to the rear casters when traveling down an aisle. A consequence of this is that the vehicle naturally tends to travel in a straight direction. Precise steering is not required because the casters tend to maintain straight motion. This is also true during vehicle turns. The caster rotates parallel to the turning radius and provides a smooth turn. This can be seen on a shopping cart as the front wheels rotate at different velocities, with different turning radius depending on how tight a turn is made. The angle and distance of the wheel axles and swivel joint (Figure 5.8.3.2) can be adjusted for different types of caster performance.

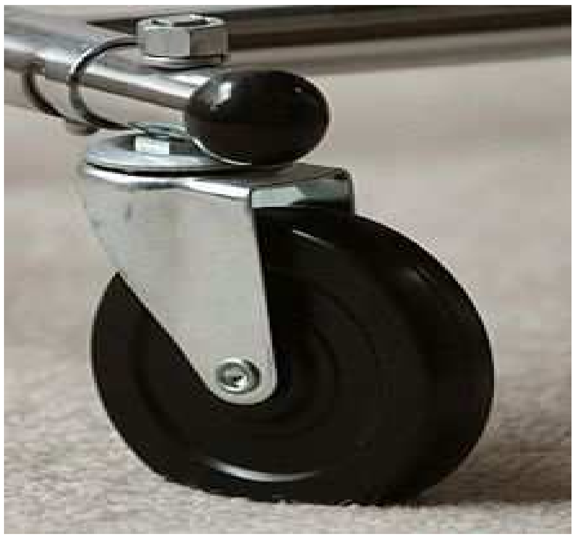


Figure 5.8.3.2 Swivel castor

5.8.3.4 CASTER FLUTTER

One major disadvantage of the casters is flutter. A common example is on a supermarket shopping cart, when one caster rapidly swings side-to-side. This oscillation is known as caster flutter and occurs naturally at certain speeds. The speed at which a caster flutters is based on the weight on the caster and the distance between the wheel axle and steering joint. This distance is known as trailing distance. Increasing this distance can eliminate flutter at moderate speeds. Generally, flutter occurs at high speeds.

What makes flutter dangerous is that it can cause a vehicle to suddenly move in an unwanted direction. Flutter occurs when the caster is not in full contact with the ground and therefore its orientation is uncontrollable. As the caster regains full contact with the ground, it can be in any orientation. This can cause the vehicle to suddenly move in the direction that the caster is pointed. At slower speeds, the caster's ability to swivel can correct the direction and can continue travel in the desired direction. But at high speeds this can be dangerous as the wheel may not be able to swivel quickly enough and the vehicle may lurch in any direction. Electric and racing wheelchair designers are very concerned with flutter because the chair must be safe for riders. Increasing trailing distance can increase stability at higher speeds for wheelchair racing, but may create it at lower speeds for everyday use. Unfortunately, the more trails the caster has, the more space the caster requires to swivel. Therefore, in order to accommodate this extra swivel space, lengthening of frame or extending the footrests maybe required. This tends to make the chair more cumbersome.

Caster flutter can be controlled by adding dampers or increasing the friction of the swivel joints. A simple method for completing is by adding washers to the swivel joint. The friction increases as the weight on the front of the chair increases. Anytime the caster begins to flutter, it slows the chair and shifts weight to the front wheels. There are several online anti-flutter kits for retrofitting wheel chair casters in this manner.

5.8.4 MOTOR SUPPORTING PLATES

The plates in Figure 5.8.4 are attached at the rear bottom of the base sheet and provide a stable and firm place for the tires to support the fine movement of car. The motors used are explained in the next section.

