GESTURE ASSISTED ROBOTIC CAR (GARC)

A PROJECT REPORT

Submitted by

AKSHAY AGARWAL (05114802710)
DEVASHEESH CHOUDHRY (06414802710)
NISHANT GOEL (05814802710)
SANCHIT HANDA (05014802710)
TUSHAR GUPTA (04914802710)
VINAMRA KUMAR SINGH (01914802710)

under the guidance of

Dr. Namita Gupta

Head of Department (CSE)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

PSP AREA, SECTOR-22, ROHINI , NEW DELHI – 110085

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

ABSTRACT

The project GARC or Gesture Assisted Robotic Car has been developed in two phases. Phase-I of the project demonstrates the abilities to control a car remotely with the help of hand gestures. Phase-II on the other hand simulates, through the use of an interactive GUI, the remote controlling of car, and automated driving of a car which covers areas like path search between source and destination, speed control, collision detection and tracking etc.

The project falls under the category of Artificial Intelligence and utilizes digital image processing as well. The car has been developed using a microcontroller, Arduino Uno, which acts as the brain of the car and is responsible for listening in to the commands from the remote user and carrying out apropos actions.

The main achievements in this project are the successful development of a remote car with minimal lags, the fact that the notion of automated driving has been reasonably simulated, and high degree of user friendliness of the overall application is there e.g. the gestures input are much like driving a car using a steering wheel.

Though further additions could be made to the car in order to better equip the user for remote driving, however they were not possible in the current project due to temporal and financial constraints.

CERTIFICATE

This is to certify that the project report entitled **GESTURE ASSISTED ROBOTIC CAR** being submitted by **Mr. Akshay Agarwal, Mr. Devasheesh Choudhry, Mr. Nishant Goel, Mr. Sanchit Handa, Mr. Tushar Gupta** and **Mr. Vinamra Kumar Singh** in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Maharaja Agrasen Institute of Technology is a record of bonafied work carried out by them under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Dr. Namita Gupta

Head of the Department, CSE

Maharaja Agrasen Institute of Technology

Delhi, India

DECLARATION

We hereby declare that all the information provided in this documentation of the project GARC is true to the best of our knowledge. We declare that all due credit has been given to research papers and people where required and if any form of plagiarism is found, then action be invoked against us.

Place: Delhi

Date:

Akshay Agarwal
(05114802710)
(8C123)

Devasheesh Choudhry
(06414802710)
(8C123)

Nishant Goel
(05814802710)
(8C123)

Sanchit Handa
(05014802710)
(8C123)

Tushar Gupta
(04914802710)
(8C123)

Vinamra K. Singh
(01914802710)
(8C123)

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this project and who influenced our thinking, behaviour, and acts during the course of the project.

We are thankful to **Dr. Namita Gupta** for her support, cooperation, and motivation provided to us during the project development for constant inspiration, presence and blessings.

We also extend our sincere appreciation to **Ms. Rekha Singla** who provided her valuable suggestions and precious time in accomplishing the project report.

Lastly, we would like to thank the almighty and our parents for their moral support and friends with whom we shared day-to-day experiences, and received lots of suggestions that improved the quality of the work.

Akshay Agarwal

Devasheesh Choudhry

Nishant Goel

Sanchit Handa

Tushar Gupta

Vinamra Kumar Singh

TABLE OF CONTENTS

CHAPTER 1

Introduction

The project utilizes a mix of artificial intelligence and digital image processing in both phase-I and phase-II. On an applied basis the project scope extends to many categories such as assistive technologies for the people with special abilities, military applications for small robotic cars used in bomb diffusal and performing recon missions, also automated cars on road can be somewhat easily managed as the concept of nonchalant driving will be eliminated.

In this project the car developed uses an Arduino Uno microcontroller which is the brain of the car and is also responsible for listening to and interpreting the messages received by the car from the user. In this a L293D IC has been used to control the car motors.

The gesture application has been developed using C++ and OpenCV libraries. C++ has been used in order to minimize the lags and hasten the processing. OpenCV libraries support the digital image processing part in C++. In this application, keeping user friendliness in mind, the idea of hand movement similar to steering wheel is there.

The simulation is used to show concept of automated driving involving path searching on a map, speed control and collision detection and tracking. The simulation has been developed using Java platform to facilitate the development of GUI and the fact that languages such as C++ need not have been used and a sufficiently high level language would have been a reasonable choice.

This project has been developed in order to demonstrate the many uses such technology may have, and also to show that the development of such cars is not a gargantuan task. Moreover, this project was undertaken as a test of our abilities, to incorporate our skills in a single platform, and to learn to design and develop real time projects.

Much work is being done in the field of automated driving such as car Stanlee and Cherry developed by the researchers at Stanford University in association with Google. Leading car giants such as Audi, Mercedes and Chevrolet also have active programs and ongoing research in this field. However, the work in this domain in India is still very limited. Moreover, GARC attempts to develop a smaller model which can be used in above mentioned areas especially the first two.

In subsequent chapters we are going to be discussing about the references used for developing this project, how the project has been structured, the nature of information flow etc. We are also going to discuss about the testing that was carried out to validate and verify the project, the issues that we faced and the different approaches we took in order to handle those issues.

CHAPTER 2

Background and Literature Survey

Gesture detection in this has been done using two colors. These colors are fixed, i.e. the color blue and green are the only ones that can be used and cannot be substituted by the user.

The gesture application takes input in video form from the webcam of the remote controller, upon receiving this input it further processes each frame to perform gesture detection.

The research done involved reading about the existing systems to gain an insight into the current work done and a basic idea about the path to be taken for development of the system This research also helped us realize the problems with the systems already developed, the resources that we might need and thus we made an effort to develop a better system.

Below are a fraction of the research papers we read and followed, these were the major ones and our background research involved a lot more.

[1a]This was the first paper we read, being a thesis it gave us a good idea about the basics of the system. It also laid the foundation on the basis of which we gained knowledge about the concepts involved.

[1b][1c]These papers inspired us to go for fingertip detection rather than develop a hand gesture detection system. [1b]Here the author has develop a muti-touch based system, even though it was not based on gestures, or specifically air gestures, however it provided excellent basis for developing a fingertip detection based system. The advantages that fingertip detection pose are numerous, starting with the fact that the number of gestures that can be there are more, to the fact that the system can be actively used for supporting computing for people with disabilities.

[1d]This paper initially helped us when we were on the track of skin detection. However the main drawback remained that in this the approach taken involved the use of depth sensor equipped camera, which didn't meet our aim of developing the system using low cost technology.

[1e]This is the paper that we mainly paved way for development of the project.

Following is the background and the gist of the research that was done. The main fact that we established while researching is that Microsoft Kinect is one the best ways to develop such a kind of system. The fact that it has superior imaging power coupled with depth sensors and a dedicated API that helps you make vision based computing a reality. Other than the features it provides, it also has a very large developer community. Many functions to facilitate skin

detection, hand detection, motion detection etc. are already present and thus need not be coded explicitly.

Other than this there is OpenCV. This too is designed to facilitate vision based computing and supports multiple languages, most commonly used with this is C#. This is the technology that we have used in our system, however it has been done with C++.

MATLAB is also a viable candidate for such an application, and it provides a lot more ease of use coupled with more versatility. However, we see that MATLAB takes a lot of time to perform simple tasks and thus could not have been used in the current scenario.

The research done involved reading about creating robotic cars and the microcontrollers used by them to gain an insight into the current work done and a basic idea about the path to be taken for development of the project. This research also helped us realize the problems with the systems already developed, the resources that we might need and thus we made an effort to develop a better system.

CHAPTER 3

Design

3.1 Phase I: Gesture Recognition with Hardware

The Project GARC is an implementation of the process as shown in the following diagram.

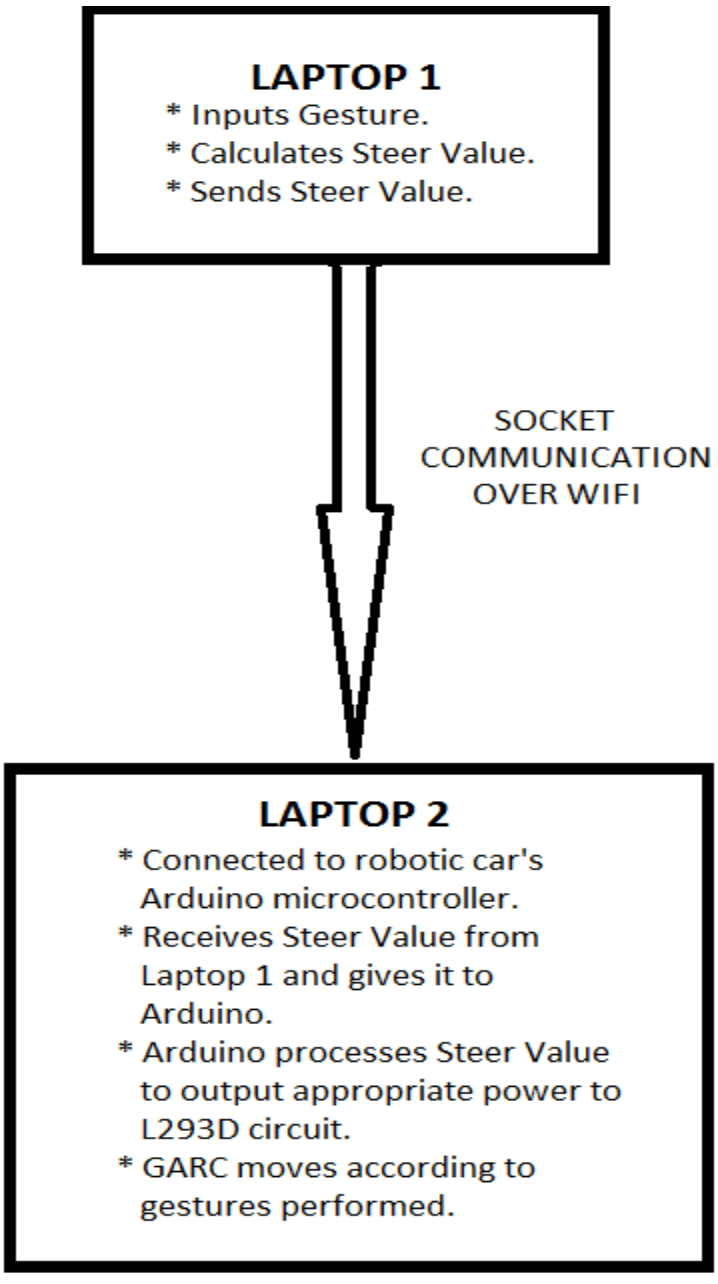


Fig 3.1: Project Structure

The above block diagram depicts the methodology followed to construct Project GARC.

