Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Project Report on

"AUTOCar"

COMP 303

(For partial fulfillment of 3rd Year/ 1st Semester in Computer Engineering)

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

Submitted to
Dr. Gajendra Sharma
Department of Computer Science and Engineering

Submission Date:
15th Feb, 2018

Bonafide Certificate

This project work on "AUTOCar" is the bonafide work of
" Manasi Kattel (18)
Araju Nepal (28)
Ayush Kumar Shah(44)
Deepesh Shrestha (50) "

who carried out the project work under my supervision.

Project Supervisor

_____

Name: Dr. Purushottam Kharel
Academic Designation: Associate Professor

Acknowledgement

First of all, we would like to express our heartfelt gratitude to our project supervisor Dr. Purushottam Kharel for for his support during the 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 "Arduino" 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.
3.3.1	Detection Flowchart	17

List of Table

Table No.	Table	Page No.
3.1.1	Software Specification	5
3.2.1.1	Python Libraries used	6
3.2.1.2.a	Arduino Functions used	7
3.2.1.2.b	Arduino Structures used	7

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.
Acknowledgement	**i**
Abstract	**ii**
List of figures	**iii**
List of tables	**iii**
Abbreviation	**iv**
Chapter 1 Introduction	**1**
1.1 Background	**1**
1.2 Objectives	**1**
1.3 Motivation and Significance	**2**
Chapter 2 Related Works	**3**
Chapter 3 System Design and Implementation	**5**
3.1 System Specifications	**5**
3.1.1 Software Specification	**5**
3.1.2 Hardware Specification	**5**
3.2 System Requirement Description	**6**
3.3 Methodology	**13**
Chapter 4 Discussion on the Achievements	**18**

Chapter 5 Conclusion and Recommendation **20**

Reference **22**

Appendix A : Car Body and Parts used **26**

Appendix B : Output **27**

Appendix C : Circuit Semantics **28**

Appendix D: Cost Estimation **30**

Appendix E: Methodology **31**

Chapter 1: Introduction

1.1. 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 arduino programming. It is based on hardware and software.

1.2. 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

1.3. Motivation and Significance

1.3.1. 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.

1.3.2. 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: Related Works

During our research, we encountered numerous project having similar concept. Some of them were prototype projects while some were large scale projects.

AutoRC Car (W,Zheng) is a modified radio controlled car which can handle three tasks: self-driving on the track, stop sign and traffic light detection, and front collision avoidance. The system consists of three subsystems: input unit (camera, ultrasonic sensor), processing unit (computer) and RC car control unit. A Raspberry Pi board (model B+), attached with a pi camera module and an HC-SR04 ultrasonic sensor is used to collect input data. The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction (steering), object detection (stop sign and traffic light), distance measurement (monocular vision), and sending instructions to Arduino through USB connection [1][2].

Stereo Vision based vehicle navigation is a project to build autonomous vehicle that detects lanes and obstacles for safe driving. The autonomous vehicle uses stereo camera named 'Minoru 3D webcam'. The webcam is used for stereo imaging to calculate depth. Stereo imaging is the process of taking two or more images and estimating a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths. The project uses neural network to take driving decisions. The inputs to the neural network are – bottom half of the image from left camera, bottom half of the disparity image, and output of two ultrasonic sensors. Resilient backpropagation (RPROP) algorithm is used to calculate the weights on the layers during training. The weights calculated are used in feed forward network to predict the output. Thus, predicted output is transmitted serially to Arduino. Arduino controls the motors and the autonomous vehicle follows the path[8].

A Moving Image Processing System for Personal Vehicle System (Tohru Ozaki, and Masumi Yoshida, Fujitsu Laboratories Ltd., Fujitsu Ltd. 1015 Kamikodanaka, Nakahara-ku, Kawasaki 211, Japan) details two moving-image processing algorithms, a visual-information processing we developed to guide a vehicle through white-line detection on a road, and our results. The detection algorithm was developed for worst-case conditions including rain, night, and shadows. The visual-direction decision algorithm enables the TV camera to obtain get optimum white-line images even when the PVS turned at an intersection or negotiated hairpin turns [7].

Chapter 3: Design and Implementation

3.1. System Requirement Specification

3.1.1. Software Specification

- **Programming Languages** Python, Arduino (set of C/C++ functions)
- **Operating System** Debian based OS (Ubuntu and Raspbian OS)
- **Image Processing Library** OpenCV
- **IDE** Arduino IDE
- **Desktop Sharing System** VNC
- **Remote Connection Tool** OpenSSH

3.1.2. Hardware Specification

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

3.2. System Requirement Description

3.2.1 Software Description

3.2.1.1 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

matplotlib	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.
cv2	cv2 is the python interface library from OpenCV. This has many functions such as imread(), imshow(), etc that helps in image manipulations.
pyserial	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

3.2.1.2 Arduino

Arduino programming language can be divided in three main parts: structure, values (variables and constants), and functions.