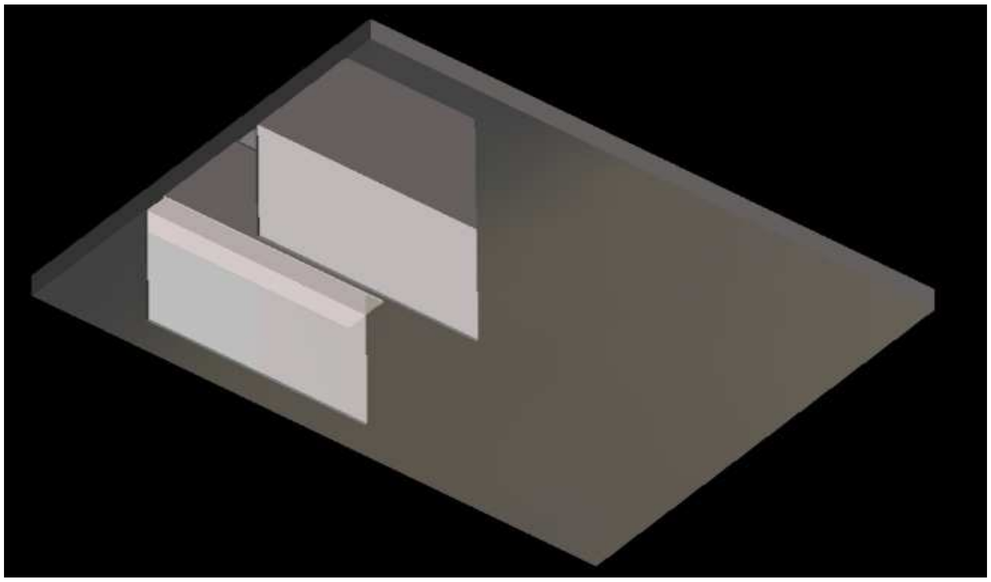


Figure 5.8.4 Motor supporting plates attached to the main platform of the Robocar

5.8.5 POWER WINDOW MOTORS

It is a two pole KEYANG dc motor (Figure 5.8.5), operates at 12 volt (48 watt) capable of providing 35 Nm torque.

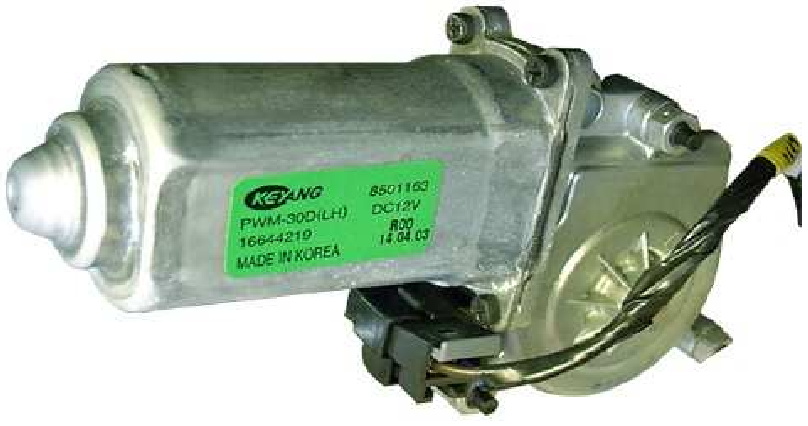


Figure 5.8.5 48 Watt power window motor

These are just a few examples of the many criteria imposed:

- Operational safety
- Assembly-friendliness
- Packaging space-efficiency
- Operational reliability
- Weight optimization
- Output efficiency
- Cost-effectiveness

CHAPTER 6

TESTING AND ANALYSIS

6.1 INTRODUCTION

This chapter includes the testing process and implementation of the project. The purpose of this project is to adapt an existing obstacle detection and avoidance algorithm to a roadway environment. The results are discussed and the improvement areas have been highlighted. Further possible potential applications for the project are described as well.

6.2 TESTING

The testing phase included the fine tuning of limits and threshold values according to the moving and turning speed of the car. The outputs of the individual elements were interfaced and the results were tested individually and collectively. For testing the working of the robocar, the car is subjected to a pre-defined path which has white boundary lines and is randomly filled with obstacles. The goal of the robocar is to stay on the path and avoid potential collisions with the obstacles that come in its path. As the program is run, the data acquisition starts first, followed by the analysis of the frames that are imported to the computer through the camera along with the processing of continuous output of the sonar.

The main data processing activities are real-time data extraction from the camera, and echo processing of the sonar. These activities are carried out in parallel in real time. However they are individually described in the following sections.

6.2.1 REALTIME DATA ACQUISITION FROM CAMERA

This model task involves creating a vision system using real time data output from the camera. The vision system provides information to the model about the position and orientation of visible road edges. The vision system is providing information to the model about road edges by using the information in the frames captured from the camera. The code runs in real time and the edge detection is carried out on the binary images which are constantly extracted through the colored frames. Thus the boundaries of the road are detected and the robocar stays on the track with the help of that.

6.2.2 SONAR ECHO PROCESSING

Testing phase includes checking whether the echo processing of sonar is enabling the robocar for obstacle avoidance correctly. After regular intervals the information is received from the sonar sensor. As the vehicle moves, new area is scanned by means of receiving echo of an ultrasonic signal. If a sonar sensor does not detect an obstacle in its cone, the entire area covered by this cone is marked as unoccupied. If the sensor detects an obstacle in its cone, then it reverses back and changes its direction.

