

**Kathmandu University**  
**Department of Computer Science and Engineering**  
**Dhulikhel, Kavre**



**A Mini Project Report on**  
**“AUTOCar”**

**COMP 306**

**(For partial fulfillment of 3<sup>rd</sup> Year/ 2<sup>nd</sup> Semester in Computer Engineering)**

**Submitted by**  
**Manasi Kattel (18)**  
**Araju Nepal (28)**  
**Ayush Kumar Shah(44)**  
**Deepesh Shrestha (50)**

**Submitted to**  
**Satyendra Nath Lohani**  
**Department of Computer Science and Engineering**

**Submission Date:24<sup>th</sup> July,2018**

## **Acknowledgement**

First of all, we would like to express our heartfelt gratitude to our course instructor Mr. Satyendra Nath Lohani for his support during the mini project and co-operating with us to help us carry our project smoothly. His experiences in the field have been a great asset for our project. His encouraging words and working techniques have made this project a successful one.

Our deepest appreciation to all those who provided us the possibility to complete this project. We extend our gratitude to all who have deliberately or unknowingly added a brick in the completion of this project. We enjoyed the duration of the work studying different modules and creating this project.

Taking this opportunity, we would like to thank all our peers and classmates who directly or indirectly helped us in making this project a successful one be it by encouraging us throughout the project or else through their valuable suggestions which we have tried our best to assimilate within our work.

## **Abstract**

Our project entitled “ **AUTOCar** ” is an autonomous vehicle that performs image processing on digital images obtained from live webcam in real time to detect lane, obstacle and traffic signs in front of it. The purpose of our project is to build a self driving car that can drive safely on roads. We are using “Raspberry Pi” to process the continuous image derived from “Minoru 3D Webcam”. To process the images we have implemented various mathematical algorithms. After processing the images, the result is implemented using “GPIO pins” to drive the vehicle. We expect this car to avoid accidents using appropriate image processing algorithms. Hence, we have been able to make an obstacle avoiding car which can drive responding to stops signs and traffic lights.

**Keywords:** *autonomous vehicle, self driving, image processing, electronics, embedded systems,*

## List of Figure

It gives information about all the figures used in the documentation. All figures used in the report should be listed, used during explanation of the works.

Fig No.	Figure	Page No.
2.2.1	Circuit Schematics	5
2.2.2	Non-maximum supression	10
2.2.3	Hysteris thresholding	10

## **Acronyms/Abbreviations (if any)**

The list all abbreviations used in the documentation is included in this section.

GPS	Global Positioning System
LIDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
GIS	Geographic Information System
RPROP	Resilient backpropagation
VNC	Virtual Network Computing
IDE	Integrated Development Environment
OpenCV	Open Source Computer Vision

## TABLE OF CONTENTS

TITLE			PAGE NO.
<b>Acknowledgement</b>			<b>i</b>
<b>Abstract</b>			<b>ii</b>
<b>List of figures</b>			<b>iii</b>
<b>List of tables</b>			<b>iii</b>
<b>Abbreviation</b>			<b>iv</b>
<b>Chapter 1</b>	<b>Introduction</b>		<b>1</b>
	1.1	Introduction to Embedded System	<b>1</b>
	1.2	Background	<b>2</b>
	1.3	Objectives	<b>2</b>
	1.4	Motivation and Significance	<b>2</b>
	1.4.1	Motivation	<b>2</b>
	1.4.2	Significance	<b>3</b>
<b>Chapter 2</b>	<b>Embedded System Development</b>		<b>4</b>
	2.1	Requirement Engineering	<b>4</b>
	2.2	Hardware-Software Partitioning, Hardware Design, Software Design	<b>4</b>
	2.2.1	Hardware Design	<b>4</b>
	2.2.2	Software Design	<b>7</b>
	2.3	Hardware and Software Implementation	<b>11</b>
	2.3.1	Hardware Implementation	<b>11</b>
	2.3.2	Software Implementation	<b>11</b>
	2.4	Hardware-Software Integration and Testing	<b>12</b>
<b>Chapter 3</b>	<b>Discussion on the Achievements</b>		<b>13</b>
<b>Chapter 4</b>	<b>Cost Estimation</b>		<b>14</b>
<b>Chapter 5</b>	<b>Result</b>		<b>15</b>

