Robotic Navigation in the Presence of Static and Dynamic Obstacles

Amit Arup Nayak
Deepankar Singh Purniya
Gigyanshu Ranjan Pradhan

National Institute of Technology Rourkela
Rourkela, 769 008, Odisha, India

Robotic Navigation in the presence of Static and Dynamic Obstacles

Project report submitted in May 2012 to the department of

Electronics and Communication Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

Amit Arup Nayak

(Roll 108EC005)

Gigyanshu Ranjan Pradhan

(Roll 108EI011)

Deepankar Singh Purniya

(Roll 108CS059)

under the supervision of

Dr. Pankaj K. Sa

National Institute of Technology Rourkela

Rourkela – 769 008, India

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Pankaj K Sa
Assistant Professor

May 12, 2012

Certificate

This is to certify that the work in the project entitled *Robotic Navigation in the Presence of Static and Dynamic Obstacles* by *Deepankar Singh Purniya*, *Amit Arup Nayak*, and *Gigyanshu Ranjan Pradhan* is a record of an original work carried out by them under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*. Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Pankaj K. Sa

Acknowledgment

First of all, we would like to express our deep sense of respect and gratitude towards our advisor and guide Prof Pankaj K. Sa, who has been the guiding force behind this work. We are greatly indebted to him for his constant encouragement, invaluable advice and for propelling us further in every aspect of our academic life. His presence and optimism have provided an invaluable influence on our career and outlook for the future. We consider it our good fortune to have got an opportunity to work with such a wonderful person.

Next, we want to express our respects to Prof. Kavi Arya and Prof. Banshidhar Majhi for teaching and also helping how to learn. He has been great sources of inspiration to us and we thank him from the bottom of our heart.

We would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

We would like to thank all our friends and especially our classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. We have enjoyed their companionship so much during our stay at NIT, Rourkela

Abstract

Computer Vision is the analytic study of motion dependent machines that extract valuable information from an image and perform some algorithmic processing on the captured images to derive necessary information to solve a particular or a set of tasks. Computer vision is the reconstruction of explicit, meaningful descriptions of physical objects obtained from their images. As far as scientific discipline is concerned, it moreover concerns itself with the theoretical aspects related to artificial intelligent systems that extract valuable information from the captured images. The role of computer vision in automated system is providing the detailed information about the working environment. A robust vision system should always be able to detect as well as identify objects reliably and provide an accurate and precise information as well as representation of the environment conditions to high level processes. The robust vision system should be highly effective and efficient, allowing all the objects which are limited and used as resource to respond quickly to a changing environment. Computer Vision is the key aspect to the design of the automatic responsive system by virtue of which the objects in the system localize themselves in the environment and act according to the changing nature of the environment variables.

The objective of the project was to build an autonomous vehicle for urban road which could guide itself to the specified location based on its decision making quality on basis of the path analysis using image processing and GPS. It would guide from the present source to the destination based on the path created using the wave points with obstacle avoidance of all objects, both static and dynamic in nature. It should also be capable of switching into manual mode control upon instruction being issued i.e. it can be controlled manually using the manual wireless control using X-Bee wireless module.

Contents

Certificate ii

Acknowledgement iii

Abstract iv

List of Figures vii

List of Tables ix

1 Introduction 1

1.1 Real Time Systems 1

1.2 Computer Vision 2

1.2.1 Role of CV in Robotics 3

1.2.2 OpenCV (the image processing tool) 3

1.3 Literature Survey 4

1.4 HSV color model 5

1.5 Goal and Objectives 6

1.5.1 Goal 6

1.5.2 Objectives 6

1.6 Hardware and Software Requirements 7

1.6.1 Hardware Requirements 7

1.6.2 Software Requirement. 15

2 Hardware Implementation **16**

2.1 Initializing Microcontroller (Atmega640) 16

2.2 Initializing Microcontroller (Atmega128) 16

2.2.1 Communicating with Pin Heading Compass Module (HMC6352) 17

2.2.2 I2C Communication Protocol 17

2.3 Initializing E-stop Remote 20

2.4 Data sending 21

2.5 Data receiving 21

2.6 Driving Motor Movement 21

2.7 Robot Control 22

3 Software Implementation **23**

3.1 Integrating OpenCV and Code::Blocks 23

3.2 Obtaining the default hue value array 24

3.3 Detecting the Lanes and the Obstacles 24

3.3.1 Capture of depth map image. 25

3.3.2 Finding the hue value of each pixel and find the difference with the corresponding hue value in the array. 26

3.3.3 Smoothening the image. 28

3.4 Path Planning 28

3.5 Communicating with Robot 29

3.5.1 Setting up of channel 29

3.5.2 Sending data 30

3.5.3 Recieving Data 30

4 Conclusion **32**

Bibliography **33**

List of Figures

1.1 HSV Color Model 5
1.2 Microsoft Kinect 7
1.3 Kinect description 7
1.4 Dots of laser from Kinect 8
1.5 Atmega640 Dev Board 8
1.6 Atmega128 Rapid Dev Board 9
1.7 Usb to Serial TTL logic converter 10
1.8 Iwave GPS Module along with the Antennae 11
1.9 X-Bee Pro wireless module 11
1.10 Duck Antennae 12
1.11 Vehicle Prototype 13
1.12 Mega torque Servo Motor 13
1.13 Li-Po Battery 11.1 V 1800m AH 14
1.14 USBASP Programmer 15

2.1 Initializing code for UART 17
2.2 HMC6352 connection with Atmega128 18
2.3 Layout of wireless communication 20
2.4 Code for transmitting data through UART 21
2.5 Code for receiving data through UART 21
2.6 Snapshot of the control code from CodeBlocks 22