6.3 POTENTIAL APPLICATIONS

The potential applications for this system include:

1) Automatic shortest route detection using GPS systems for personal and professional uses.
2) Military applications e.g. to be used in armored vehicles in battle fields.

3) Camera-based imaging applications where the route or path is required to be determined.
4) Submarine and under water applications by employing active sonar systems aided with visual imaging.

6.4 KEY FEATURES & BENEFITS

1) It reduces the chances of possible road accidents and collisions thereby decreasing the traffic hazards considerably.
2) It can be used in parking lots for proper parking of vehicles by consuming the least space
3) It uses an autonomous design where the manual driver interaction is negligible.
4) It can be employed for minimizing the response time.

6.5 CONCLUSION & FUTURE ENHANCEMENTS

The robocar achieved its desired objectives. The system provided an intelligent autonomous platform of the robocar that performed path detection and obstacle avoidance using a multi-sensor system independently. The future enhancements in the project may include further work on stereo vision, obstacle categorization using camera and more accurate obstacle detection by the use of a combination of different sensing techniques e.g. LIDAR, LADAR, RADAR or by the use of an array of a particular type of sensors producing more accurate and precise results.

REFERENCES

[1].J. H. M.J. Daily and K. Reiser, \Detecting obstacles in range imagery," Proc. of [ARPA] Image Under-standing Workshop, pp. 87{97, 1987.

[2] P. Veatch and L. Davis, \E_cient algorithms for obstacle detection using range data," CVGIP 50, pp. 50{74, 1990.

[3] B. Barshan and R. Kuc, \A bat-like sonar system for obstacle localization,"IEEE Transactions on Systems, Man, and Cybernetics 22, pp. 636{646, 1992.

[4] J. Borenstein and Y. Koren, \Obstacle avoidance with ultrasonic sensors," IEEE J. Robotics Automation 4,pp. 213{218, 1988.

[5] N. Ancona, A fast obstacle detection method based on optical ow," Proc. Of European Conference on Computer Vision, pp. 267{271, 1992.

[6] W. Enkelmann, \Obstacle detection by evaluation of optical ow _elds from image sequence," Image and Vision Computing 9, pp. 160{168, 1991.

[7] M. Bertozzi, A. Broggi, A. Fascicoli: “Vision-based
intelligent vehicles: State of art and perspectives”.
Robotics and Autonomous Systems, Vol. 32, pp. 1-16,October 1, 1999.

[8]P. Foggia2, A. Limongiello1, M. Vento \A moving object and obstacle detection system in real-time AVG and AMR Applications\1*1 Dip. di Ingegneria dell'Informazione ed Ingegneria Elettrica*

[9] *Intruder and Obstacle Detection Systems (IODS) for Railroads–1998 RequirementsWorkshop.* Carroll, A.; Meltzer, N.; Carpenter, JE. December 2001. DOT-VNTSC-FRA-00-07, DOT/FRA/ORD-01/13.

[10] *State-of-the-Art Technologies for Intrusion and Obstacle Detection for Railroad Operations*. daSilva, M.; Baron, W. February 2007. DOT-VNTSC-FRA-07-04, DOT/FRA/ORD-07/06.

[11] Sylvain Cardin, Daniel Thalmann and Frederic Vexo\Wearable Obstacle Detection System for visually impaired People\Virtual Reality Laboratory (VRlab) Ecole Polytechnique Fédérale de Lausanne (EPFL)

[12] A. J. Healey, D. P. Horner,, S. P. Kragelund \OBSTACLE DETECTION AND AVOIDANCE USING BLAZED ARRAY FORWARD LOOK SONAR\Center for Autonomous Underwater Vehicle Research

[13] Horner, D. P., Healey, A. J., "AUV Experiments in Obstacle Avoidance", *Proceedings of IEEE Oceans Conference, 2005*

[14] Healey, A. J., “Guidance Laws, Obstacle Avoidance , Artificial Potential Functions”,Chapter 3, Advances in Unmanned Marine Vehicles, IEE Control Series 69, Eds. Robertsand Sutton, March, 2006

[15] Hemminger, Daniel, L., “Vertical Plane Obstacle Avoidance And Control Of The REMUS Autonomous Underwater Vehicle Using Forward Look Sonar “, MSME Thesis,Naval Postgraduate School, Monterey, CA., 2006