<b>Chapter 6</b>		<b>Conclusion and Recommendation</b>	<b>16</b>
	6.1	Limitation	<b>14</b>
	6.2	Future Enhancement	<b>14</b>
<b>Reference</b>			<b>17</b>
<b>Appendix A : Car Body and Parts used</b>			<b>20</b>

# Chapter 1: Introduction

## 1.1. Introduction to Embedded System

An embedded system is an electronic system that has a software and is embedded in computer hardware. It is programmable or non- programmable depending on the application. As with any electronic system, this system requires a hardware platform and that is built with a microprocessor or microcontroller.

Embedded systems are classified into four categories based on their performance and functional requirements:

- Stand alone embedded systems
- Real time embedded systems
- Networked embedded systems
- Mobile embedded systems

Embedded Systems are classified into three types based on the performance of the microcontroller such as

- Small scale embedded systems
- Medium scale embedded systems
- Sophisticated embedded systems

Our project falls under stand alone and real time embedded system according to the performance and functional requirements and small scale embedded system according to the performance of the microcontroller.

### **Stand alone embedded system-**

It is a system that works by itself, and does not need a host system like a computer. It takes digital or analogs as inputs, and do some processing to them and outputs the data as it is attached to the output service.

### **Real time embedded system-**

It gives the output in a specific time, so it basically follows deadlines to complete the tasks. The following two systems are the types of Real Time system.



## **Small scale embedded system-**

These usually use 8 bit or 16 bit microcontrollers, and have minimum hardware and software. They are so small and require little power they may be powered by a battery.

## **1.2. Background**

Most of the people are aware of driving safely. Despite of that, accidents happen quite often and most of them are the results of reckless distracted driving. Autonomous car, controlled by computer is one of the solution to this problem as there are no situations for a computer to be distracted.

Self-driving car is one topic that has got a lot of attention in the media lately. Technology has enhanced various sectors of our life in many ways. Things are becoming automatic every day. The vehicles are being autonomous. Obstacle detection autonomous vehicles detect obstacles that come on its way and analyze the obstacles to make proper decision for its movement. The autonomous vehicle analyzes the environment using the sensors attached to it. Based upon environmental situation, the autonomous vehicle can move or drive on its own as manually driven by human. Some of the popular technologies used in autonomous vehicles are camera, ultrasonic sensor, RADAR, LIDAR and GPS.

We plan to design an autonomous car which is a self-driving car that avoids obstacles and detects the path through image processing and input output programming. It is based on hardware and software.

## **1.3. Objectives**

1. To design an autonomous car on the track
2. To detect the objects that come on the path of car and avoid them
3. To detect lanes and drive accordingly
4. To avoid the head collision

## **1.4. Motivation and Significance**

### **1.4.1. Motivation**

In today's date almost, everyone has their own vehicle. The rapid increase in population causes increase in number of vehicles while the road size remains the same. People these days are always in a hurry and easily distracted by their cellphones or people around them. Due to these serious reasons accidents occur frequently. To avoid accidents and save lives we wanted to contribute something that might be of some help to the society. We were also motivated by the

concept of Tesla car which was trending in technology field recently. This autonomous car doesn't need a driver, so they are less prone to accidents as computers don't make mistakes as humans do.

#### **1.4.2. Significance**

This car is used for decreasing the number of accidents, making lives easier and faster and showcasing where technology has reached to. This system can be extended to remote management and GIS based system.

While developing the system, there shall be space for further modification. There shall be a proper documentation so that further enhancement becomes easy. As a whole the car is focused as a useful system that will come in handy to make lives more digital and automatic.