3.1 Linker Settings 23

3.2 Compiler Search Directives 24
3.3 Linker Search Directives 24
3.4 raw depth map image with line for default hue value 25
3.5 Retrieving Depth Map Image From Kinect 25
3.6 Depth Map Image 25
3.7 RGB image 26
3.8 Depth Map Image 27
3.9 Top View 27
3.10 Smoothening Function 28
3.11 Camera View 28
3.12 Path Image 29
3.13 Signal flow Diagram 31

List of Tables

2.1 Configurations 20

Chapter 1

Introduction

1.1 Real Time Systems

In computer science, real-time computing (RTC), or reactive computing, is the study of hardware and software systems that are subject to a real-time constraint i.e., operational deadlines from event to system response. Real-time programs must execute within strict constraints on response time [1]. By contrast, a non-real-time system is one for which there is no deadline, even if fast response or high performance is desired or preferred. The needs of real-time software are often addressed in the context of real-time operating systems, and synchronous programming languages, which provide frameworks on which to build real-time application software. A real-time system may be one where its application can be considered (within context) to be mission critical. The anti-lock brakes on a car are a simple example of a real-time computing system the real-time constraint in this system is the time in which the brakes must be released to prevent the wheel from locking. Real-time computations can be said to have failed if they are not completed before their deadline, where their deadline is relative to an event. A real-time deadline must be met, regardless of system load.

1.2 Computer Vision

Computer Vision is a discipline in which techniques are studied to reconstruct, interpret, understand and represent a 3D scene from its acquired 2D images in terms of the general properties of the structures present in the scene. Computer Vision is the analytic study of motion dependent machines that extract valuable information from an image and perform some algorithmic processing on the captured images to derive necessary information to solve a particular or a set of tasks. Computer vision is the reconstruction of explicit, meaningful descriptions of physical objects obtained from their images. The outputs of computer vision are the descriptions or interpretation and some quantitative measurements of the object structures in the 3D scene. Image processing, pattern recognition and model based detection are many techniques that computer vision employs to achieve its goals. As far as scientific discipline is concerned, Computer Vision moreover concerns itself with the theoretical aspects related to artificial intelligent systems that extract valuable information from the captured images. The acquired image could be a single form of information or a collection of different forms and format. To be precise it could be a single format BRG image, a video sequence various formats, IR based camera image, laser scanned image, multiple images from multiple cameras or it could be fusion of multidimensional data from a single device, etc. Computer Vision aims to apply the theories and models relating artificial intelligent system to the construct and design computer vision based systems.

Why do we use Computer Vision in this project?

- Localization determined robot location traced autonomously.
- Obstacles avoidance purpose.

1.2.1 Role of CV in Robotics

Vision is essential for both humans and robots for providing detailed information and the reconstruction of the environment. It should be possible for a robust vision system to detect objects reliably and provide an accurate representation of the world for processing by the higher level processes. The vision system must necessarily be highly efficient, allowing a resource constrained agent to respond quickly and timely to a changing environment. Each frame acquired by a digital camera must be processed in a small, and fixed, amount of time. Algorithmic complexity is therefore a guiding factor, introducing a trade of between processing time and the quality of the information acquired. In most robotic systems, the vision system is the main perception medium and autonomous robots must be capable of using it in order to self localize and locate and identify the objects that they have to manipulate, and respond accordingly.

1.2.2 OpenCV (the image processing tool)

The OpenCV Library is mainly aimed at real time computer vision. Some example areas would be Human-Computer Interaction (HCI); Object Identification, Segmentation, and Recognition; Face Recognition; Gesture Recognition; Motion Tracking, Ego Motion and Motion Understanding; Structure from Motion (SFM); and Mobile Robotics. The OpenCV Library is a collection of low-overhead, high-performance operations performed on images. The OpenCV implements a wide variety of tools for image interpretation. It is compatible with Intel Image Processing Library (IPL) that implements low level operations on digital images. In spite of primitives such as binarization, filtering, image statistics, pyramids, OpenCV is mostly a high level library implementing algorithms for calibration techniques (Camera Calibration), feature detection (Feature) and tracking (Optical Flow), shape analysis (Geometry, Contour Processing), motion analysis (Motion Templates, Estimators), 3D reconstruction (View Morphing), object segmentation and recognition (Histogram, Embedded Hidden Markov

Models, Eigen Objects)[2].

1.3 Literature Survey

The first vehicle automation dates back to 1939, when Norman Geddes exhibited such cars in the New York World's Fair sponsored by General Motors [2]. Since then there has been several significant developments in this field [3, 4]. The vision task sequencer (VITS) by Turk *et. al.* [5] presents a vision system for the autonomous land vehicle (ALV) which in particular addresses the road following. The system builds a symbolic descriptions of the road and obstacles on the road. It uses both video and range sensors to accomplish the job. The authors have used various segmentation algorithms to identify the road in the video feed. They use the road boundaries to center the vehicle in the 3D space.

Luettel *et. al.* in their article [6] have presented an overview of the most current trends in the autonomous vehicles. They have discussed the concepts which are common to most successful systems and along with their differences. They have concluded with an optimistic note that robotics can help in improvements of vehicular traffics in terms of increase in highway capacity, traffic flow because of faster response time, less fuel consumption, and less pollution. They also hope to have less accidents due to foresighted driving and collision avoidance systems.

Google's driverless car is one of the most recent and publicized project [7]. This project being headed by Sebastian Thrun combines the information from Google street view with artificial intelligence, input from video cameras inside the car, sensor on top of the vehicle, radar sensor on the front, and position sensor attached to one of the real wheels to locate the car's position on the map.

