Self Driving Car Using AI & ML

Prof. Sushma Wankhede, Kaustubh Kadam, Aditya Patil, Ashish Patil,
[1, 2] Dept of Instrumentation
[1, 2] B.V.C.O.E

Abstract- *The project aims to build a monocular vision autonomous car prototype using Raspberry Pi as a processing chip. An HD camera along with an ultrasonic sensor is used to provide necessary data from the real world to the car. The car is capable of reaching the given destination safely and intelligently thus avoiding the risk of human errors. Many existing algorithms like lane detection, obstacle detection are combined together to provide the necessary control to the car.*

Keywords- Raspberry PI, lane detection, obstacle detection.

I. INTRODUCTION

Rushing around, trying to get errands done, thinking about the things to be bought from the nearest grocery store has become a part of our daily schedule. Driver error is one of the most common cause of traffic accidents, and with cell phones, in- car entertainment systems, more traffic and more complicated road systems, it isn't likely to goaway.
With the number of accidents increasing day by day, it has become important to take over the human errors and help the mankind. All of this could come to an end with self-driving cars which just need to know the destination and then let the passengers continue with their work. This will avoid not only accidents but also bring a self-relief for minor day to day driving activities for small items.

II. HARDWARE DESIGN

List of Hardware

A pre-built four wheel drive (4WD) chassis is used as a base on which following hardware components are fit [9]:

- Raspberry Pi 3 Model B+ for GPU and CPU computations
- Jumper wires to connect individual components
- Camera case
- Picamera
- Ultrasonic sensor to detect obstacles

Hardware and software description

Raspberry Pi

The Raspberry Pi is a credit card-sized single-board computer.

In this project, we have used the model 3 B+. It comprises of a 1 GB RAM model with 4 USB ports and 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet.

Fig 1: Features offered in Raspberry Pi 3 Model B+

Pi Camera

It is the camera shipped along with Raspberry Pi [18]. Pi camera module is also available to which can be used to take high-definition videos as well as still photographs [18].

Ultrasonic Sensors

Ultrasonic sensors evaluate attributes of a target by interpreting the echoes from radio or sound waves respectively [1]. In this project, they are used to detect the distance of obstacles from the car [1].

Raspbian OS

Of all the operating systems Arch, Risc OS, Plan 9 or Raspbian available for Raspberry Pi, Raspbian comes out on top as being the most user-friendly, best-looking, has the best range of default softwares and optimized for the Raspberry Pi hardware [19]. Raspbian is a free operating system based on Debian (LINUX), which is available for free from the Raspberry Pi website [19].

Python

Python is a widely used general-purpose, high-level programming language [18,20, 21]. Its syntax allows the programmers to express concepts in fewer lines of code when compared with other languages like C, C++or java [20, 21].

RPi.GPIO Python Library

The RPi.GPIO Python library allows you to easily configure and read-write the input/output pins on the Pi's GPIO header within a Python script [18, 20]. This package is not shipped along with Raspbian.

OpenCV

It (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.

It has over 2500 optimized algorithms, including both a set of classical algorithms and the state of the art algorithms in Computer Vision, which can be used for image processing, detection and face recognition, object identification, classification actions, traces, and other functions [21]. This library allows these features be implemented on computers with relative ease, provide a simple computer vision infrastructure to prototype quickly sophisticated applications [20, 21].

The library is used extensively by companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota. It is also used by many research groups and government [21].
It is based on C++ but wrappers are available in python as well. In our project is used to detect the roads and guide the car on unknown roads [21].

Hardware Components Connection

The 4 wheels of the chassis are connected to 4 separate motors. The motor driver IC L293D is capable of driving 2 motors simultaneously [22]. The rotation of the wheels is synchronized on the basis of the sides i.e. the left front and left back wheels rotate in sync and right front and right back- wheels rotate in sync. Thus the pair of motors on each side is given the samedigital input from L293D at any moment. This helps the car in forward, backward movements when both side wheels rotate in same direction with same speed. The car turns when the left side wheels rotate in opposite direction to those in right[22].