In the above diagram, it is understood that 2 laptops are involved in the project.

Laptop 1 inputs the gestures given by the user through the laptop's webcam.

According to the gestures performed by the user, the steer value is calculated which determines the relative speeds of the two hind tires for the movement of the robotic car. For e.g. for taking a right turn, the left wheel rotates faster than the right wheel and vice versa. For moving straight, both wheels rotate with equal rotations per minute. These steer values are continuously calculated for various positions of the user's hands in real time thereby giving the look and feel of driving an original vehicle.

These values are transmitted to Laptop 2 over a WiFi connection by the method of socket communication between the ports of the two laptops.

At Laptop 2, the Arduino microcontroller of the robotic car is connected to laptop2 by a USB cable. The steer values coming from Laptop1 are input to Arduino by Laptop2. The Arduino processes the steer values to calculate the power to be given to the two motors of the robotic car. These values are inputs to the motor driving circuit which is controlled by IC L293D. The desired power is output to the motors and the tires of the car rotate according to desired power.

Thus, GARC moves according to the real time gesture inputs of the user.

3.3 Phase II: Car Automation System Simulation with Gesture Recognition

The elements of the AI simulation are discussed below:

- **Arena Processing**
- **Path Planning**
- **Path Smoothing**
- **Controller**
- **Car Motion**
- **Car Tracking**

Arena Processing: Given an arena image of any size (5:2), the application groups the image pixels into the following types of grid cells:-

- Obstacles (Black)
- Danger Area (Gray)
- Free Area (White)

Through in arena processing, we identify the free areas in which the autonomous car can move obstacles which are strictly no-no and danger areas which are close to the obstacles but car can travel on it.

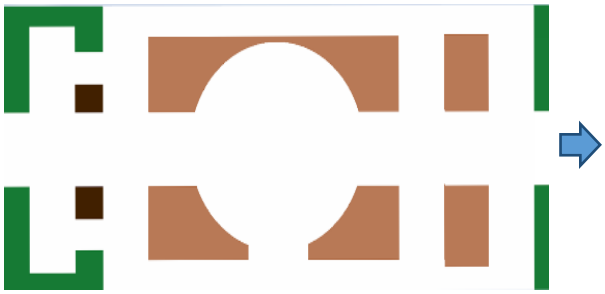


Fig 3.2: Arena Image (5:2)

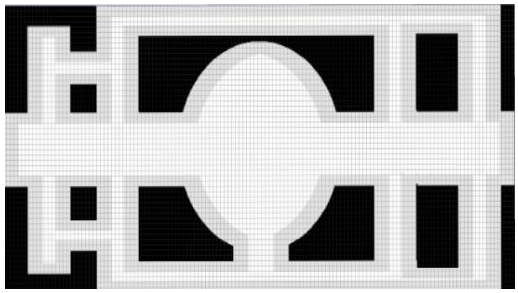


Fig 3.3: Grid Cells (600*600)

Path Planning and Smoothing: It is the process wherein the system finds the shortest, or rather the safest path between the source and destination provided to the system. We have used A* Search algorithm for path searching along with Manhattan Heuristics with Obstacle Pressure. Two Steps smoothing algorithm is used to produce a smooth path from search path on which the car model can travel easily.

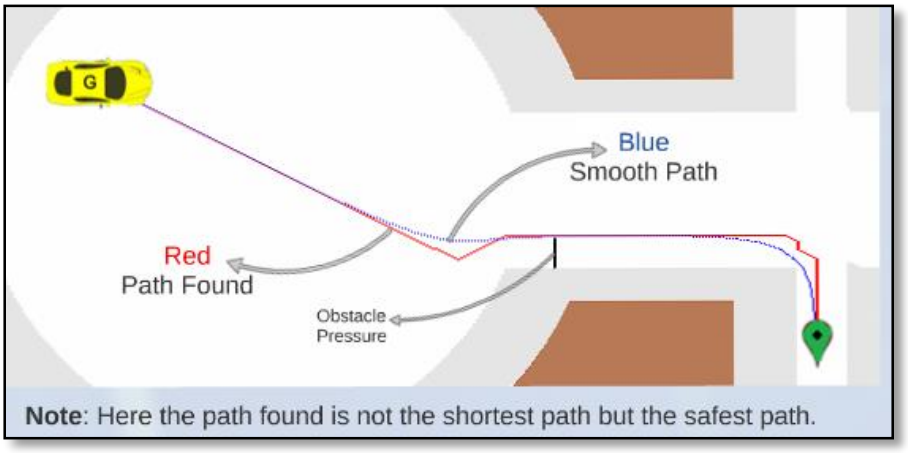


Fig 3.4: Path Planning & Smoothing

Controlling: It is the control mechanism of the robot car model, through which its motion is controlled, like steering control, acceleration control.

Steering Control:
Car steering depends upon the value of cross track error and differenial cross track error:

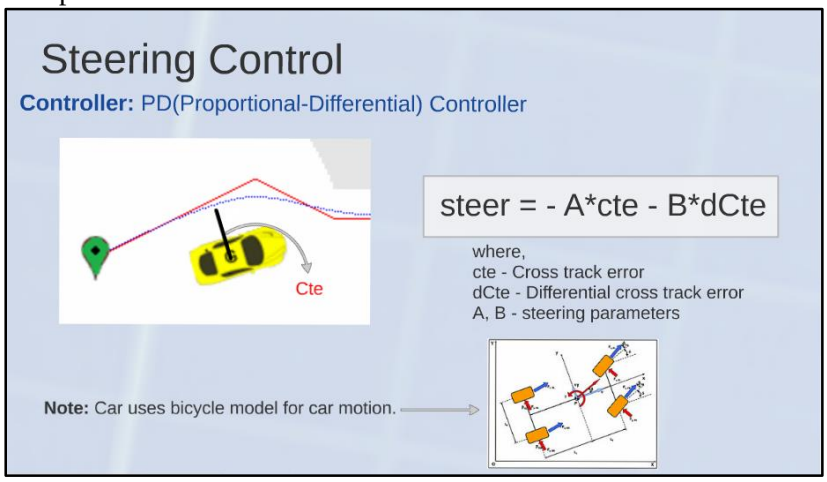


Fig 3.5: Steering Controller

Acceleration Control:
Car uses three speed model for driving:

- Max Speed
- Min Speed
- Reverse Speed

Speed Formula:

speed = (1.0 - abs(steer)/maxSteer) * maxSpeed

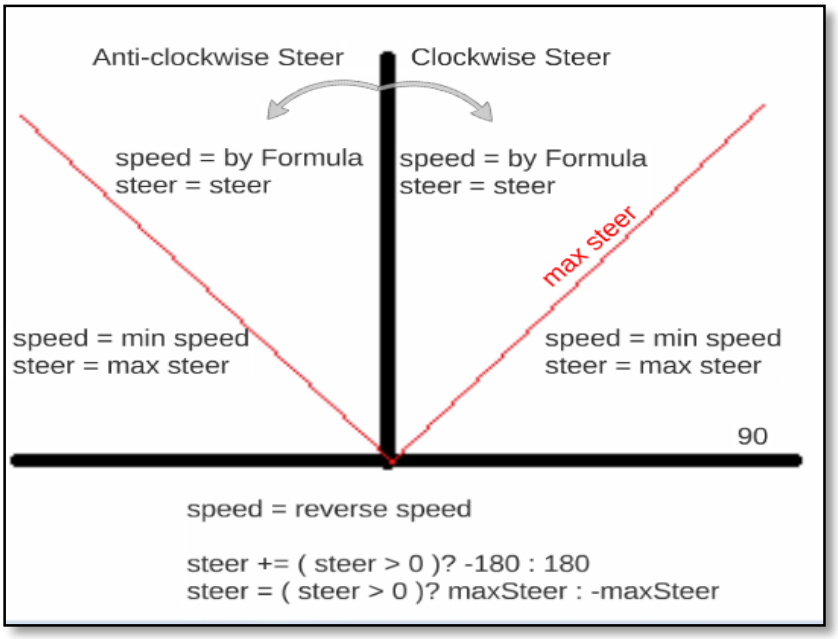


Fig 3.6: Speed Controller

Car Tracking

It is collision avoidance mechanism where the robot car model detects other cars in the environment. We have used elliptical car tracking approach for collision detection.