Dolgov *et. al.* in their US patent [8] have presented a system for vehicle navigation avoiding obstacles in the presence of multiple objects. They have decomposed the area surrounding the obstacles with Voronoi diagram resulting in a number of edges. The route of the vehicle is determined using the minimum traversal cost from the current position of the vehicle to the adjacent area.

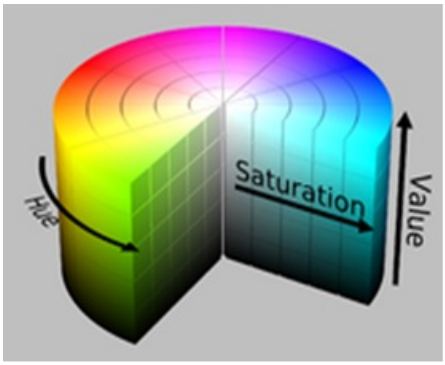

Figure 1.1: HSV Color Model

1.4 HSV color model

Color vision can be processed using HSV color space or RGB color space. RGB describes colors in term of the quantity of red, green, and blue present. HSV describes colors in term of the Hue, Saturation, and Value. In situations where color description plays an integral role, the HSV color model is preferred over the RGB model. The HSV model describes colors similarly to the way how the human eye tends to perceive colors. Colors in RGB are defined in terms of a combination of primary colors, where HSV describes color using more familiar comparisons such as color, vibrancy and brightness. [9, 10]

The angle around the central vertical axis corresponds to the “hue”, the distance from the axis is the “saturation”, and the distance along the axis is “value”, “lightness” or “brightness”. Hue represents the color type. It can be described in terms of an angle on the above circle. Although a circle contains 360 degrees of rotation, the hue value is normalized to a range from 0 to 255, with 0 being red. Vibrancy is the saturation of the color. Its value ranges from 0 to 255. Lower the saturation value, the more gray is present in the color, which causes it to appear faded. Value represents the brightness of the color. Ranges from 0 to 255, with 0 being completely dark and 255 being fully bright. White has HSV value of 0–255, 0–255, 255. Black has HSV value of 0–255, 0–255, 0. The dominant description for black and white is the term, value. The hue and saturation level

do not make a difference when value is at max or min intensity level.

To convert RGB to HSV, the normalized values of red blue and green are found. $r = R/(R+G+B), g = G/(R+G+B), b = B/(R+G+B)$

Each normalized H, S and I values are obtained by

$$h = \cos^{-1}\left[\frac{\frac{1}{2}\times\{(r-g)+(r-b)\}}{\sqrt{(r-g)^2+(r-b)(g-b)}}\right],\ h \in [0,\pi]\ for\ b \leq g$$

$$h = 2\pi - \cos^{-1}\left[\frac{\frac{1}{2}\times\{(r-g)+(r-b)\}}{\sqrt{(r-g)^2+(r-b)(g-b)}}\right], h \in [\pi,2\pi]\ for\ b \geq g$$

$$s = 1 - 3\times\min(r,g,b); s \in [0,1]$$

The ranges are set to [0,180][0,255], [0,255]

$$H = h \times 90/3.14$$

$$S = s \times 255$$

$$V = i \times 255$$

1.5 Goal and Objectives

1.5.1 Goal

Build a computer controlled robot which can traverse on a road with lanes and avoid the obstacles coming in front of it.

1.5.2 Objectives

Build a computer controlled robot which can traverse on a road with lanes and avoid the obstacles coming in front of it.

1. Building a mechanical robot which can travel on street road.

2. Designing a developing software which is capable of communicating between robot and computer.

3. Developing image processing software which can extract desired information.

Figure 1.2: Microsoft Kinect

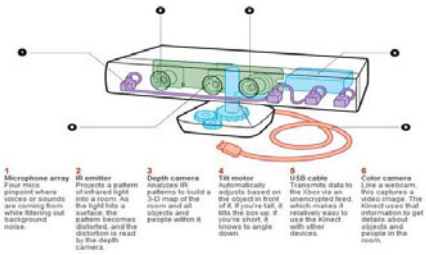

Figure 1.3: Kinect description

1.6 Hardware and Software Requirements

1.6.1 Hardware Requirements

Microsoft Xbox 360 Kinect Sensor

Kinect is a motion sensing input device by Microsoft for the Xbox 360 video game console and Windows PCs.

The Kinect has a color camera, which can act as a webcam, capturing frames at incredible speed of 30 frames per second. For depth sensing it has a IR emitter and Depth camera, which act together to give the depth map. To calculate the depth, the IR emitter projects a pattern of infrared light and the depth map analyzes the depth or 3-D map by estimating the time of flight of the infrared rays. The time of flight refers to the time taken by light to get reflected off of the scene and return to the depth camera.

The Infrared laser projects a grid of 50,000 dots and calculates the time of flight of each.

Figure 1.4: Dots of laser from Kinect

Figure 1.5: Atmega640 Dev Board

Atmega640 Development Board

It is used to receive driving signals from PC and Remote controller. It controls Servo Motor and the relays connected to the Vehicle circuit.

Its salient features are:

- Microcontroller : ATMEGA640 with 14.7456 MHz crystal
- 4 UART ports.
- 2.4GHz ZigBee (XBee) wireless module adaptor
- 2x16 Character Alphanumeric LCD
- USB to Serial converter at compatible with CMOS / TTL levels
- side high quality PTH PCB for added strength.