## **Chapter 2: Embedded System Development**

### **2.1. Requirements Engineering**

The requirements of the system to be developed was realised as:

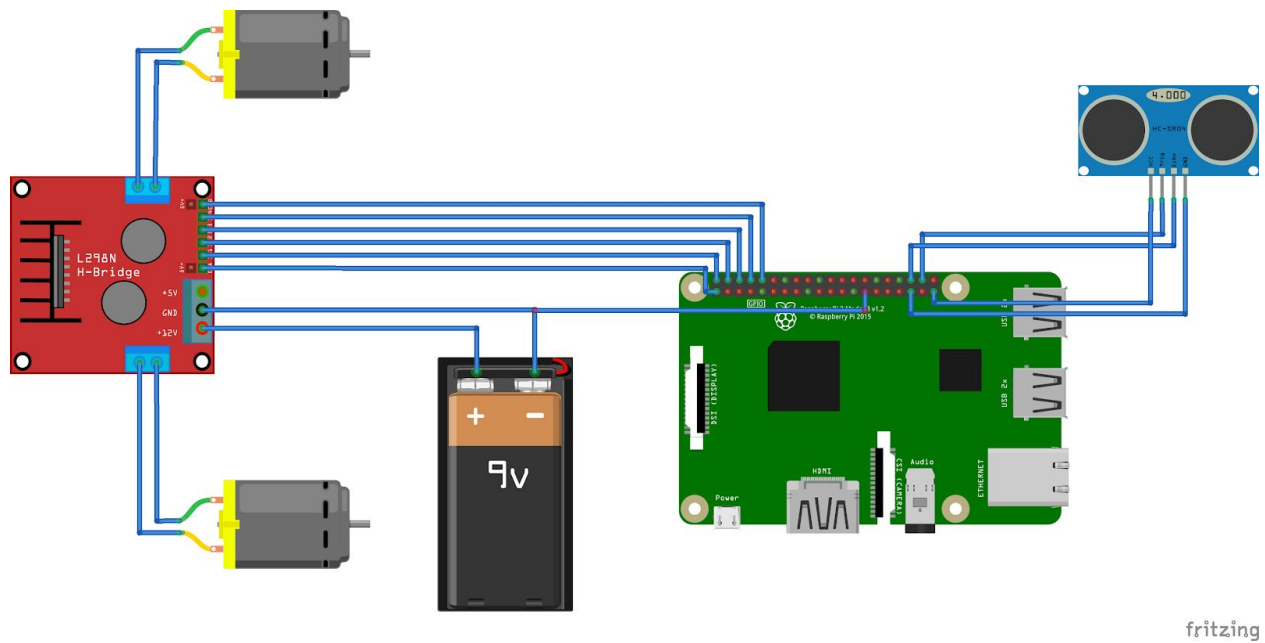
- a. Inputs - the live video was input through a camera while driving on a lane .
- b. Output- lane detection and obstacle avoidance
- c. Processing - detection of lane,obstacles and stop sign need to be done in real time in order to avoid accidents
- d. Input supply volt - 12 volt for motor and 5 volt for raspberry pi
- e. Communication interface - IEEE 802.11G
- f. Size - small scale embedded system
- g. Operating environment -
- h. Standards - universal stop size i.e. octagonal shape, red background with white text and white border.

### **2.2. Hardware-Software Partitioning, Hardware Design, Software Design**

#### **2.2.1 Hardware design**

##### **2.2.1.1 Hardware Specification**

- Raspberry Pi 3 Model B+
- Minoru Stereo Camera
- Motor Shield
- Motor and Wheels
- Ultrasonic Sensor
- Power Source



• Fig 2.2.1: Circuit Schematics

### 2.2.1.2 Hardware Description

#### 1. Raspberry Pi

The Raspberry Pi is open hardware, except for the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip). The Raspberry Pi was designed for the Linux operating system. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does. With its suitable internal specifications, we were able to perform image processing on it with low frame per sec (FPS) output.