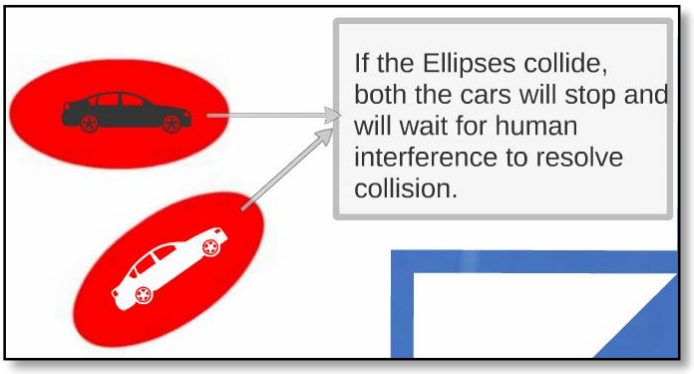


Fig 3.7: Car Tracking

Equation used for 3D Elliptical Car Tracking:

$$A(x-h)^2 + B(x-h)(y-k) + C(y-k)^2 = 1$$

Where,

(h,k) – center of ellipse

(x,y) – Any point on ellipse

A,B,C – Ellipse parameters

CHAPTER 4

Implementation

4.1 Phase I: Gesture Recognition with Hardware

Gesture Recognition

In the gesture detection application, there are 3 main components of the program.

1. Input video from the user webcam.

2. Take the frames input and detect the red and green colored components.

3. Based on orientation draw a line and print it out indicating the gesture.

4. Send this slope to the simulation or the car.

The implementation was first done using MATLAB, however it posed significant problems. Though MATLAB provides great ease of use, however it is relatively very slow and inefficient. MATLAB takes about four times the time taken by C++ to load a frame, the processing takes even greater time.

However, the prototype of the system was still developed in MATLAB so as to check the algorithms used and make changes in order to increase the efficiency.

The final implementation was done in C++ using OpenCV libraries which provide support for image processing. Continuous frames are taken using the webcam of the machine, once a frame is acquired, its resolution is checked and it is sent to another function for further processing.

From the frame, we first extract the required colored components. This extraction of the colored components is done on the basis of HSV i.e. hue, saturation, value rather than RGB scale. We don't use RGB color scale since color differentiation on that scale is difficult.

Initial model used skin detection and further k-curvature in order to detect hands. However, as shown in the test cases the problem with doing so is that skin detection is highly inaccurate. This is further complicated by lighting and illumination errors which are commonly observed in digital image processing.

Hence, we switched to color detection using colored objects. In this we detect the blue and the green color. The nature of this selection is based on the fact that both these colors are quite prominent and hence easy to detect. Prominence is expressed in terms of immunity to lighting and illumination errors.

After the colored components have been extracted, we further perform edge extraction using Canny edge extraction algorithm. In this if multiple components of the blue or the green color are detected then it poses a problem. The algorithm holds true for a single object detection, however there is a possibility that the wrong object may be detected.

This error can be removed by employing different approaches:

- The first approach to error removal involves that a specific yet simple pattern be present on both the colored objects, using this we can AND the results of pattern detection and color detection in order to identify the correct objects. However, the problem with this approach is that pattern detection is a relatively heavy process and with the processing power currently available to us it may lead to lags in the program, thus defeating the purpose of being real time.
- The second approach and perhaps a more sensible one, would be to use a camera with depth sensor, this way we can either fix the range of the distance of objects from the cam, or we can go with detection of the closest colored objects. However, it is evident that it poses its own set of problems.
- The best approach would be to actually combine the above two, and actually give the user to accept or reject the detection. In this we primarily use the second approach, however if it fails then we move on to pattern detection, this would ensure that pattern detection is used only when necessary.

Finally after the extraction of the colored components and their edge detection, we further process this to find the centroid of both the objects. Since we need to find the orientation of the two objects in order to determine whether or not the car should turn and by how much, thus we see that we are not actually gestures. Gesture detection has been technically defined as the process of detecting a fixed no. of gestures, however since in this case the number of gestures would turn out to be a lot, thus the traditional approach of machine learning is discarded.

After calculation of the centroid, we thereafter draw a line between the centroid of the two objects and calculate the slope of this particular line. This slope is then transmitted via socket communication to the simulation program or the car as required.

The major hurdle in this project was identification of the technology to be used in order to reduce the processing time and make it more suitable to real time systems. After selecting C++ to be used along with OpenCV libraries, another major problem faced was insufficient developer support.

Thus it took a lot of time to delve into understanding the functions of OpenCV and implementing them in C++.

Hardware

The construction of GARC has been implemented by integrating the Arduino UNO R3 microcontroller with the L293D motor driving circuit.

The following are some features of the microcontroller used, i.e., Arduino UNO R3

Arduino UNO R3:

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform;

Specifications of Arduino UNO R3

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Table 4.1: Specifications of Arduino UNO R3

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Arduino Uno R3 Front *Arduino Uno R3 Back*

Fig 4.1: Arduino UNO R3

IC L293D motor driving circuit:

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors.

L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction. The motor operations of two motors can be controlled by input logic at pins 2 & 7 and 10 & 15. Input logic 00 or 11 will stop the corresponding motor. Logic 01 and 10 will rotate it in clockwise and anticlockwise directions, respectively.

Enable pins 1 and 9 (corresponding to the two motors) must be high for motors to start operating. When an enable input is high, the associated driver gets enabled. As a result, the outputs become active and work in phase with their inputs. Similarly, when the enable input is low, that driver is disabled, and their outputs are off and in the high-impedance state.

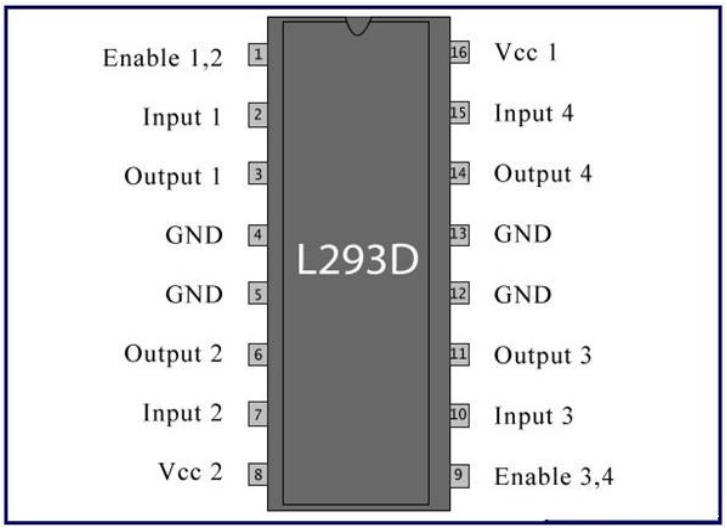
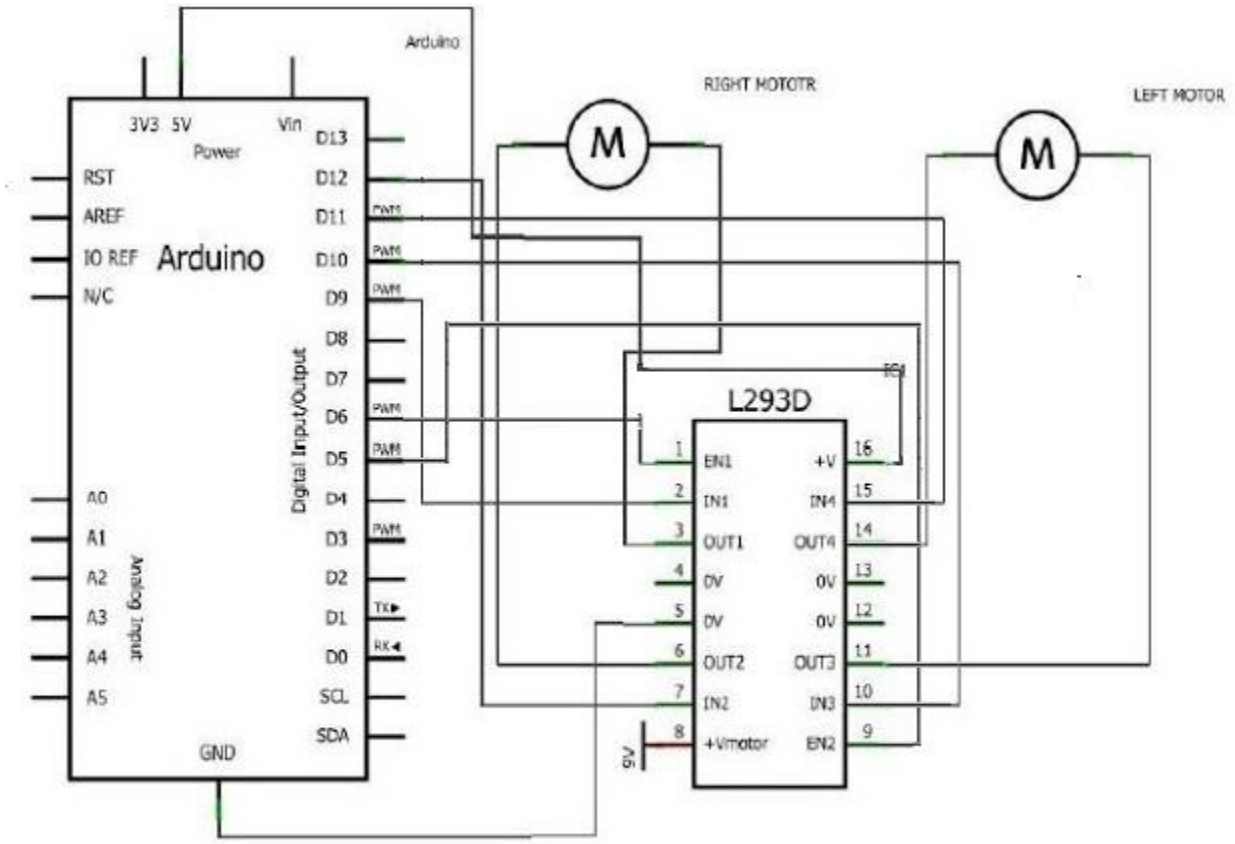


Fig 4.2: Pin Diagram of IC L293D

Pin Description of IC L293D

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	$V_{cc\ 2}$
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	$V_{cc\ 1}$

Table 4.2: Pin Description of IC L293D



Hardware Circuit Setup

Fig 4.3: Integration Circuit of Arduino UNO R3 with IC L293D

The circuit is connected as shown in the above circuit diagram for the functional GARC.