Figure 1.6: Atmega128 Rapid Dev Board

- Switches: Boot, Reset, Power
- Power : 7 to 15V,DC, Heat sink on 7805 for better current rating

Atmega128 Rapid Development Board

Is used to receive GPS coordinates from the IWave GPS module (Serially) and acquire Pin heading sensor (HMC6352) data using I2c protocol. It appends these two information and send as one byte character.

Its salient features are:

- Microcontroller : ATMEGA128 with 16 MHz crystal
- 2 UART ports.
- Connector for 8 servo motors
- Switches: Boot, Reset, Power
- 10 pin FRC connectors for all ports
- Power : 7 to 15V,DC, Heat sink on 7805 for better current rating
- Power : 7 to 15V,DC, Heat sink on 7805 for better current rating
- Reverse polarity protected

Figure 1.7: Usb to Serial TTL logic converter

USB to Serial TTL/CMOS Logic

This converter connects Atmega128 rapid development board to the Laptop.

Its salient features:

- USB powered
- Tx and Rx indicator LEDs
- Achievable data rates: 2400 to 115200 bps
- Serial pins available: TXD, RXD, CTS, DCD, DSR, DTR, RTS, RI and Ground
- 5V, 400mA and 3.3V, 30mA available for powering external device
- Used for serial communication involving flow control and for ISP for microcontrollers which uses serial port

IWAVE GPS KIT - ACTIVE EXTERNAL ANTENNA

This is the GPS module which provides the character string containing the Latitude and Longitude coordinates.

Its salient features are:

Figure 1.8: Iwave GPS Module along with the Antennae

Figure 1.9: X-Bee Pro wireless module

- External Antenna provides a very good location fixing
- Works at RS232 level as well as TTL level.

X-Bee Pro Long Range Wireless Module and RPSW antenna

X-Bee Wireless module provides the communication medium b/w the Atmega640 of the Vehicle and the Atmega640 of the Remote controller.

Specification:

Operating Frequency: ISM 2.4 GHz

- Antenna type: RPSW antenna
- Indoor/Urban Range up to 500 ft.

Figure 1.10: Duck Antennae

- Outdoor RF line-of-sight Range up to 2600meters
- Interface: Serial(UART) at 1200-115200 bps
- Supply Voltage: 2.8 - 3.4 V
- Transmit Current (typical) 215mA (@ 3.3 V)
- Idle / Receive Current (typical) 55mA (@ 3.3 V)
- Dimensions: 0.960 in x 1.297 in (2.438 cm x 3.294 cm)
- Operating Temperature: -40 to 85 C (industrial)

Features:

- AT and API Command Modes for configuring module parameters
- Compatible with Atmega640 Development Board

Vehicle Prototype Chassis.

To tackle the mechanical model a Toy car is purchased from the market and modified accordingly.

Specification of the toy car:

Figure 1.11: Vehicle Prototype

Figure 1.12: Mega torque Servo Motor

- 6V DC Motor 400 RPM connected to wheels with a gear assembly of ratio 7:1
- Requires 6V 4.5Ah Lead Acid Battery.
- Payload capacity - 20Kgs

Mega Torque Servo Motor for Steering.

A high torque servo motor is used to steer the front wheels.

Specifications:

- Required Pulse: 3-5 Volt Peak to Peak Square Wave.
- Operating Voltage: 4.8-6.0 Volts.

Figure 1.13: Li-Po Battery 11.1 V 1800m AH

Lithium Polymer (Li-Po) Rechargeable Battery 11.1V 1800mAH.

A light weight and small size battery. It is used to power the vehicle's on-board circuits and equipments. 2/3 Cell Lithium Polymer Balance Charger is used for charging this battery[8]

Its specific features are -

- Long life with full capacity for 1000 charge cycles
- 3X Li-Po 3.7V 1800mAh cells
- 192Grams Weight
- Volume: 10cm X 3.3cm X 2cm
- Discharge Current: 20 X 1800maH = 36Amp
- Max Charging Current: 1A

USBasp AVR Programmer.

A specially designed circuit to burn the program on AtmegaAVR microcontrollers. It has a USB based interface.

Figure 1.14: USBASP Programmer

Misceleneous.

Other miscellaneous materials such as connecting wires, USB extension cables etc. are also used.A specially designed circuit to burn the program on AtmegaAVR microcontrollers. It has a USB based interface.

1.6.2 Software Requirement.

(1) Windows Operating System

(2) OpenCV Library

(3) CLNUI Library for Kinect

(4) CodeBlocks IDE

(5) WinAVR library

(6) AVR Studio 4

(7) PC-MCU serial link driver

Chapter 2

Hardware Implementation

2.1 Initializing Microcontroller (Atmega640)

This microcontroller (μC) connects to the

(i) Servo Motor.

(ii) Relay connected to driving motor.

(iii) X-Bee Wireless module.

(iv) To the PC.

PWM signal is generated on the 2nd BIT of PORTJ so Input Pulse to the servo motor is given from PIN J2. Relay is actuated using the onboard Motor Driver IC(L293d) which is connected to 2,5,6,7 bits of PORTA.X-Bee wireless module is interfaced through UART port so it is connected to UART1.and finally the connection to PC is established using another UART port, so PC - μC serial link module is connected to UART0. UART is initialized for the transmission of data between computer and the robot. The baud rate is set to 9600.

2.2 Initializing Microcontroller (Atmega128)

This μC is connected to the

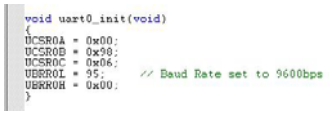

Figure 2.1: Initializing code for UART

(i) I Wave GPS Module .

(ii) PIN heading Compass Module.

(iii) To the PC.