#### 2. Minoru

The Minoru 3D Webcam is a stereoscopic webcam. The webcam, which hooks to a computer via USB port, consists of two cameras, being "roughly the same distance apart as human eyes," that are held together in a device that resembles Wall-E, a Disney character, in appearance. It is capable of 3D, 2D, and Picture-in-picture graphics. Its image can be output from 320x240 pixels to 800x600 pixels and is capable of 30 frames per second. The 3D imagery can be produced in anaglyph and side by side format. The camera has two VGA 640x480 CMOS sensors, two high-quality wide-angle lenses, and a built in USB microphone. It is most effective when the user is at a minimum of three feet away. It derives its name from the Japanese term Minoru, which means "reality."

### **3. Motor Shield**

L298N H-bridge Dual Motor Controller Module 2A with Raspberry Pi. This allows you to control the speed and direction of two DC motors, or control one bipolar stepper motor with ease. The L298N H-bridge module can be used with motors that have a voltage of between 5 and 35V DC.

### **4. Motor and Wheels**

We used two 6V 250 Rpm Plastic Gearmotor, wheel and a free moving wheel at the front to construct the moving part for the car.

### **5. Ultrasonic Sensor**

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object. It can be used to solve even the most complex tasks involving object detection or level measurement with millimeter precision, because their measuring method works reliably under almost all conditions.

### **6. Power Source**

For providing power to our hardware, we used two different power sources. We used a 5V 1mAh power bank to power the Raspberry Pi. We used 3 pieces of 4.2 V sealed acid battery in series to obtain a total of 12.6 V to power the Motor and the motor shield.

### **7. Car Chassis and Body Parts**

Two car chassis plate was used along with fasteners and zip ties to assemble the body for the car.

## 2.2.2 Software design

### Software Specification

<b>Programming Language</b>	Python
<b>Operating System</b>	Debian based OS (Raspbian and Ubuntu)
<b>Image Processing Library</b>	OpenCV
<b>Desktop Sharing System</b>	VNC
<b>Remote Connection Tool</b>	OpenSSH

### Software Description

#### 1 Python

Python is a modern, easy-to-learn, object-oriented programming language. It has a powerful set of built-in data types and easy-to-use control constructs. We implemented various algorithm in Python for Image Processing to use with OpenCV.

#### Libraries used

<b>matplotlib</b>	matplotlib is a plotting library for Python with Numpy as a numerical mathematical library.  Numpy. This helps to render the numpy array into a plot that can be manipulated.
<b>cv2</b>	cv2 is the python interface library from OpenCV. This has many functions such as imread(), imshow(), etc that helps in image manipulations.
<b>pyserial</b>	This library encapsulates the access for the serial port. It provides backends for Python. The module named “serial” automatically selects the appropriate backend.

**math**            math is a mathematical library for Python that provides access to the mathematical functions defined by the C standard.

**numpy**            numpy

## **2. Debian based OS**

Debian is a free (as in freedom) GNU/Linux operating system created by volunteers. We used two different operating systems based on Debian, Ubuntu and Raspbian OS.

**Ubuntu** was used on our development machine to code the algorithm.

**Raspbian OS** was used on the Raspberry Pi for Image Processing.

## **3. OpenCV**

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. It runs on Windows, Linux, macOS, FreeBSD, NetBSD, OpenBSD (desktop operating system) and Android, iOS, Maemo, BlackBerry 10(mobile operating system).

## **4. VNC**

Since Raspberry Pi was on the go and it did not have a display, VNC was used to remotely access the GUI interface on the Raspberry Pi to view the output generated by the program.

## **5. OpenSSH**

OpenSSH is a remote connection networking protocol. It was used to access Raspberry Pi remotely to configure it.

## **Lane Detection Algorithm**

Lane detection involves a video input and processing it to detect the lane within which the vehicle is moving.