The circuit is implemented on breadboard which is placed on the robotic car to plug in the wires which supply power to the circuit.

The power to the motor driving circuit is provided by a battery supply of 4V and 2.5AH current.

The power to the Arduino UNO R3 microcontroller is supplied by the laptop connected to it by a serial-to-USB cable.

Phase II: Car Automation System Simulation with Gesture Recognition

File Structure

The file structure of the project is portrayed by the following figure:

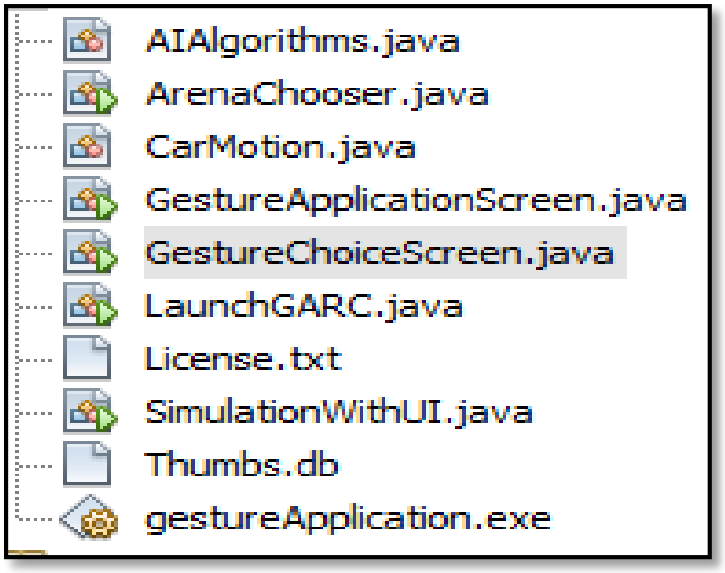


Fig 4.4: File Structure of the Project

File Name	File Type	Purpose
AIAlgorithms	Java	A*, Heuristics and smoothing
ArenaChooser	Java	Selection of virtual arena
CarMotion	Java	Controller and Car Tracking
LaunchGarc	Java	Project Launch Application
License	Text File	BSD 3.0 License File
SimulationWithUI	Java	UI File for AI Simulation
gestureApplication	C++	Image Processing

Table 4.3: Description of source files

CHAPTER 5

Results

Finally we have developed a versatile system that would take detect and track colored components and on the basis of the orientation of the colored objects, it would further send instructions the real or simulated car as to how much to turn, to go straight or to stop etc.

The system is independent of the position of the colored objects, i.e. it does not matter whether green or blue object is on the left or the right hand side, and any object can be present on either side. We have made provisions to stop the car if object is not detected. This was necessary as sometimes the object go outside the scope of the webcam while driving the car.

Further additions to the gesture program can be easily made, and provisions have been made in the code written in order to ensure a high degree of independence from any new module that is written.

After the completion of final phase of the project, the following screenshots of the application were taken:

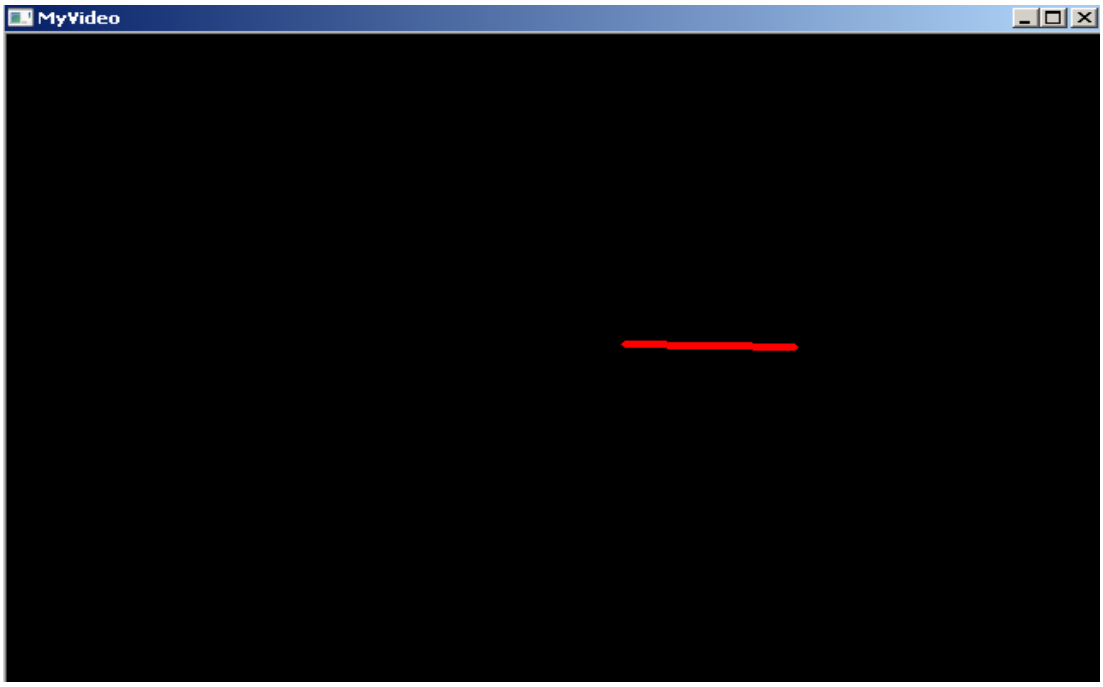


Fig 5.1: Relative orientation of colored objects in gesture application

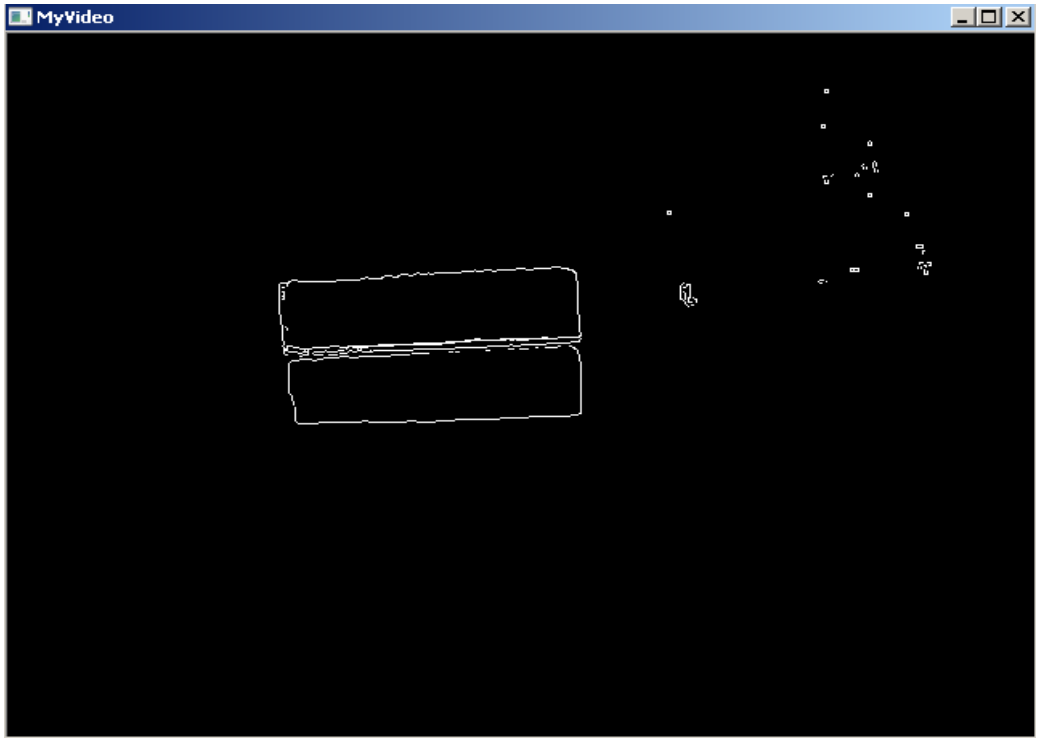


Fig 5.2: Colored object and edge detection

Open Gesture Application that takes gesture inputs on Laptop 1

In the following screen, always enter the port no as 53000.

Enter the IP Address of Laptop 2.