The chassis has two shelves over the wheels separated by 2 inch approx. The IC is fixed on the lower shelf with the help of two 0.5 inch screws. It is permanently connected to the motor wires and necessary jumper wires are drawn from L293D to connect to Raspberry Pi [9, 22]. The rest of the space on the lower shelf is taken by 8 AA batteries which provide the power to run themotors.

To control the motor connected to pin 3 (O1), pin 6 (O2), the pins used are pin 1, pin 2 and pin 7 which are connected to the GPIOs of Raspberry pi via jumper wire

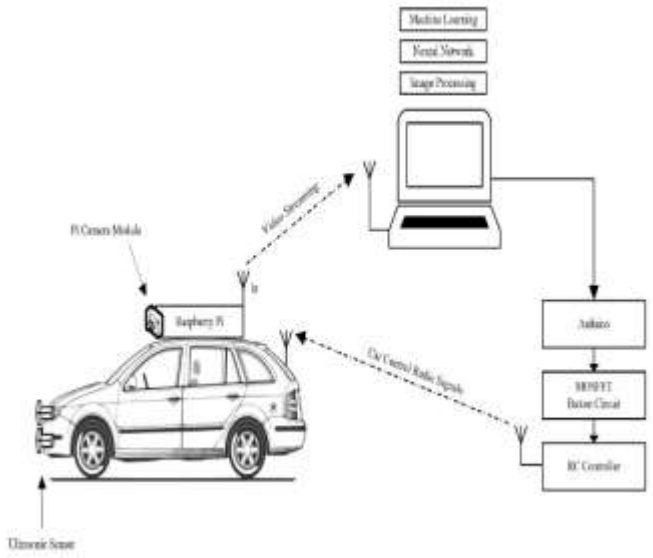

Fig2: Block Diagram

Fig 3: Hardware Connections

The raspberry pi case is glued on the top shelf along with the

L shaped aluminum strip. The pi is fit in the case and the aluminum strip gives the support to the camera fit on servo motor and the ultrasonic sensor [1, 18, 20].
The Wi-Fi dongle is attached to the USB port in Raspberry Pi in order to connect to it wirelessly.

The complete connection of the raspberry pi with motor controller L293D can be found in fig 2[9, 22]. Since raspberry pi needed its own IP, it needs to be connected to a Wi-Fi router or Hotspot [9]. For the same we need to make some changes in the field specified so as to make raspberry pi recognize the router every time it boots up.

III. OVERVIEW & OBJECTIVE

In recent years, both the public and industry's interest in autonomous cars have increased dramatically. The reason that our group decided to pursue such a project is due to our shared interest in this matter. The goal of this project is to provide the team with hands on experience on creating an autonomous car that has basic self-driving capabilities. Most importantly, we learned and understood the enormous challenges it takes to create a fully functional autonomous car safe for real-life use. Despite our project is only a simple model, we aimed to perfect each of our capabilities - road tracking, traffic sign detection and collision avoidance.

Input Stage - Pi Camera & Ultrasonic Sensor

Our project uses Picamera and ultrasonic sensor as our input stage, the Picamera and ultrasonic sensor are connected to the Raspberry Pi and communicates with the laptop via TCP socket, each input has its unique port number and same IP address, which is our laptop current IP address. The Picamera takes 10 frames per second, and send the streamed image frames to the laptop, the image data is used to achieve the road tracking and traffic signs detection. The ultrasonic sensor is HC-SR04, and its operating voltage is 5V. As the Raspberry Pi gets the sensor data, the data will send to the laptop. The sensor is put on the front of car, it is used to measure the distance of the front object in order to avoid collision.

Training Stage - Artificial Neural Network (ANN) & Classifier

After making sure that we can stream real-time video into our own computer by frames, we began to collect data and then train the data by using two simple machine learning algorithms which are 3-layer Artificial Neural Network for road tracking and Haar feature-based cascade classifiers for object detection.

For road tracking part, we aimed to achieve that car can predict four directions while driving by classifying road frames token in real time into those four categories thus it can run automatically by tracking the road. Every input frame will be reshaped into 320*120 size with 38,400 pixels. Every pixel is used as one neuron xk in the input layer. The hidden layer will have 32 neurons as the basic feature extraction. For the output layer, we will have four neurons for the four different directions which are forward, left, right, and reverse.