**Step 1:** Cropping to a Region of Interest that fully contains the lane lines

**Step 2:** Detecting shape edges in the remaining (cropped) image data using Canny edge detection algorithm.

**Step 3:** The set of pixels representing edges are linked together to generate a list of lines using Hough Transform.

**Step 4:** The detected lines are rendered back onto the video.

### **Canny edge detection algorithm**

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986.

#### **1. Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

#### **2. Finding Intensity Gradient of the Image**

Smoothed image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (  $G_x$ ) and vertical direction (  $G_y$ ). From these two images, we can find edge gradient and direction for each pixel as follows:

$$\text{Edge\_Gradient}(G) = \sqrt{G_x^2 + G_y^2} \text{-----} \text{Angle}(\theta) = \tan^{-1}(G_y/G_x)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

#### **3. Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient.



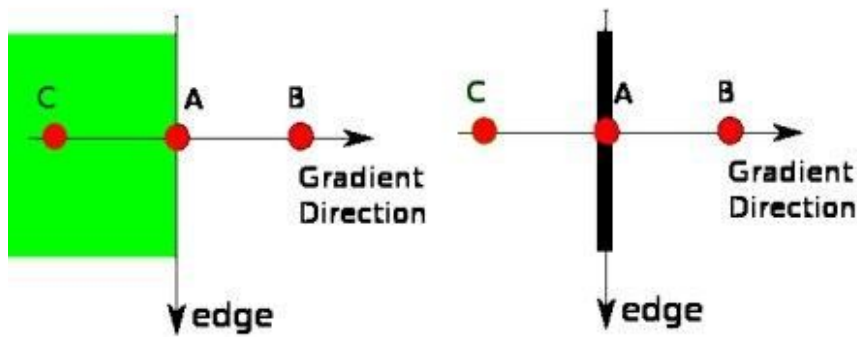


Fig 2.2.2: Non-maximum suppression

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So, point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with "thin edges".

#### 4. Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values,  $\text{minVal}$  and  $\text{maxVal}$ . Any edges with intensity gradient more than  $\text{maxVal}$  are sure to be edges and those below  $\text{minVal}$  are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded.

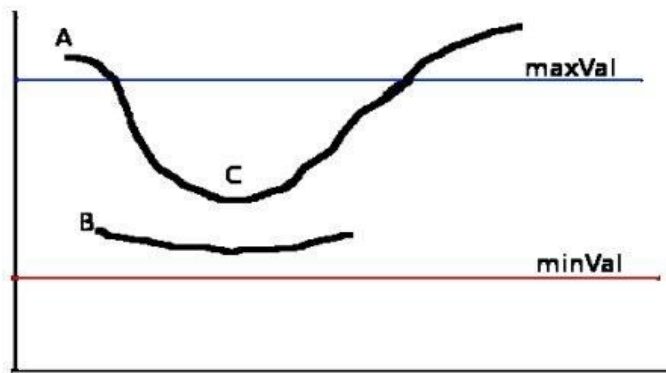


Fig 2.2.3: Hysteresis thresholding

#### Image

The edge A is above the  $\text{maxVal}$ , so considered as "sure-edge". Although edge C is below

maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So, what we finally get is strong edges in the image.

## **2.3 Hardware and Software Implementation**

### **2.3.1 Hardware Implementation**

The designed circuit was converted into PCB was the components was assembled. Then the individual components was tested for defects.

### **2.3.2 Software Implementation**

The algorithm was written into code using Python language. The code was written on the host system, tested and then it was tested on the target system.

#### **1. Converting design into code**

The design was converted into code and has been uploaded in github.

<https://github.com/ayushkumarshah/autocar>

#### **2. Dividing software into different modules**

Different modules such as control.py for controlling the motor of the car, ultrasonic.py for calculating distance from obstacle and final2.py for detecting the lanes and stop signs were created.

#### **3. Optimization of code**

The code was optimized by:

##### **1. Elimination of dead code**

The dead codes were checked and removed in all the modules

## **2. Remove unnecessary debugging code**

While debugging, various debugging codes were used. These unnecessary debugging codes were removed at the last

## **3. Avoid recursion**

Recursion was avoided in the overall code.