IWave GPS module gives the information bytes serially so it is connected to the UART1 of the Atmega128, while Compass Module (HMC6352) gives the information through the I2C protocol so it is connected to PD0 and PD1 pins of PORTD. As mentioned above in the Atmega640 section PC to Robot communication is done using UART0 and it is initialized in the same way to Baud Rate of 9600.

2.2.1 Communicating with Pin Heading Compass Module (HMC6352)

The Honeywell HMC6352 Pin Heading sensor is a fully integrated compass module which combines the 2-axis magneto-resistive sensor with the required analog as well as digital support circuit, microprocessor and algorithms for the heading computation. It uses the I2C protocol to communicate and as Atmega128 has the capability to handle I2C protocol devices so it is connected to the HMC6352 module.

2.2.2 I2C Communication Protocol

The HMC6352 communicates with the Atmega128 interface module via a two-wired I2C bus system as a slave device to the master controller Atmega128. The HMC6352 pin heading sensor uses a layered protocol to communicate with

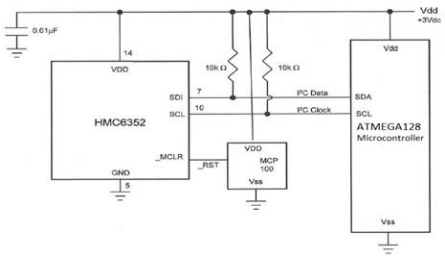

Figure 2.2: HMC6352 connection with Atmega128

the interface protocol i.e. Atmega128 defined by the I2C bus specifications, and the lower end command protocol defined by Honeywell. The data rate for communication is the standard-mode 100kbps (high speed) rate which is defined in the I2C Bus Specifications. The bus bit format consist of an 8-bit Data/Address sent and gets back a 1-bit acknowledge bit. The format of the data bytes are case sensitive ASCII characters or binary form data to the HMC6352 pin heading slave, and binary form data returned. Negative binary values will be represented in two's complement form. The default (or default factory set) HMC6352 7-bit slave address is set as 42(hexadecimal) for all write operations and 43(hexadecimal) for read operations. Both the HMC6352 Serial Clock Line (SCL) as well as Serial Data Line (SDA) does not have the internal pull-up resistors and also the required resistive pull-up (Rp) between the master device (Atmega128 microprocessor) and the HMC6352. The pull-up resistance are of values of 10k ohms are recommended in addition with a nominal 3.0-volt supply voltage as shown in figure 1. The SCL as well as SDA lines in the bus can also be connected to a number of devices. The bus can be with a single master to multiple slaves or it can have a multiple master configuration. All data transfers are only initiated by the master device/devices which are responsible for generation of the clock signal (SCL) and the data transfer is of 8 bit length. All the devices are addressed with a unique 7 bit address by I2C. After every 8-bit transfer, the master device always generates the 9th clock pulse and frees the SDA line. The destination device (receiving slave device) will make it to pull the SDA line low (logical low 0) to acknowledge (ACK signal) the successful transfer of data or leave the SDA high as it was to send negative

acknowledge (NACK).

As per the I2C specifications, all transitions taking place in the SDA line must occur when SCL is in low state level. This leads to two different conditions on the bus concerned with the SDA transition when SCL line is high. When master device pulls the SDA line low while the SCL is high, it indicates the starting condition (S), and the stopping condition (P) is marked when the SDA line is pulled to high state while the SCL is high. The I2C protocol allows for the restart (R) conditions in which the master device would issue a second start condition on succession without issuing a stop.

Reading bytes using I2C protocol

In all bus transactions, it begins with the master device issuing signal (S) to start sequence followed by the particular slave address byte. The address byte sent contains the slave address in which the 8 bit address has upper 7 bits and the Least Significant Bit distinguished. The Least Significant Bit of the address byte designates if the operation carried out is a read LSB=1 and a write operation LSB=0. At the start of 9th clock pulse, the receiving (particular) slave device will reply with the issue of the ACK or NACK. Following these sequential bus events, the master will send the data bytes for a write operation to be carried out or the slave node will transmit back the data for a read operation. All bus transactions are terminated with the master issuing a stop sequence.

Calculating Heading direction from bytes obtained from HMC6352

The serial data obtained from the HMC6352 is of two bytes i.e.16 bits of data is obtained from the sensor which is collected in form of character. The two characters obtained are denoted as A and B. Upon receiving the two characters serially using I2C protocol on the bus, heading value is obtained by multiplying A to 256 and then adding the resultant with B and then the resultant will be divided by 10 to get the angle of heading.

Real Heading Angle $H = (A \times 256 + B)/10$

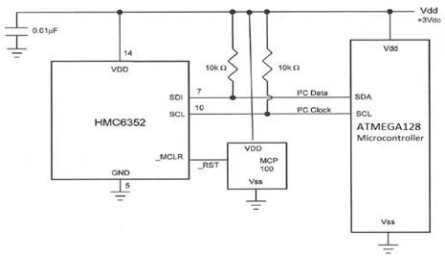

Figure 2.3: Layout of wireless communication

Real Heading Angle(H) lies between 0 to 360 degrees.

2.3 Initializing E-stop Remote

In case of an emergency the vehicle can be stopped wirelessly. When the E-Stop button is pressed, breaking instruction is sent to the motors is generated which causes the vehicle to stop. The wireless communication is done by using XBee Pro 24 ZNET RPSMA module with a range up to 6 miles (with strong antenna) These modules are interfaced with a microcontroller and communication between two microcontrollers is done as shown in fig.