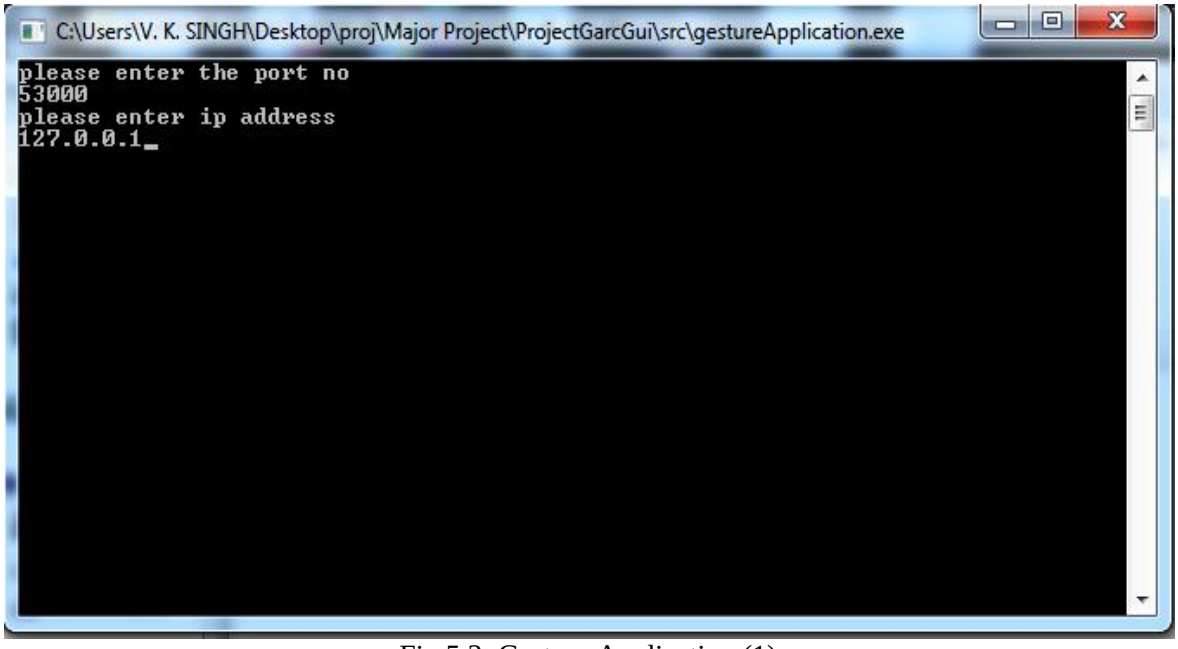


Fig 5.3: Gesture Application (1)

Once the above asked information is entered, next screen is ready to capture gestures.

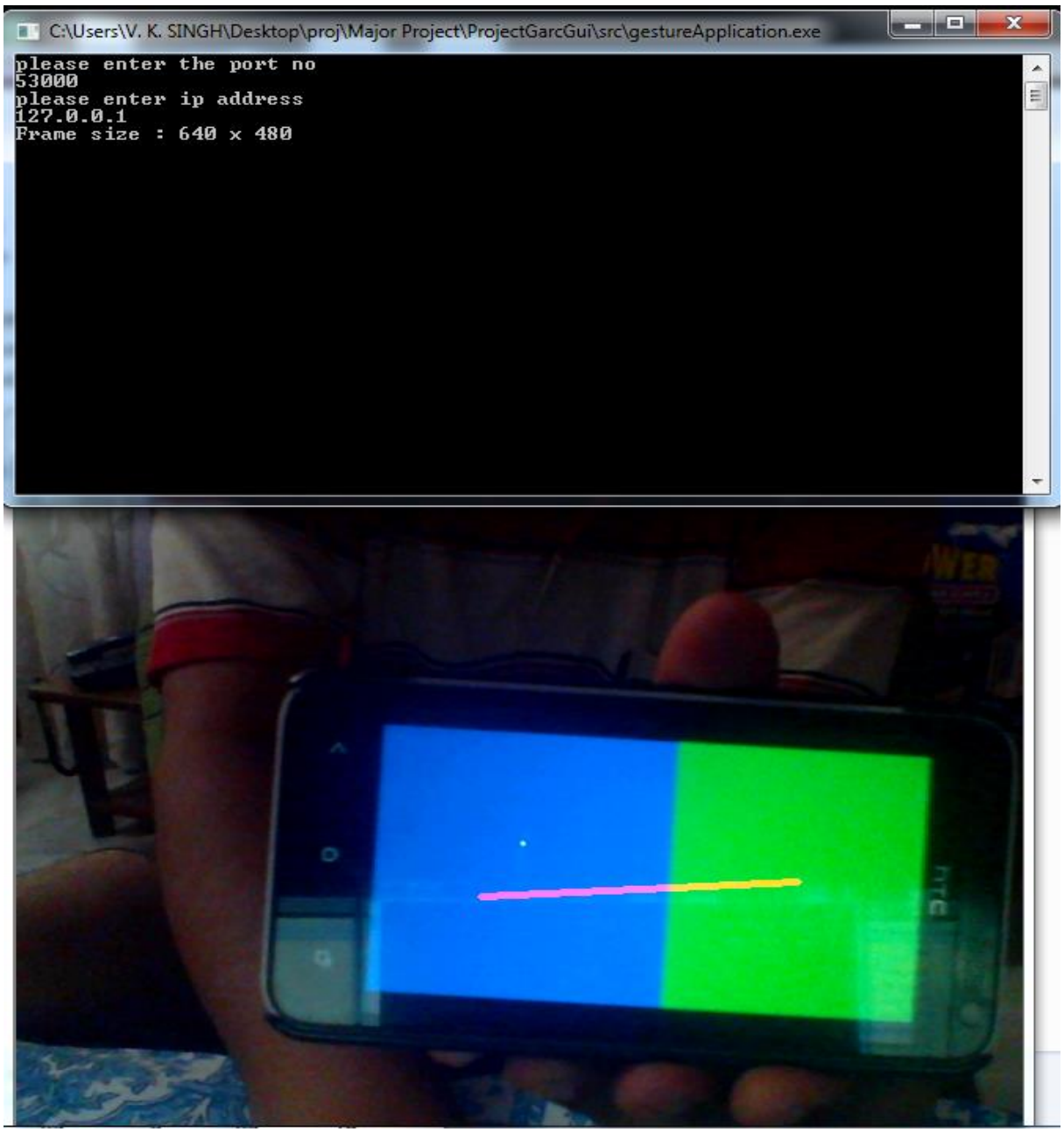


Fig 5.4: Gesture Application (2)

In the above screenshot, the red line created between the blue and green regions is the direct measure of inclination or steer value of our "Virtual Steering". These values can be captured by the Java program SocketReceive.java. A screenshot for the values being captured is shown below.

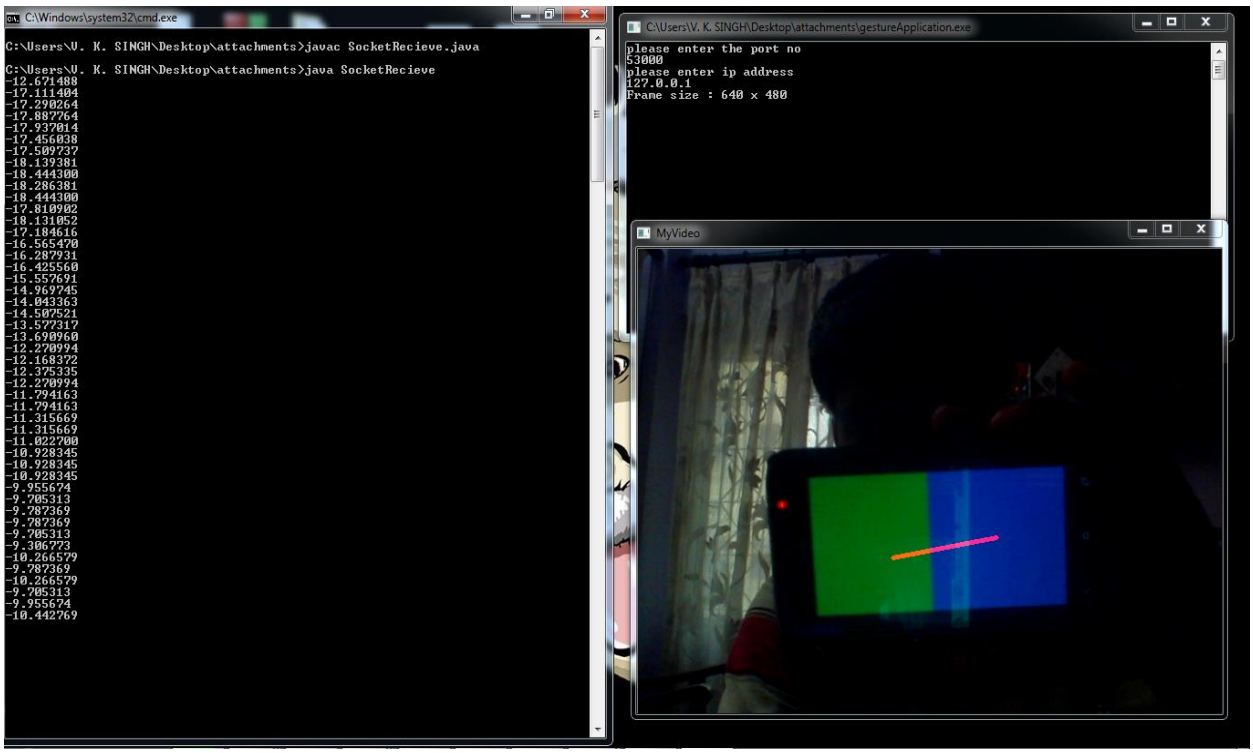


Fig 5.5: Steer values from gesture application (1)