## **4. Transferring the code from the host system to the target system**

The codes were initially written in our laptops i.e. the host system and finally transferred to the target system i.e. raspberry pi.

## **2.4 Hardware-Software Integration and Testing**

### **Testing-**

The types of testing implemented in our project is-

#### **White box testing-**

Here, the implementation details are considered while testing the system, you give the test inputs in such a way that each and every line of code is tested.

#### **Acceptance testing-**

After the completion of the project it was presented in ITMeet to the audience and judges evaluated the system if it was working as per the requirements.

#### **Regression testing-**

After testing the system and finding bugs, we rectified the software to remove the bugs. To ensure that modification in one portion of code had no effect on other portions of the code we carried out testing once again.

## **Chapter 3: Discussion on the Achievements**

## **Challenges faced:**

The major challenge we faced during the project is limited resources and insufficient CPU and GPU to train our car. As a result, we couldn't build a neural network and implement Artificial Intelligence (AI) to make the car completely self-driving.

Likewise, since this was our first hardware project, the parts kept on malfunctioning at times which wasted a lot of time. The poor-quality connectors and wires also created a hindrance in getting the desired result from the car or program.

Moreover, a major problem was finding the appropriate power source to supply sufficient power for the H-board and DC motors. Lack of power source hindered us from making the system wireless completely as we had to use AC to DC adapters for power supply.

## **Features:**

### **1. The car is autonomous i.e. it is driverless.**

It doesn't require any manual control to drive the car. Currently, the car can go on a straight path and avoid collision to obstacles.

### **2. The car detects obstacles and avoids them in their path.**

We have used an ultrasonic sensor to detect the obstacles. When an object is detected, a signal is sent to the GPIO pins which stops the car and hence avoids collision.

### **3. The car processes the live image and detects the path ahead.**

Using OpenCV library, we have developed a program which detects the lane ahead on the live video stream captured by the Minoru camera.

### **4. The car responds to stop sign.**

We have used Haar Cascade Classifier training to train the standard stop sign. Hence the car (camera) can detect it and hence respond accordingly. The car stops on detecting a stop sign.

## **Chapter 4: Cost Estimation**

S.N.	Equipment	Cost per Unit	Quantity	Total Cost
1	Minoru 3D Webcam	3000	1	3000
2	Raspberry Pi	4000	1	4000
5	L298N Motor Shield	550	1	550
6	Wheels	150	2	300
7	Motors	200	2	400
8	Chassis	500	2	1000
9	Power Bank (5V 1mAH)	1000	1	1000
11	Ultrasonic Sensor	300	1	300
			<b>Total</b>	<b>10550</b>

## Chapter 5: Result

## **1. Controlling the motors by Raspberry Pi**

We dismantled the car and removed the Arduino and used the onboard Raspberry Pi. This improved the system performance by the following:

### **A. Power consumption decreased.**

Removing the Arduino decreased the power consumption. Arduino need a 5V power supply but now the saved power can be used for powering the motor.

### **B. Reduction in number of IC**

Removing two Arduino from the system reduced two Atmega IC which ultimately reduced size of the embedded system

### **C. Reduction in cost**

Reducing the number of components helped in lowering the total cost of the system by Rs 8000.

### **D. Improved transfer of data between the system and the motor**

Previously an Arduino was used to control the motor. The data bits was generated on the system was transferred to the Arduino serially which then controlled the motor.

Now the motors are directly controlled by the Raspberry Pi which reduced the delay time.

## **Chapter 6: Conclusion and Recommendation**

Hence, we have been able to make the car self-driving that can move on a straight path avoiding collisions on detecting any obstacles, can detect the lanes of the road on live video stream captured by the camera and can take actions according to the signs that have been trained for the car. During the working phase, we learnt and gained knowledge on various topics like image processing with OpenCV library in python, hardware designing, working with raspberry pi, etc. So, this project proved to be very beneficial for us.