Table 2.1: Configurations

Properties	**Module 1**	**Module 2**
Source Address	13A200	13A200
Destination Address	4050AF12	4050AF1B
PAN ID	1234	1234
Baud Rate	9600	9600

The Source Address of one transreceiver should be the Destination Address of other in one to one communication. PAN ID should be a 4 digit number and should same for both transreceiver. The Baud rate is set to 9600 bps.

Figure 2.4: Code for transmitting data through UART

Figure 2.5: Code for receiving data through UART

2.4 Data sending

For transmitting data we have to wait till the 1st bit on UCSR0A is not equal to 1. First bit on UCSR0A indicates whether output buffer is empty or not. Once output buffer is empty 8 bit data is put in the UDR (UART data register) for transmission.

2.5 Data receiving

For receiving data we wait till 4th bit on UCSR0A is not equal to 1. 4th bit on UCSR0A indicates whether input buffer is empty or not. Once output buffer is full we can take 8 bit data from the UDR (USART data register).

2.6 Driving Motor Movement

For driving the Wheel motors we have to actuate the relays which are connected to L293d IC (onboard IC n Atmega640 Board). The input to this IC is connected to 3,5,6,7 PINs of PORTA. So the given code has the functions which on calling will move the bot forward, backward or stop.

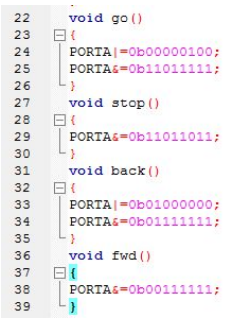

Figure 2.6: Snapshot of the control code from CodeBlocks

2.7 Robot Control

The driving signals are given to the Robot via Microcontroller board i.e. PC gives signals to the Atmega640 μC: which has the control over the driving motors and the servo motor. Hence we control the Robot from the PC.

Chapter 3

Software Implementation

3.1 Integrating OpenCV and Code::Blocks

OpenCV 2.0 is integrated with Code::blocks. After the installation of Code::Blocks is complete, the Compiler and debugger setting are changed to integrate OpenCV. The compiler is set to GNU GCC Compiler. Then to the linker settings, the library files are added, so that the processes are linked to these libraries while execution.

In the Settings → Compiler and debugger window, Linker Settings are changed by adding the all the following directories one by one.

In the Global compiler settings window, directory search is modified by adding the following directories to Search directories → Compiler

The Linker settings are modified by adding the following directory to the tab

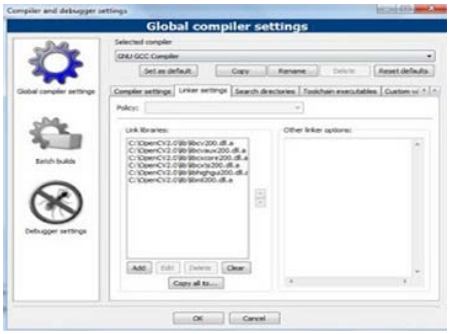

Figure 3.1: Linker Settings

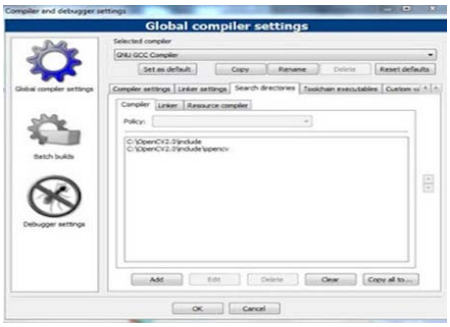

Figure 3.2: Compiler Search Directives

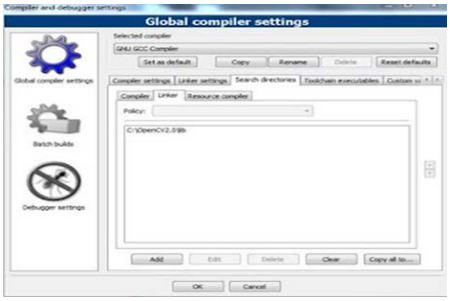

Figure 3.3: Linker Search Directives

"Search Directories→Linker"

3.2 Obtaining the default hue value array

For thresholding we have to make an array containing the hue values of the pixels between the (320,0) and (320,480) in no obstacle state.

3.3 Detecting the Lanes and the Obstacles

The first problem in solving the given problem statement is obtaining the depth map image from the Kinect sensor and process it for the non-traversable regions.

The following sub-steps have to be followed to perform this step

(1) Capture of depth map image.

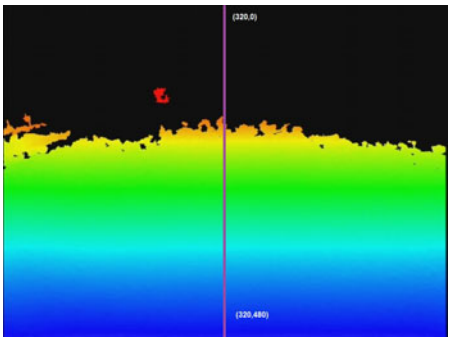

Figure 3.4: raw depth map image with line for default hue value

```
PBYTE rgb32_data = (PBYTE) malloc(640*480*(24/8));
PDWORD depth32_data = (PDWORD) malloc(640*480*4);
CLNUICamera cam = CreateNUICamera(GetNUIDeviceSerial(0));

StartNUICamera(cam);

while(1)
{
GetNUICameraColorFrameRGB24(cam, rgb32_data); //cam
GetNUICameraDepthFrameRGB32(cam, depth32_data);  //depth

        rgb32 = cvCreateImageHeader(cvSize(640,480), 8, 3);
        depth32 = cvCreateImageHeader(cvSize(640, 480), 8, 4);

        cvSetData(rgb32, rgb32_data, rgb32->widthStep);
        cvSetData(depth32, depth32_data, depth32->widthStep);


        cvCvtPixToPlane(depth32,red,green,blue,alpha);
        cvCvtPlaneToPix(red,green,blue, 0, depth);

}
```

Figure 3.5: Retrieving Depth Map Image From Kinect

(2) Finding the hue value of each pixel and find the difference with the corresponding hue value in the array .