Fig. 4 3-layer neural network consisting of input layer, hidden layer, and output layer.

We wrote two python scripts to collect frames data first and to train data for direction prediction. In details, we built white road and drove the car running along the road manually by using the four-direction keys in our keyboard to control the directions for the car. While car was running, it would stream the real-time video back to our laptop and when we clicked the right decision for one input frame by using pygame module, the frame would be added into image array as the data for the input layer and the decision will be added into label array as the date for the output layer. All the data will be kept in file. After collecting more than 4500 frames each time, we splitted all the frames into training data (0.6) and testing data (0.4). We put all the training data into training python script which we used the 3-layer ANN module with backpropagation in OpenCV to get the right prediction parameters. At the same time, testing data can help us see the performance of the prediction parameters. We have trained for numerous times and finally got the 96% accuracy for training and 93% accuracy for testing.

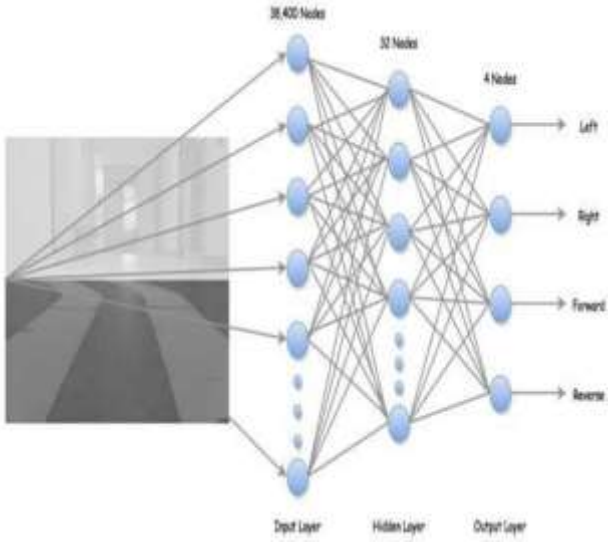

Fig 3:Neural Networks Prediction

Forward and Right

Right

Forward and left

Fig 4: Actual Paper Paths

Below shows the training data collection process. First each frame is cropped and converted to a numpy array. Then the train image is paired with train label (human input). Finally, all paired image data and labels are saved into anpz file. The neural network is trained in OpenCV using back propagation method. Once training is done, weights are saved into a xml file. To generate predictions, the same neural network is constructed and loaded with the trained xml file.

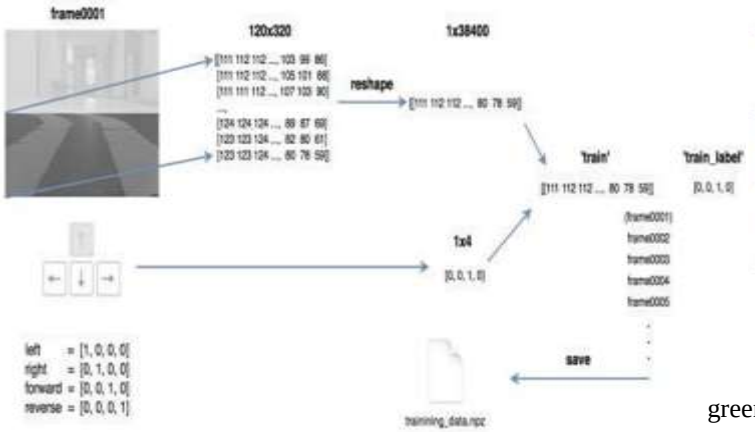

Fig 5: Numpy Arrays.

Object Detection

This project adapted the shape-based approach and used Haar feature-based cascade classifiers for object detection. Since each object requires its own classifier and follows the same process in training and detection, this project only focused on stop sign and traffic light detection.

OpenCV provides a trainer as well as detector. Positive samples (contain target object) were acquired using a cell phone, and were cropped that only desired object is visible. Negative samples (without target object), on the other hand, were collected randomly. In particular, traffic light positive samples contains equal number of red traffic lights and green traffic light. The same negative sample dataset was used for both stop sign and traffic light training. Below shows some positive and negative samples used in this project.