However, we couldn't implement artificial intelligence (AI) and make the car self-driving completely due to limited resources and time. We aim to eradicate the limitations that are mentioned below and enhance our project to its full extent. We hope the college will help us with the required resources to complete this project in future.

## **6.1. Limitation**

Although we put in our best efforts to complete this project there are few points where we lack like:

- The car moves only in forward direction.
- Artificial intelligence(AI) is yet to be implemented.
- The model is trained to only stationary object.
- The processing power of Raspberry Pi is not enough.
- The lane detection algorithm does not work on curved road.
- The system gets frozen sometimes during processing.
- The car cannot drive according to the detected lane.

## **6.2. Future Enhancement**

In order to implement Artificial Intelligence (AI) in our car, we need to train our navigation model, which needs a lot of processing power with powerful CPU and GPU. We can train our model to detect various objects and overcome a wide range of obstacles and situations while driving using large processing power to diversify our image processing capabilities. We can also train the car to drive in real roads by collecting a lot of training data. Provided that we get sufficient resources, we will surely achieve all the above additional features to make the car completely self-driving.

## Reference

- [1] Wang, Z. (2017, November 6). *Self Driving RC Car*. Retrived from <https://zhengludwig.wordpress.com/projects/self-driving-rc-car/>
- [2] *OpenCV Python Neural Network Autonomous RC Car*. (2017, November 9). Retrieved from <https://www.youtube.com/watch?v=BBwEF6WBUQs&feature=youtu.be/>
- [3] Chong, B. ,Yan, R. ,Li, A. (2017, November 9). *A modified remote control car that drives autonomously between lanes and avoids forward collisions*. Retrieved from [http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/bcc44\\_acl84\\_my259/bcc44\\_acl84\\_my259/index.html/](http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/bcc44_acl84_my259/bcc44_acl84_my259/index.html/)
- [4] Zakaria, M.A. ,Zamzuri\*, H.,Mamat, R. ,Mazlan, S.A. (2017, November 11). *A Path Tracking Algorithm Using Future Prediction Control with Spike Detection for an Autonomous Vehicle Robot*. Retrieved from <http://journals.sagepub.com/doi/full/10.5772/56658/>
- [5] (2017, November 9). *What are pros and cons of OpenCV and TensorFlow for computer vision?*Retrievedfrom <https://www.quora.com/What-are-pros-and-cons-of-OpenCV-and-TensorFlow-for-computer-vision/>
- [6] Oros, N.,Krichmar, J. (2017, November 12). *Android™ Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers*. Retrieved from <http://www.socsci.uci.edu/~jkrichma/ABR/>
- [7] Shimizu, S. , Ozaki, T. ,Yoshida, M. (2017, November 11). *A moving image processing system for personal vehicle system*. Retrieved from <http://ieeexplore.ieee.org/document/254489/>
- [8] Shrestha, A., Gajurel, K., Ranjit, P. (2017). *Stereo Vision Based Vehicle Navigation* (Bachelor dissertation).
- [9] Hunter, J. ,Dale, D. ,Firing, E. ,Droettboom, M. ,Matplotlib development team. (2017, November 19). *Image tutorial*. Retrieved from [https://matplotlib.org/2.0.0/users/image\\_tutorial.html#plotting-numpy-arrays-as-images/](https://matplotlib.org/2.0.0/users/image_tutorial.html#plotting-numpy-arrays-as-images/)
- [12] Rosebrock, A.(2017, November 23). *Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV* 3. Retrieved from <https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/>