3.3.1 Capture of depth map image.

The depth map image is captured from the Kinect Sensor using command in OpenCV as shown in fig.

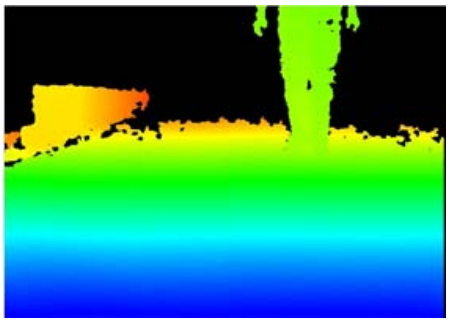

Figure 3.6: Depth Map Image

Figure 3.7: RGB image

3.3.2 Finding the hue value of each pixel and find the difference with the corresponding hue value in the array.

The depth map obtained from Kinect is used for object detection and path planning. In this approach, the lanes, the static as well as dynamic obstacle are all seen as obstacles to be avoided. The depth map from the depth map is inferred in terms of the color, i.e. the hue value reduces from 360 to eventually 0. The distance can be found out by scaling the hue value. In order to detect obstacles, the depth map is converted to an overhead view, which would make the finding of correct path easy. The depth map is divided into segments and each segment is analyzed. For obstacle detection, main emphasis is given on the upper half segment, as the lower segment is taken as the lower segment is specified for road. The lower half is analyzed for small obstacles, humps or lane detection. The hue value is scaled and a distance from top of the image is assigned to it, representing its distance from the vehicle. The raw overhead view fluctuates very much and needs to be stabilized. This is done using Kalman filtering. This gives a stable top view, which assists in path finding.

The colored areas in the top part of the overhead view image are the obstacles to be avoided. The correct path to be followed is estimated by finding the path which has enough clearance for the vehicle to move and avoiding the obstacles simultaneously. For this purpose, the overhead view image is divided

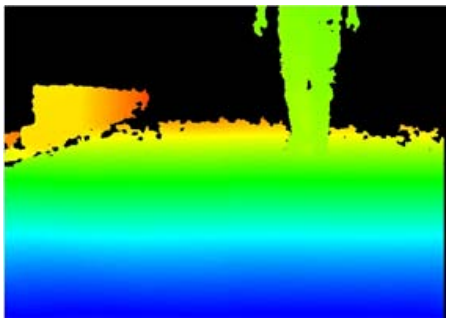

Figure 3.8: Depth Map Image

Figure 3.9: Top View

```
//smoothing
// used types-gaussian and median
cvSmooth(imgsource,imagefinal,Gaussian/Median, x,y);    x,y are odd numbers max upto 11
```

Figure 3.10: Smoothening Function

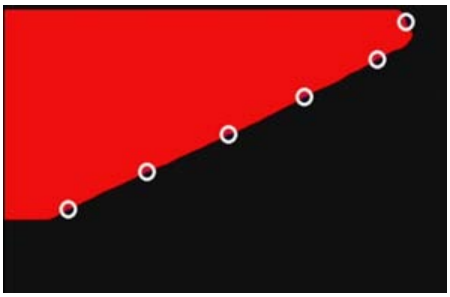

Figure 3.11: Camera View

into segments, and the open paths are estimated at each segment.

3.3.3 Smoothening the image.

3.4 Path Planning

The boundaries of the obstacles are found using edge detection. The image is first threshold for the color of the obstacle from the overhead view. Then the threshold image is passed through edge detection, giving us edges or the outline of the obstacles. The Outline or edge detected image is divided into 10 segments regularly distanced vertically in the image and search for the outline points in those y- coordinates are done to determine the x-coordinates. This is done by taking a blank image with binary white lines drawn at those y-coordinates and performing the 'and' operation between this image and the threshold image. This gives us the intersection between the outlines in both images, the points of interest. To avoid multiple continuous points blob detection is used, which uses connected component analysis, and if a set of connected points are found, they are removed from their being candidate for points of interest.

As shown in the figure red circles are obtained and a final path line is drawn

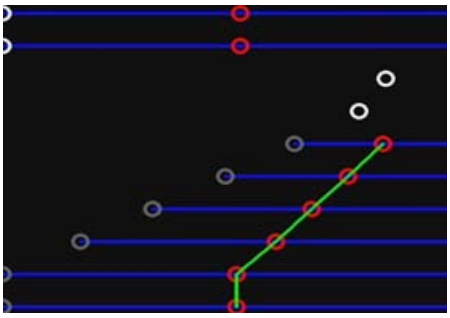

Figure 3.12: Path Image

from the the angle of steering is determined.

3.5 Communicating with Robot

Once the lanes are detected and the angle and direction of motion are established, the values are serially communicated to the robot.

Following steps are involved in setting up the serial communication:

(1.) Setting up of channel

(2.) Sending data

(3.) Receiving data

3.5.1 Setting up of channel

Here we create a handle on pcCommPort (Com4) with read/write exclusive access and default security attributes.