Fig 6: Actual Prototype.

Type	Number of positive sample	Number of negative sample	Sample size (pixels)
Stop sign	20	400	25x25
Traffic light	26	400	25x45

Fig 6: Cascade Classifier Training

To recognize different states of the traffic light(red, green), some image processing is needed beyond detection. Flowchart below summarizes the traffic light recognition process.

Fig 7:TrafficLight Recognition

Firstly, trained cascade classifier is used to detect traffic light. The bounding box is considered as a region of interest (ROI). Secondly, Gaussian blur is applied inside the ROI to reduce noises. Thirdly, find the brightest point in the ROI. Finally, red or green states are determined simply based on the position of the brightest spot in the ROI.

Distance Measurement

Raspberry Pi can only support one pi camera module. Using two USB web cameras will bring extra weight to the

RC car and also seems unpractical. Therefore, monocular vision method is chosen.

This project adapted a geometry model of detecting distance to an object using monocular vision method proposed by Chu, Ji, Guo, Li and Wang (2004).

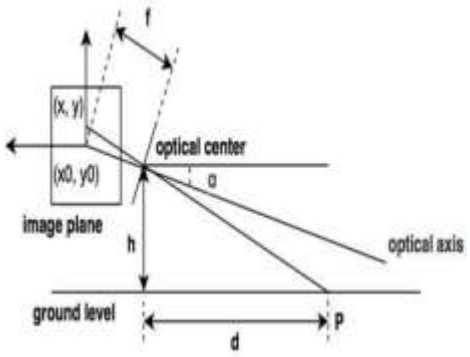

Fig 8: Distance Measurement Using PI camera.

P is a point on the target object; d is the distance from optical center to the point P. Based on the geometry relationship above, formula (1) shows how to calculate the distance d. In the formula (1), f is the focal length of the camera; ∂ is camera tilt angle; h is optical center height; (x0, y0) refers to the intersection point of image plane and optical axis;

(x, y) refers to projection of point P on the image plane. Suppose O1 (u0,v0) is the camera coordinate of intersection point of optical axis and image plane, also suppose the physical dimension of a pixel corresponding to x-axis and y-axis on the image plane are dx and dy. Then:

$$d = h/\tan(\partial + \arctan((y - y_0)/f)) \quad (1)$$

$$u = \frac{x}{dx} + u_0 \quad v = \frac{y}{dy} + v_0 \quad (2)$$

Let $x_0 = y_0 = 0$, from (1) and (2):

$$d = h/\tan(\alpha + \arctan((v - v_0)/\alpha_y)), \quad (\alpha_y = f/dy) \quad (3)$$

v is the camera coordinates on y-axis and can be returned from the object detection process. All other parameters are camera's intrinsic parameters that can be retrieved from camera matrix.

OpenCV provides functions for camera calibration. Camera matrix for the 8MP pi camera is returned after calibration. Ideally, a x and a y have the same value. Variance of these two values will result in non-square pixels in the image. The matrix below indicates that the fixed focal length lens on pi camera provides a reasonably good result in handling distortion aspect. Here is an interesting article discussing the focal length of pi camera with stock lens and its equivalent to 35mm camera .

$$\begin{bmatrix} a_x = 331.7 & 0 & u_0 = 161.9 \\ 0 & a_y = 332.3 & v_0 = 119.8 \\ 0 & 0 & 1 \end{bmatrix}$$

The matrix returns values in pixels and h is measured in centimeters. By applying formula (3), the physical distance d is calculated in centimeters.

RC Car Control Unit

The RC car used in this project has an on/off switch type controller. When a button is pressed, the resistance between the relevant chip pin and ground is zero. Thus, an Arduino board is used to simulate button-press actions. Four Arduino pins are chosen to connect four chip pins on the controller, corresponding to forward, reverse, left and right actions respectively. Arduino pins sending LOW signal indicates grounding the chip pins of the controller; on the other hand sending HIGH signal indicates the resistance between chip pins and ground remain unchanged. The Arduino is connected to the computer via USB. The computer outputs commands to Arduino using serial interface, and then the Arduino reads the commands and writes out LOW or HIGH signals, simulating button-press actions to drive the RC car.