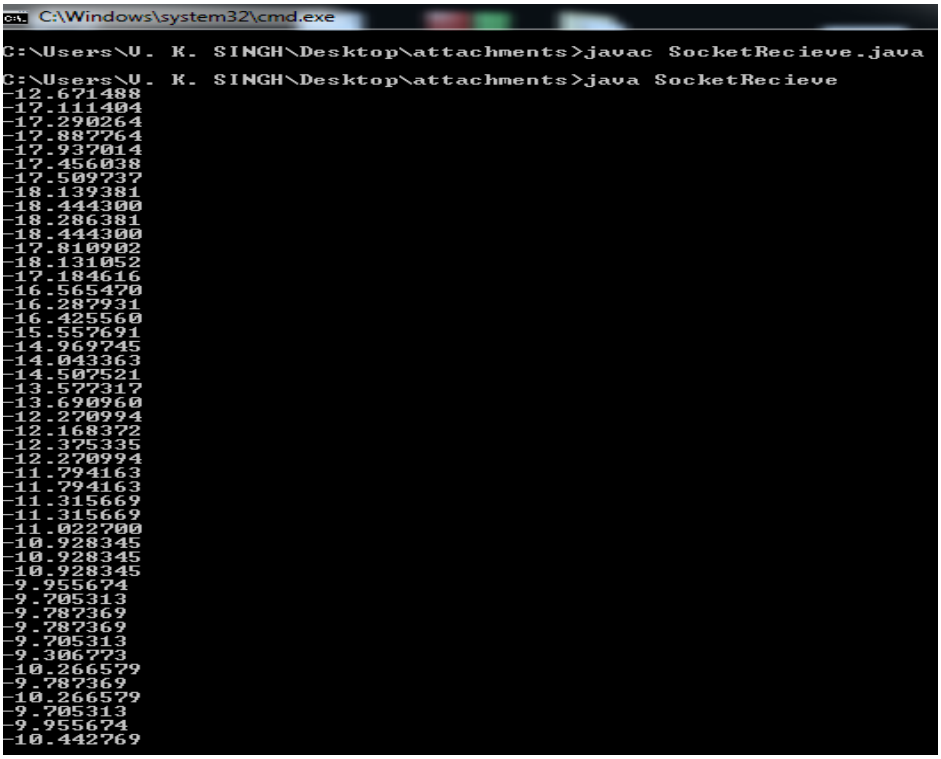


Fig 5.6: Steer values from gesture application (2)

The values being captured like the ones in the above screenshot are automatically written to a file named “testfile.txt” and the data in this file is sent to Laptop 2 over WiFi connection which runs a code in Processing IDE and sends these values to GARC’s Arduino UNO R3 microcontroller via a Serial-to-USB cable which processes these values to output the amount of power to be given to the respective motors in the robotic car.

Screenshots of GARC:

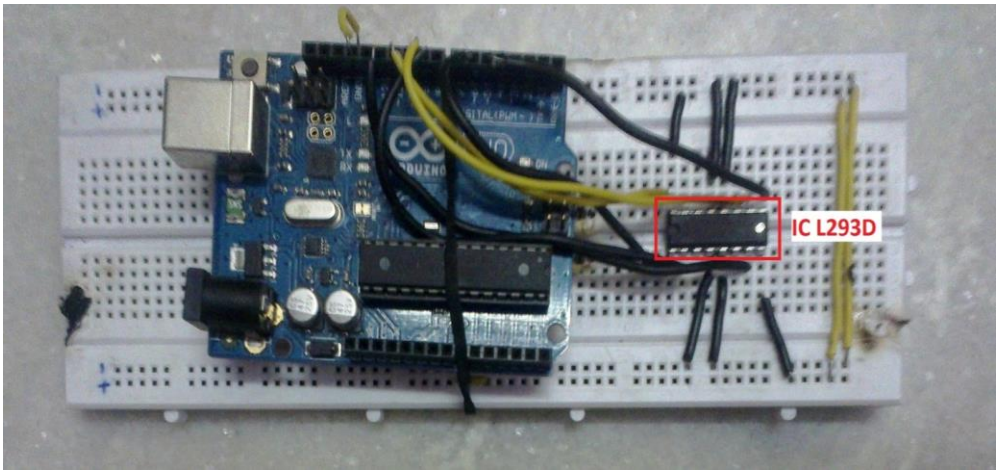


Fig 5.7: Integrated Connections between Arduino UNO R3 and IC L293D

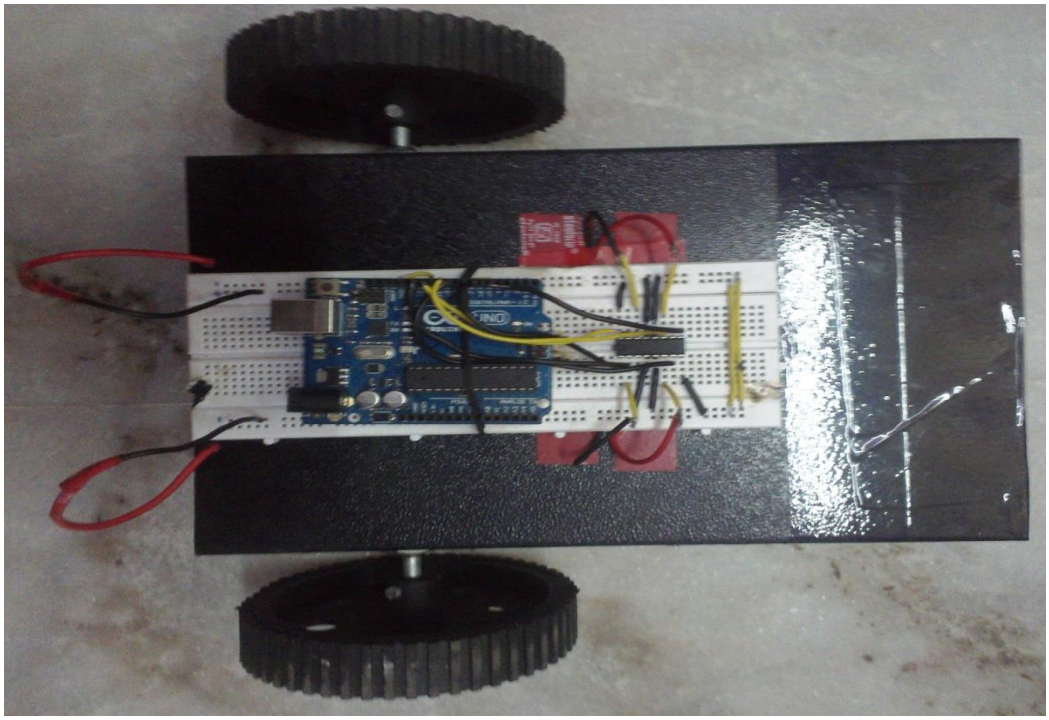


Fig 5.8 Top View of GARC

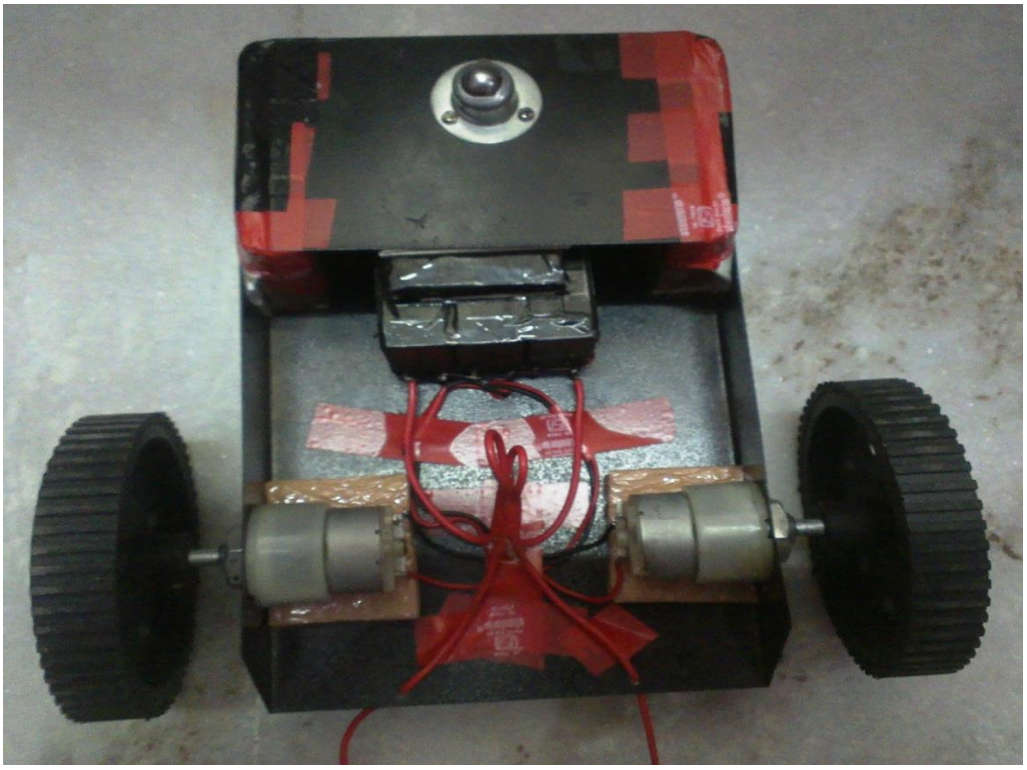


Fig 5.9 Bottom View of GARC

GARC:

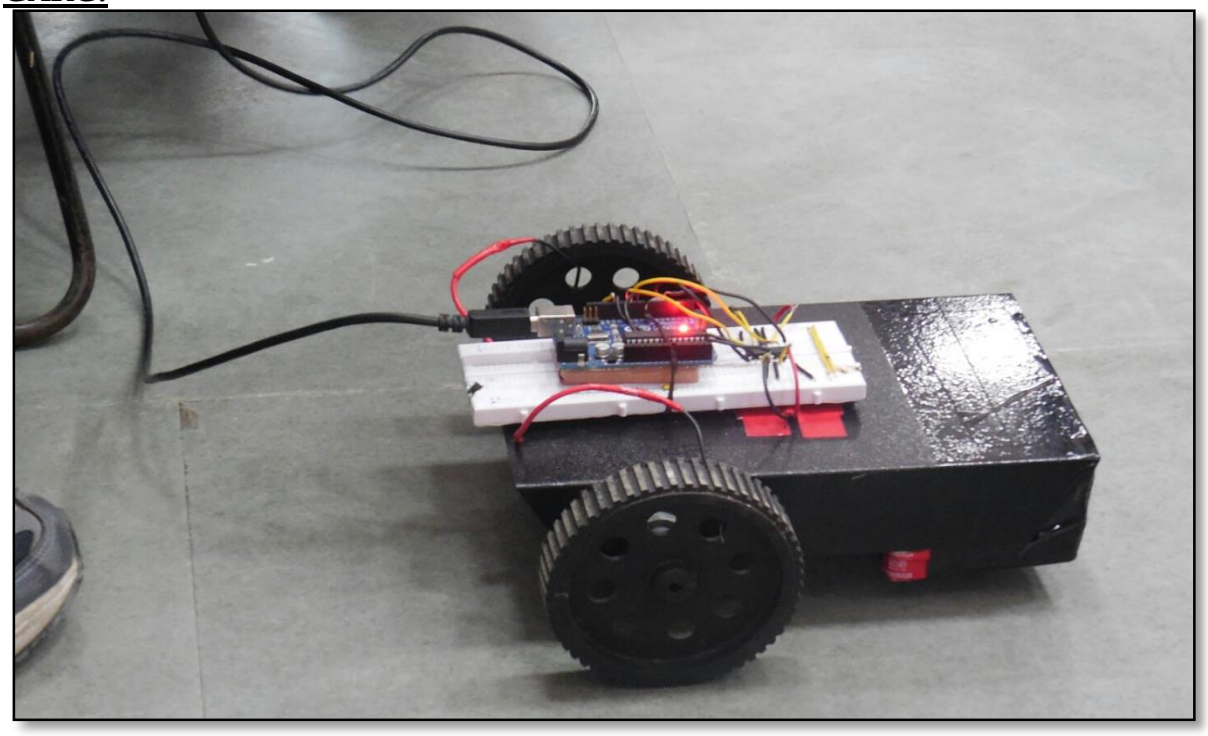


Fig 5.10: GARC

Application Launch Screen:

5.11: Application Launch Screen

Gesture Choice Screen:

Fig 5.12: Gesture Choice Screen

Gesture Application Screen:

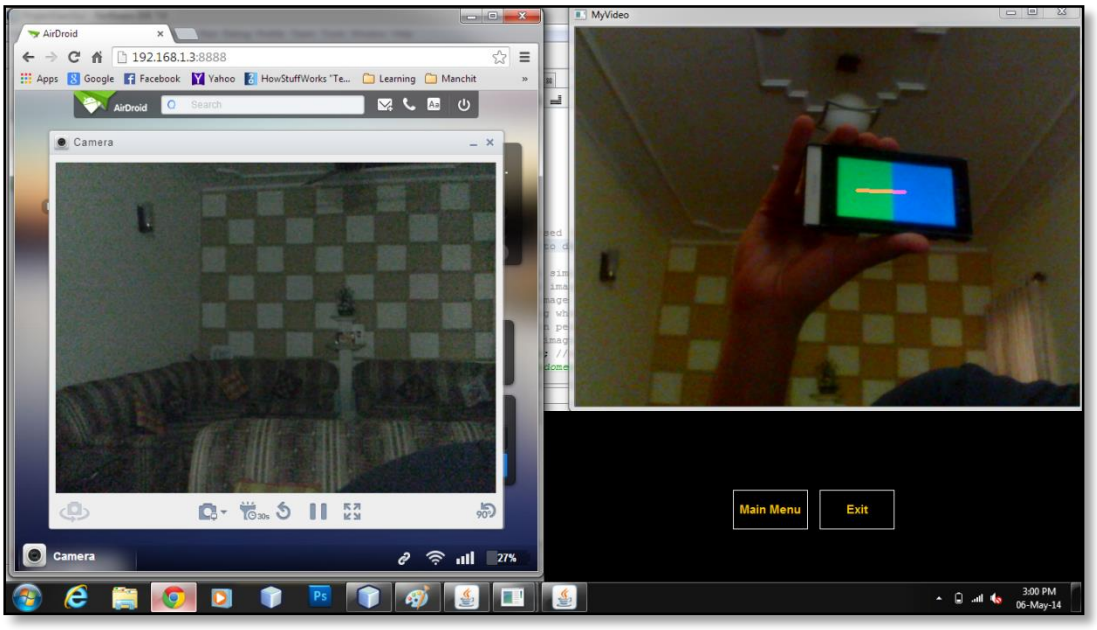


Fig 5.13: Gesture Application Screen

AI Choice Screen:

Fig 5.14: AI Choice Screen

Single Car AI Simulation:

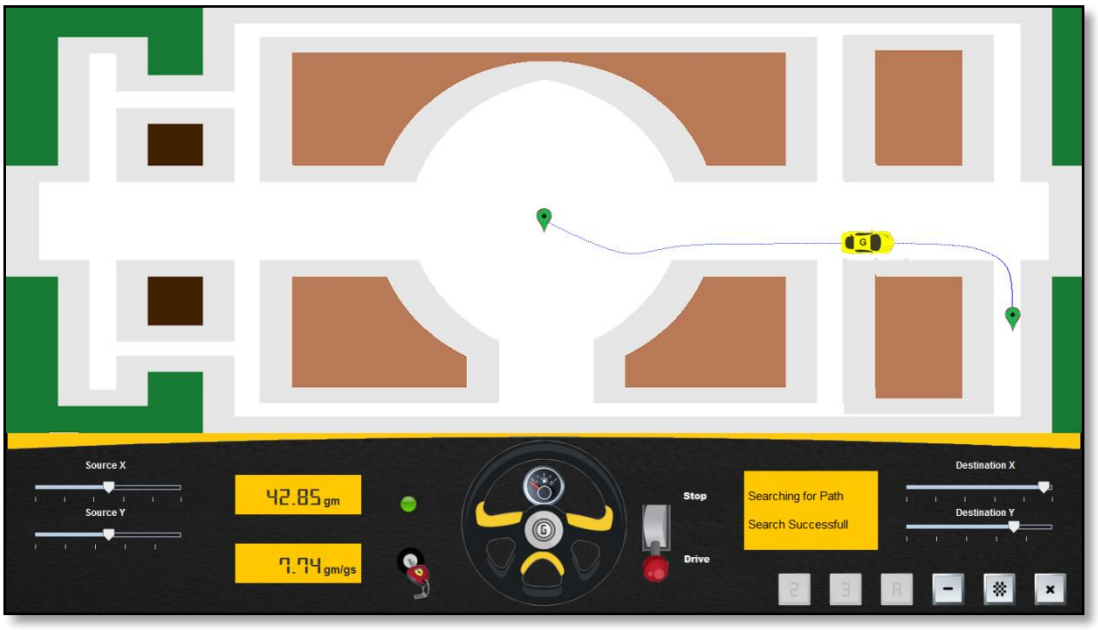


Fig 5.15: Single Car AI Simulation

Multiple Car AI Simulation:

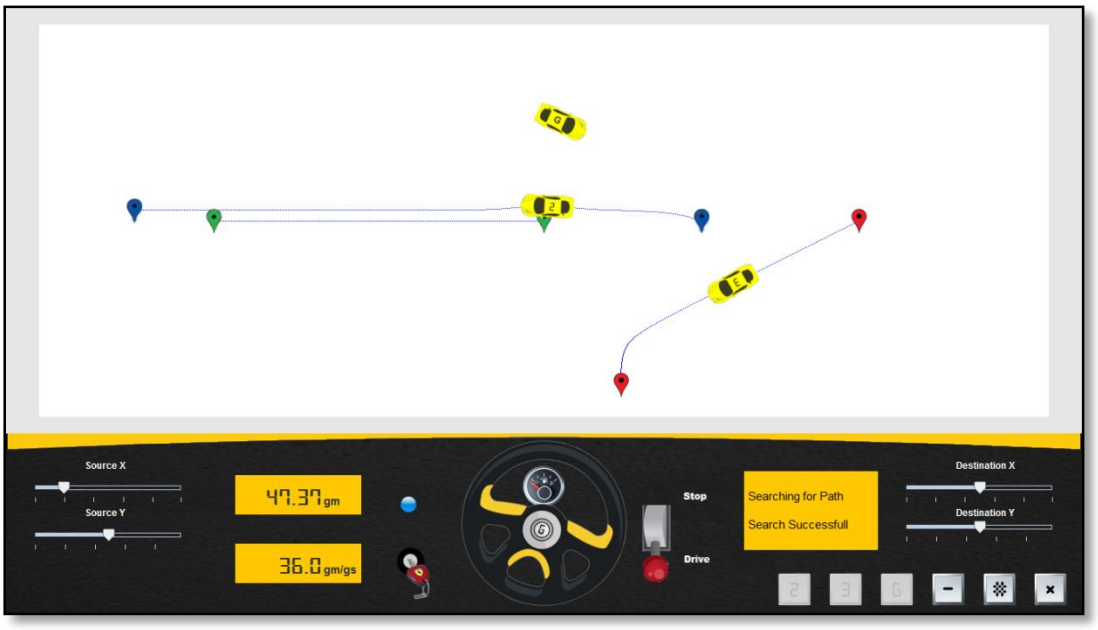


Fig 5.16: Multiple Car AI Simulation

The following figure shows the actual run of Project GARC:

Fig 5.17: Actual Project Implementation

CHAPTER 6

Testing

Manual Testing:-

Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user, and use most of all features of the application to ensure correct behavior. To ensure completeness of testing, the tester often follows a written that leads them through a set of important . It is the oldest and most rigorous type of software testing. Manual testing requires a tester to perform manual test operations on the test software without the help of Test automation. Manual testing is a laborious activity that requires the tester to possess a certain set of qualities; to be patient, observant, speculative, creative, innovative, open-minded, resourceful, unopinionated, and skillful.