- [13] Kar, A. (2017, November 24). *How to use IP Webcam with opencv as a wireless camera*. Retrieved from <https://thecodacus.com/ip-webcam-opencv-wireless-camera/>
- [14] Adrian Rosebrock. (2017, November 25). *How to install OpenCV 3 on Raspbian Jessie*. Retrieved from <https://www.youtube.com/watch?v=YStdNJwcovY/>
- [15] (). Mordvintsev, A. ,Abid, K. (2017, November 27). *Getting Started with Images*. Retrieved from [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_gui/py\\_image\\_display/py\\_image\\_display.html#display-image/](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html#display-image/)
- [17] T, Marvin. (2017, December 14). *KittiBox, A car detection model implemented in Tensorflow*. Retrieved from <https://github.com/MarvinTeichmann/KittiBox/>
- [18] (2017, December 15). *Openalpr, Automatic License Plate Recognition library*. Retrieved from <https://github.com/openalpr/openalpr/>
- [19] Sonam Puntsong. (2017, December 22). *Self Driving car project using Machine Learning*. Retrieved from <https://www.youtube.com/watch?v=Yceqk8vmXPM/>
- [20] (2018, January 2). *How do I send single character ASCII data to a serial port with python*. Retrieved from <https://stackoverflow.com/questions/3984602/how-do-i-send-single-character-ascii-data-to-a-serial-port-with-python>
- [21] doxygen. (2018, January 2). *Depth Map from Stereo Images*. Retrieved from [https://docs.opencv.org/3.1.0/dd/d53/tutorial\\_py\\_depthmap.html/](https://docs.opencv.org/3.1.0/dd/d53/tutorial_py_depthmap.html/)
- [22] mkokshoorn. (2018, January 2). *Computer vision project to get depth information from a pair of low cost webcams*. Retrieved from <https://github.com/mkokshoorn/Stereo-Webcam-Depth-Detection/>
- [23] Erget. (2018, January 2). *Building and calibrating a stereo camera with OpenCV (<50€)*. Retrieved from <https://erget.wordpress.com/2014/02/01/calibrating-a-stereo-camera-with-opencv/>
- [24] Ballew, G. (2018, January 12). *Building a lane detection system using Python 3 and OpenCV*. Retrieved from <https://medium.com/@galen.ballew/opencv-lanedetection-419361364fc0/>

- [25] Hardwock, M. (2018, January 12). *Simple Lane Detection with OpenCV*. Retrieved from <https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0/>
- [26] Fridman, L. (2018, January 15). *MIT 6.S094: Deep Learning for Self-Driving Cars*. Retrieved from <https://selfdrivingcars.mit.edu/>
- [27] Brailovsky, D. (2018, January 15). *Recognizing Traffic Lights With Deep Learning*. Retrieved from <https://medium.freecodecamp.org/recognizing-traffic-lights-with-deep-learning-23dae23287cc/>
- [28] Rosebrock, A. (2018, January 19). *Raspberry Pi: Deep learning object detection with OpenCV*. Retrieved from <https://www.pyimagesearch.com/2017/10/16/raspberry-pi-deep-learning-object-detection-with-opencv/>
- [29] Siver, D. (2018, January 25). *In-Depth on Udacity's Self-Driving Car Curriculum*. Retrieved from <https://medium.com/self-driving-cars/term-1-in-depth-on-udacitys-self-driving-car-curriculum-ffc46af0c08/>
- [30] Cmuedu. (2018). Cmuedu. Retrieved 24 July, 2018, from <https://web2.qatar.cmu.edu/~mndarwis/embeddedsystem.html>

## APPENDIX : Car Body Parts

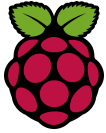


Fig Raspberry Pi



Fig DC Motor with Gear

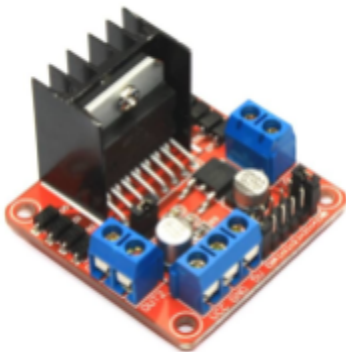


Fig L298N H-Bridge Motor Controller



Fig Ultrasonic Sensor