FUNCTIONS

pinMode()	Configures the specified pin to behave either as an input or an output. See the description of (digital pins) for details on the functionality of the pins.
digitalRead()	Reads the value from a specified digital pin, either HIGH or LOW.
digitalWrite()	Write a HIGH or a LOW value to a digital pin.
analogRead()	Reads the value from the specified analog pin.
analogWrite()	Writes an analog value (PWM wave) to a pin.
delay ()	Pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

VARIABLES

Various variables were used to write the arduino code.

STRUCTURE

setup()	The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup() function will only run once, after each power up or reset of the Arduino board.
loop()	After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

3.2.1.3 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, program the arduino and debug the program.

Raspbian OS was used on the Raspberry Pi for Image Processing and communication with the Arduino.

3.2.1.4 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).

3.2.1.5 Arduino IDE

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

3.2.1.6 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.

3.2.1.7 OpenSSH

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

3.2.2 Hardware Description

3.2.2.1 Raspberry Pi

The Raspberry Pi is open hardware, except for the primary chip on the Raspberry Pi, the Broadcomm 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.

3.2.2.2 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."

3.2.2.3 Arduino

Arduino is open-source hardware. Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino UNO is the Opti boot bootloader. Boards are loaded with program code via a serial connection to another computer. Arduino board designs

use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus(USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++.

3.2.2.4 Motor Shield

L298N H-bridge Dual Motor Controller Module 2A with Arduino. 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.

3.2.2.5 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.

3.2.2.6 Ultrasonic Sensor

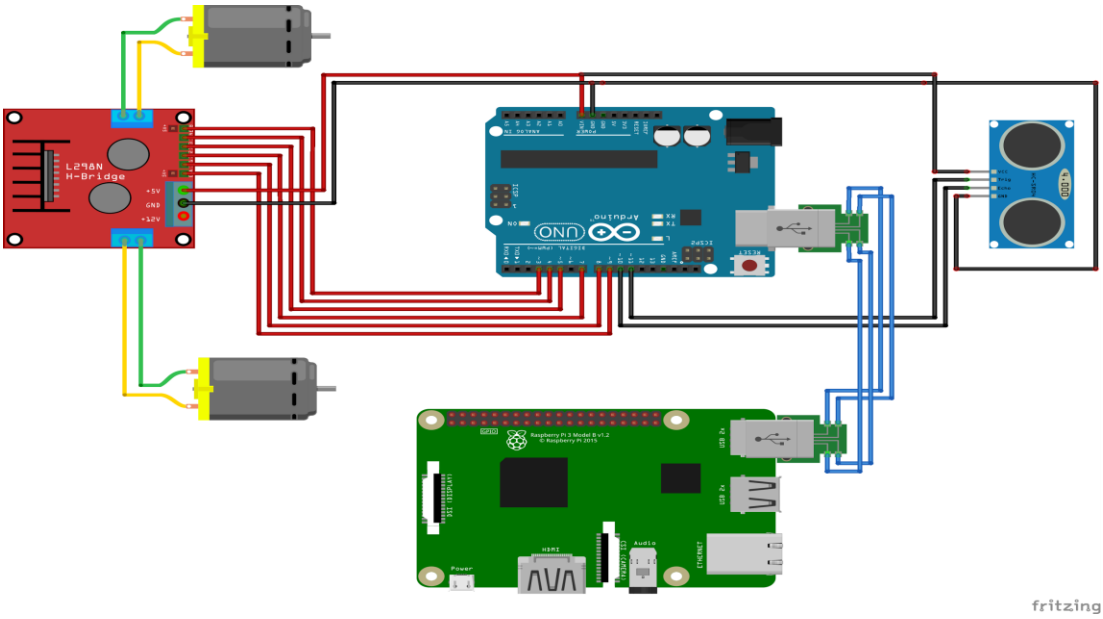
An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound 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.

3.2.2.7 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 Arduino, Motor and the motor shield.