Unit Testing:-

This initial stage in testing normally carried out by the developer who wrote the code and sometimes by a peer using the white box testing technique. The most 'micro' scale of testing; to test particular functions or code modules. Typically it is done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code; may require developing test driver modules or test harnesses.

Here in our project we have tested each module uniquely by using unit testing. We have provided facility to recover project when error occurs.

Integration testing:-

It is a testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems. This stage is carried out in two modes. As a complete package or as an increment to the earlier

package. Most of the time black box testing technique is used. However, sometimes a combination of Black and White box testing is also used in this stage.

The testing paradigm followed for the given project included the following the testing routines:

- **Unit Testing:** A* Search, Smoothing, Controller, Car Tracking, UI Screens, Gesture Program and more.
- **Integration Testing:** Complete GUI with Hardware.
- **Function testing:** Movement of Robotic car in multiple environments.

The results obtained from these tests were fully satisfactory to the purpose of the project and have been displayed in the RESULTS section.

6.1 Test Cases:

[Please note in each case there are 50 trials were there.]

Table 6.1: Detection of blue and green objects (individually)

Project:	Gesture Recognition				
Module:	Detection of Blue and Green Objects				
Form REF:	Main				
Test Case No:	1				
Functional Specification	Colored Object Detection				
Test Date:	28/2/2014				
Test Objective:	To detect whether blue or green object detected.				
Test Data:	Blue and green color cap.				
Precondition	Camera should be on and program running.				
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**Accuracy(%)**
1	No blue or green object present in frame.	Camera input	Show message as object not detected.	YES, message is displayed	81
2	Use blue object.	Camera input	Show message as object detected.	YES, message is displayed	91
3	Use green object.	Camera input	Show message as object detected.	YES, message is displayed.	92

Table 6.2: Detection of red and green objects (together)

Project:	Gesture Recognition				
Module:	Detection of Blue and Green Objects				
Form REF:	Main				
Test Case No:	2				
Functional Specification	Colored Object Detection				
Test Date:	12/03/2014				
Test Objective:	To detect whether blue and green object detected.				
Test Data:	Blue and green color cap.				
Precondition	Camera should be on and program running.				
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**Accuracy(%)**
	Both Blue and Green object present in frame.	Camera input	Show message as object detected and image displaying centroid of both.	YES, message is displayed and image generated is correct.	87

Table 6.3: Object Presence in Scope

(To detect whether an object has moved in the scope or an already present in scope has moved out)

Project:	Fingertip Gesture Recognition				
Module:	Object Presence In Scope				
Form REF:	Main				
Test Case No:	4				
Functional Specification	Object Presence In scope				
Test Date:	27/03/2014				
Test Objective:	To detect if an object has moved into the scope or moved out of the scope.				
Test Data:	Blue and/or green color cap.				
Precondition	Camera should be on and program running.				
Step No	**Steps**	**Data**	**Expected Results**	**Actual Results**	**Accuracy(%)**
1	No object present in scope.	Camera input	Show message as object or motion not detected.	YES, message is displayed	100
2	Move blue/green object in scope.	Camera input	Show message object moved in.	YES, message is displayed	96
3	Remove object from scope.	Camera input	Show message as object moved out.	YES, message is displayed.	98

6.2 Likely Sources of Error and Inaccuracies

The chances of object not being detected increase exponentially in both low lighting and very bright lighting conditions. It is essential that frames be captured under optimal lighting.

Moreover, the color caps used should be bright enough. Due to illumination constraints and the fact that multiple object detection is enabled, therefore, it is possible that if in the same frame more than one green or red objects are present then they might be detected.

Another important point to note is that if the image is too large then it takes some time to load, hence care must be taken to select images that are not very large in size.

CHAPTER 7

Conclusions

We have achieved the target of controlling a robotic car by gestures which are sent by the user with the help of color detection of blue and green objects. The driving instructions too, are sent from the user's machine to the machine connected with the car over a WiFi network.

On the Autonomous Travel System front, we have achieved the target of driving a virtual car in a virtual map by implementing artificially intelligent algorithms and also by acquiring gestures which are provided by the user. Also, feature of collision detection between various virtual cars has been implemented in the presented simulation.

The areas with scope for improvement and the limitations with the current Project GARC are enumerated in the following sections.

7.1 Limitations

The limitations currently seen in project GARC are inherently due to lack of available time and finances to carry out further work.

Currently blue and green objects are being used currently, using thermal imaging and depth sensors, skin detection can be performed with reasonable accuracy thus eliminating the need of using colored components.

Collision handling is not incorporated in the simulation part, this can be further done. A requirement to multiple car model enhancements is the use of a machine with greater processing power.

The automated driving concept has not been incorporated into the car due to the time limitations. This can be done by placing additional sensors on the car and delegating the navigation processes to a computer. Furthermore, all this can be handled within the car by using a better microcontroller which is capable of performing at better speeds.

7.2 Future Scope

As mentioned in the introduction section, the scope of this project extends across various areas of work such as military applications, assistive technology for people with special abilities etc.

In order to realize the above goals, further additions may be required to the project. Essentially it requires the placement of a depth sensor based camera on the car which possibly provides a 360 degree view, this would help to navigate the car in a more precise fashion. The communication limitations that may exist over long distances can be overcome depending on the distance that may be expected between the car and the controller, for very large distances though the 4G technology is appropriate but for even higher speeds satellite communication may be required.

The automated driving mechanism can be incorporated into the present car model, which has not been done due to temporal and financial limitations. However in the immediate future, we have made provisions for self navigation of car in which case the car would need to delegate navigation processes to a computer.

CHAPTER 8

References and Bibliography

Book References

[1a]HAND GESTURE RECOGNITION USING KINECT By Yi Li B.S .. Communication University of China, 2010

[1b] TACTILE GESTURE RECOGNITION FOR PEOPLE WITH DISABILITIES
Yu Yuan, Ying Liu, Kenneth Barner

[1d] Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distance with a Commodity Depth Camera*;*Zhou Ren Junsong Yuan, Zhengyou Zhang,Mircosoft Research 2012,NTU Singapore

[1c]A Novel System for Hand Gesture Recognition; Matthew S. Vitelli, Dominic R. Becker, Thinsit (Laza) Upatising

[1e] Finger Tips Detection and Gesture Recognition Ankit Gupta,IIT-K research,Kumar Ashis Pati

[1f] http://ai.stanford.edu/~mitul/cs223b/seg.html

Real-time Hand Tracking and Finger Tracking for Interaction
CSC2503F Project Report
By: Shahzad Malik (smalik@cs.toronto.edu)

[2] Finger Tips Detection and Gesture Recognition
Course Instructor: Prof. Simant Dubey, CS676
Indian Institute of Technology, Kanpur
November 12, 2009

[3] International Journal of Engineering Research & Technology (IJERT)
Vol. 1 Issue 8, October – 2012

[4] Harish Kumar Kaura, Vipul Honrao, Sayali Patil, Pravish Shetty – Gesture Controlled Robot using Image Processing. (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 5, 2013

[5] A Novel System for Hand Gesture Recognition
By: Matthew S. Vitelli, Dominic R. Becker, Thinsit (Laza) Upatising
(mvitelli,drbecker,lazau)@stanford.edu

[6] Hand Gesture Recognition Using Different Algorithms Based on Artificial Neural Network

By :
1. Shweta. K. Yewale, Prof. Ram Meghe Institute of Technology & Research, Badnera (shwetayewale@rediffmail.com)
2. Pankaj. K. Bharne, Sipna College of Engg. & Technology,Amravati (pankajbharne@gmail.com)

Web References

[7] "Udacity Course- Programming Robotic Car", https://www.udacity.com/course/viewer#!/c-cs373/l-48739381/m-48735024

[8] "Connecting Arduino to Processing.", https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/introduction

[9] "Arduino and L293D Connections", http://www.instructables.com/id/Arduino-and-L293D-Robot-Part-1-/?ALLSTEPS

[10] "Guide to Create a Gesture Controlled Robot", http://gesturecontrolledrobot.blogspot.in/2012/06/chapter-1.html

[11] "Arduino Uno Board ", http://arduino.cc/en/Main/ArduinoBoardUno

APPENDIX-I

List of figures

APPENDIX-II

List of tables

APPENDIX-III

LICENSE