Its attributes are then set to

BaudRate = 9600,

ByteSize =8,

Parity = NOPARITY,

StopBits = ONESTOPBIT

$HANDLEhCom = CreateFile(pcCommPort, GENERIC_READ|GENERIC_WRITE,$

$0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)$

3.5.2 Sending data

The following code is used to send data through the serial port. voidSerialPutc (HANDLE *hCom, char txchar)

{
BOOL bWriteRC ;
staticDWORD iBytesWritten ;
bWriteRC = WriteFile(*hCom, &txchar , 1,&iBytesWr i t ten ,NULL) ;
if(!bWriteRC)
printf(" error ") ;
return ;
}

3.5.3 Recieving Data

The following code is used to receive data from the serial port.

charSerialGetc(HANDLE *hCom)
{
char rxchar ;
BOOL bReadRC;
static DWORD iBytesRead ;
bReadRC = ReadFile (*hCom, &rxchar , 1 , &iBytesRead , NULL) ;
if(! bReadRC)
printf(" error ") ;
return rxchar ;
}

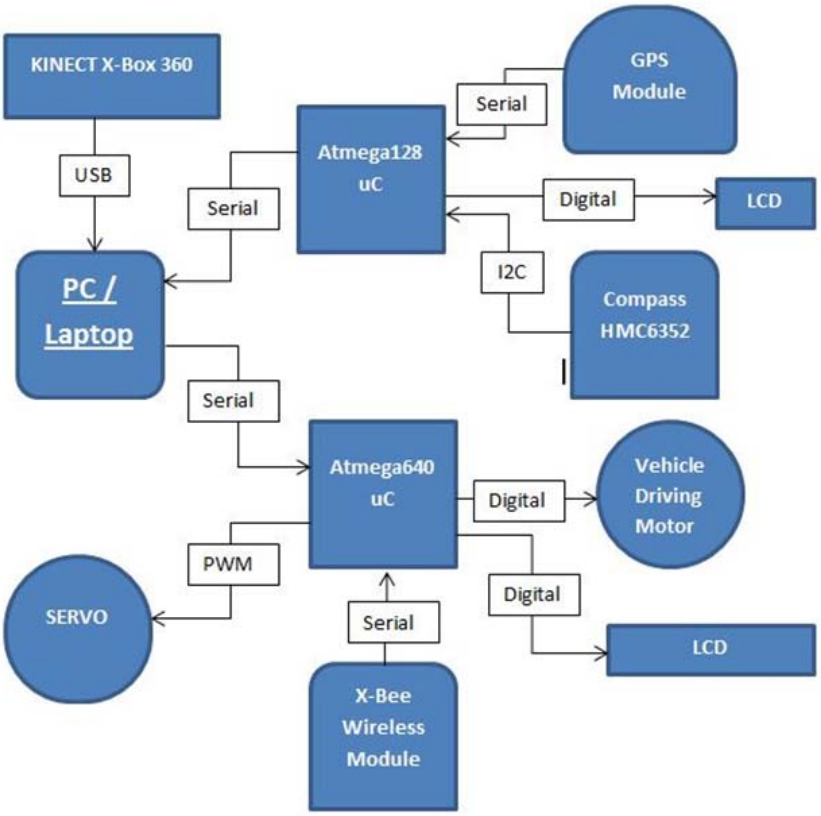

Figure 3.13: Signal flow Diagram

Chapter 4

Conclusion

An automated vehicle was designed with all the mentioned design parameters as described in the section 2 earlier. The vehicle was put to tests under varied circumstances which could complete the above stated task under certain circumstances. Certain problems relating to detection and concluding actions could be improved further by use of the learning algorithms. The purpose of guiding vehicle in presence of static and dynamic objects gave tremendously good result which could also be applied in application for example as in case of automated cars. This could also be extend its stance in field of automated transport of industrial goods in industries as well as for physically handicap by use automated guided wheel chair for navigation to destination. Future works could include more robust algorithms to improve the traversal algorithm by learning and taking into account of the very rough road condition in which travelling is a bit difficult. Auto calling system could be attached for supervision of status by people at remote places.

Bibliography

[1] M. Ben-Ari. *Principles of Concurrent and Distributed Programming*. Prentice Hall, 1990.

[2] NYWF. The 1939 new york world's fair. `http://www.1939nyworldsfair.com/worlds_fair/index.htm`.

[3] Douglas W. Gage. Ugv history 101: A brief history of unmanned ground vehicle (ugv) development efforts. *Unmanned Systems*, 13:9–32, 1995.

[4] D.A. Forsyth and J. Ponce. Computer vision: a modern approach. 2002.

[5] M.A. Turk, D.G. Morgenthaler, K.D. Gremban, and M. Marra. Vits-a vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:342–361, 1988.

[6] Thorsten Luettel, Michael Himmelsbach, and Hans-Joachim Wuensche. Autonomous Ground Vehicles - Concepts and a Path to the Future. *Proceedings of the IEEE*.

[7] Time Techland. Sebastian Thrun, `http://techland.time.com/2012/05/08/googles-driverless-cars-now-officially-licensed-in-nevada/`.

[8] Dmitri A. Dolgov and Sebastian Thurn. Vehicle navigation system with obstacle avoidance. Patent, US 2009/0105939 A1, April 2009.

[9] Charles Poynton. *Digital Video and HDTV Algorithms and Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1 edition, 2003.

[10] Rafeel C. Gonzalez and Richard E. Woods. *Digital Image Processing*, volume ISBN 978-81-317-2695-2. Prentice Hall PTR, 2088.

[11] *Intel OpenCV Reference Manual*, 1999-2001. `http://developer.intel.com`.

[12] J. Davis and Bobick. The representation and recognition of action using temporal templates. Technical Report 402, MIT Media Lab, 1997.

[13] Gary R. Bradski and James W Davis. Motion segmentation and pose recognition with motion history gradients. *Machine Vision Applications*.