3.2.2.8 Car Chassis and Body Parts

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



3.3 Methodology

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**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as

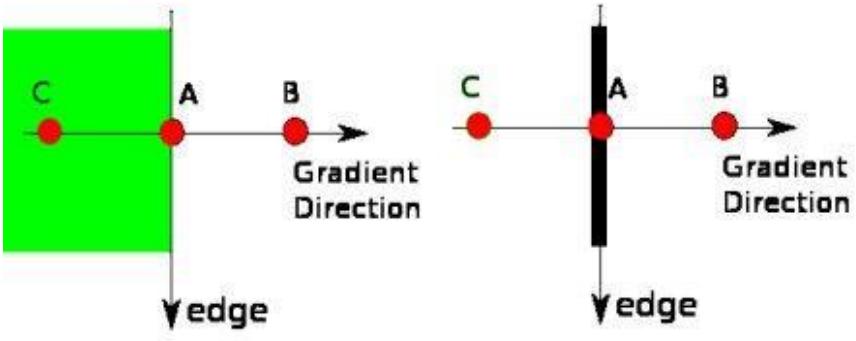
follows:

$$\text{Edge\_Gradient}(G)=\sqrt{G2x+G2y}\qquad \text{Angle}(\theta)=\tan^{-1}\left(\frac{Gy}{Gx}\right)$$

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. Check the image below:



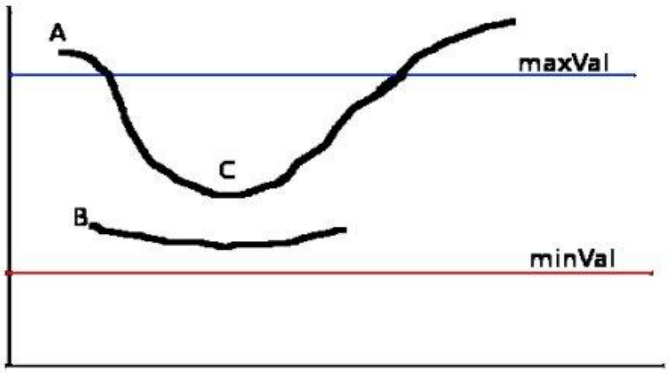
image

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, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below 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. See the image below:



image

The edge A is above the 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.

Chapter 4: 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 misfunctioning 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, Arduino 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 arduino 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 and traffic signals.**

   We have used Haar Cascade Classifier training to train the standard stop sign and a traffic light sample. Hence the car (camera) can detect them and hence respond accordingly. The car stops on detecting a stop sign or a red light and drives on detecting a green traffic light.

Chapter 5: Cost Estimation

S.N.	Equipment	Cost per Unit	Quantity	Total Cost
1	Minoru 3D Webcam	3000	1	3000
2	Raspberry Pi	6500	1	6500
3	Raspberry Pi Casing	1500	1	1500
4	Arduino	1500	2	3000
5	L298N Motor Shield	550	1	550
6	Cooler Fan	300	1	300
7	Wheels	150	2	300
8	Motors	200	2	400
9	Chassis	500	2	1000
10	Power Bank (5V 1mAH)	1000	1	1000

11	Sealed Battery	65	3	200
12	Ultrasonic Sensor	300	1	300
13	USB Cable (Type B)	300	1	300
			Total	**18350**

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 arduino programming, 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/

[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?* Retrieved from 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/

[10] Nedelkovski, D. (2017, November 21). *Ultrasonic Sensor HC-SR04 and Arduino Tutoria*l. Retrieved from http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[11] Lesson Studio. (2017, November 21). *HOW TO: control DC Motors with Arduino + L298N*. Retrieved from https://www.youtube.com/watch?v=kv-9mxVaVzE/

[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/

[16] A, Eric. (2017, December 12). *Arduino IDE error - avrdude: ser_open(): can't open device "/dev/ttyACM0": Permission denied*. Retrieved from http://arduino-er.blogspot.com/2014/08/arduino-ide-error-avrdude-seropen-cant.html/

[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/

[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-ffcf46af0c08/

APPENDIX : Car Body Parts

Fig Raspberry Pi

Fig Arduino

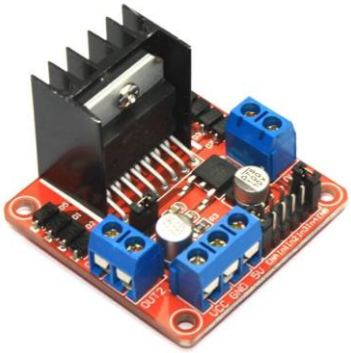


Fig L298N H-Bridge Motor Controller

Fig Ultrasonic Sensor

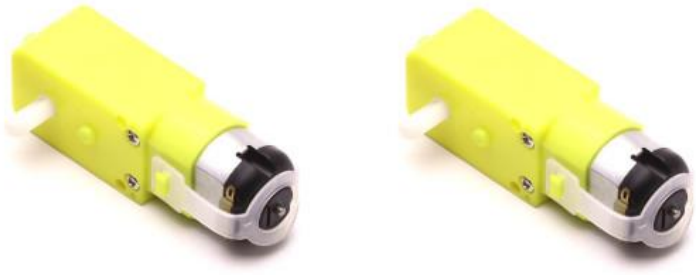


Fig DC Motor with Gear