IV. RESULTS

Prediction on the testing samples returns an accuracy of 85% compared to the accuracy of 96% that the training samples returns. In actual driving situation, predictions are generated about 10 times a second (streaming rate roughly 10 frames/s).

Haar features by nature are rotation sensitive. In this project, however, rotation is not a concern as both the stop sign and the traffic light are fixed objects, which is also a general case in real world environment.

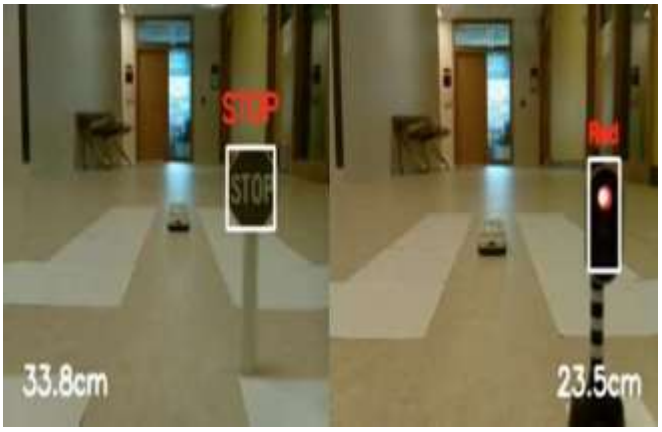

Fig 9: Actual Output.

For distance measurement aspect, the ultrasonic sensor is only used to determine the distance to an obstacle in front of the RC car and provides accurate results when taking proper sensing angle and surface condition into considerations. On the other hand, Pi camera provides "good enough" measurement results. In fact, as long as we know the corresponding number to the actual distance, we know when to stop the RC car. Experimental results of detecting distance using pi camera are shown as below:

Order	1	2	3	4	5	6	7
Actual value (cm)	15	20	25	30	35	40	45
Measured value (cm)	15.5	19.7	24.1	27.5	32.0	35.2	39.0

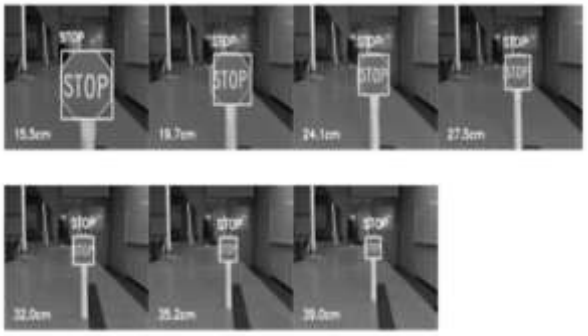

Fig 10: Pi cam vs Ultrasonic Distance Measurement

In this project, the accuracy of distance measurement using monocular vision approach could be influenced by the following factors: (1) errors in actual values measurement, (2) object bounding box variations in detecting process, (3) errors in camera calibration process, (4) nonlinear relationship between distance and camera coordinate: the further the distance, the more rapid change of camera coordinate, thus the greater the error.Overall, the RC car could successfully navigate on the track with the ability to avoid front collision, and respond to stop sign and traffic light accordingly.

REFERENCES

[1] OpenCV Documentation -Haar Feature- based Cascade Classifier for Object Detection
[2] OpenCV Documentation - Cascade Classifier Training
[3] NaotoshiSeo - Tutorial: OpenCVhaartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)
[4] Material for NaotoshiSeo's tutorial
[5] OpenCV Answers- "about traincascadeparemeters, samples, and other…"
[6] OpenCV Answers - "memory consumption while training > 50GB"
[7] David J Barnes on Robotics & Mechatronics - OpenCVHaarTraining - Object Detection with a Cascade of Boosted Classifiers Based on Haar-like Features - Part I
[8] David J Barnes on Robotics & Mechatronics - OpenCVHaarTraining - Object Detection with a Cascade of Boosted Classifiers Based on Haar-like Features - Part II
[9] github.com/foo123/HAAR.js
[10] github.com/mtschirs/js-objectdetect
[11] github.com/inspirit/jsfeat
[12] Computer Vision Software - FAQ: OpenCVHaartraining
[13] StackOverflow - haar training OpenCV assertion failed