AUTOMATED DETECTION OF DRIVING PATHWAY USING IMAGE PROCESSING

Poonam Priyadarshini
ECE
B.I.T. Patna
Patna, India
poonampriyadarshini@bitmesra.ac.in

Avish Jha
CSE
V.I.T. Vellore
Vellore, India
avish.jha@protonmail.com

Mayank Raj
ECE
B.I.T. Patna
Patna, India
mayankraj8091@gmail.com

2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.203

Abstract— Image data is one of the most popular real world input data that can be used for variety of applications ranging from robotics and computer vision to security systems. In combination with other methods such as neural network, Artificial neural network and image processing techniques, manipulation of image data can lead to applications such as detection of objects, tracking, identification and vision based robotics and so on. Advanced Driver Assistance System (ADAS) also use image for camera based driver assistance systems.

The report covers a hardware model system that tests the software work of detection of traffic signs and path for it own ADAS systems. Different problems were tackled, including the choice of OS, and additional hardware components needed to tackle. The choice of programming languages, equipment, OS and methods were based on simplicity and practicality. Artificial neural network in combination with Open CV libraries were used for stop sign, traffic light and path road detection. The hardware model consisted of RC Car attached to raspberry pi board with a mounted pi camera for video streaming and an arduino controller attached to a radio transmitter for controlling through Open CV running in windows PC.

Keywords— *Advanced Driver Assistance System, raspberry, Classifiers, driving technology.*

I. INTRODUCTION

The lack of proper enforcement of traffic rules in our country is one of the biggest reasons for roads accident. In 2015 and 2018, reports showed that India had an average mortality rate of 11.6 per 100,000 people just due to traffic accidents alone. The data thus shows the lack of satisfactory average safe driving environments in our country. To tackle this problem, strict enforcement of traffic rules can be one of the solutions, but this requires enforcing large traffic police. Implementation of method that is efficient and can be applied on majority of drivers of the country, tries to improve their traffic habits and prevent accidents on its own. The system that meets these requirements is the Advanced Driver Assistance System (ADAS) [1].

Through ADAS systems, the car itself supports the driver to perform tasks that are necessary and critical for his safety. In hilly areas in India there is lack of protective railings on the cliff side and car deviates from its path and results in fatal accident. Since testing new ADAS systems on real cars is impractical, developing a miniature hardware model for testing purposes is one of the logical ways to go. Miniature hardware models and simulation helps in the testing of such systems [2-3]. Since new variables are constantly met with on the roads and due to its nature of unpredictability, it is essential that the ADAS uses learning such machine learning technique instead of pre-defined program. The input data for such miniature hardware model can be anything but image data is reliable, inexpensive and intuitive and together with neural networks can be used for creating and implementing ADAS systems on such models. Development of ADAS with features such as traffic light detection, road sign detection and lane detection would greatly diminish accidents as a large chunk of road accidents occurs due to these failures.

II. LITERATURE SURVEY

Aditya Jain in his work [1] proposed a working model of self-driving car which is capable of driving to different types of tracks such as curved tracks, straight tracks and straight followed by curved tracks. A camera module is mounted over the top of the car along with Raspberry Pi sends the images from real world and used Convolution Neural Network which predicts other directions. Arpad Takacs et.al [3] give detailed description of self driving technology, the level of autonomy of the car, the assistance systems, basic theory behind making a workable autonomous car, industries which are interested in these technologies . David Singleton [7] demonstrates the use of using even a cheap model RC Car in successfully modeling a driver assistance system and testing it. Erik Coelingh et.at [8] outline why simulation is not a good modeling practice for a self driving car and discusses the advantages of training in real world data, especially for the car.

Previous versions of this work were attempted on macOS and Linux systems. The macOS system was implemented by Zheng Wang, while the one using Linux and android were implemented prior to that. We tried to implement this system on windows hardware and attempted to know the accuracy and efficiency of such setup on a windows system. The use of optical isolators is an addition to previously made Wang's setup which is a necessary improvement since the radio TX-2B doesn't function properly due to electrical interference between direction pins. The use of optical isolators is inspired from relay boards, since they too use optical isolators for the same purpose.

III. SOFTWARE TECHNIQUES USEFUL IN ADVANCED DRIVER ASSISTANCE SYSTEM

ADAS systems require certain algorithms and software techniques for its implementation. ADAS inputs can be of various types ranging from LIDAR, X-Ray to image or video streaming data. Here focus is on image data as image input give more information and variation of any of the parameters such as contrast, brightness, color mapping, etc. which leads to numerous data samples that can be used for further manipulation and processing. Neural Network is used for Image processing and image classification. Neural networks are one of the learning algorithms used within machine learning. They consist of different layers for

analyzing and learning data. When the layers number in a neural network is increased, learning is more and pattern detection is more accurate. Neural Networks learn and attribute weights to the connections between the different neurons each time the network processes data. Artificial neural networks use back propagation as a learning algorithm to compute a gradient descent with respect to weights. For training a machine learning model gradient descent algorithms is used. It is used in case of supervised training model. Newton's Method is a second-order optimization algorithm as it makes use of the Hessian matrix. It is an extension of the gradient-based delta learning rule. In this after finding an error (the difference between desired and target), the error is propagated backward from the output layer to the input layer via the hidden layer. It is used in case of Multilayer Neural Network. Deep learning algorithms are associated with Artificial Neural Networks. *It* is a subset of machine learning, which uses neural networks with many layers. In machine learning, the algorithm is given a set of relevant features to analyze, however, in deep learning, the algorithm is given raw data and derives the features itself. One of the most basic learning algorithms used in ANN is the delta rule used for the backpropagational neural networks (BPNN). Here learning is a supervised process where the machine or network tries to learn at every epoch or cycle, where each cycle is marked by a different input pattern. Based on how much different the guessed answer is from the actual one, the network makes adjustment to its weight.

Object detection in the form of traffic light detection and stop sign detection is done through trained neural classifiers. Every label of direction is attached with the sample image and neural training is done based on that. The neural network algorithm in the training code causes the 38400 pixels of the image to correspond to one of the three direction labels. The predicted direction labels on the testing samples are compared with the actual ones, letting the algorithm to calculate the training and the validation accuracy. Using neural network the car was manually run on track through keyboard arrow presses. Each arrow press caused a capture of a video frame as image. The image is labeled with corresponding direction key pressed. After few hundreds of images are taken, training is done through Artificial Neural Network whose functions are present in OpenCV library of python. After training, a trained ".xml" model is generated. Detection of path corners based on the ".xml" neural network model was generated. Car turns right or left based on the corners detected. A trained neural classifier of the traffic light was used for the detection. RC car stops at red light and moves at green light. Trained neural classifier of the stop sign was used for the detection purposes. RC car stops at stop sign

Training data is gathered from image samples taken in hundreds through a camera. A software model is created after training whose accuracy can be tested through further image samples taken which can be termed as the test samples. Successful classification of images through neural networks can applied for purposes where different classes of images should trigger different output responses. The basic methodology behind machine learning techniques include: using pre-classified training examples and known attributes to develop a search method, using a search method defined on the training samples to generate a classifier inducing algorithm After the classifier inducing algorithm or the evaluation function is generated, it is applied on new unclassified test samples to create classified image sets with a respectable accuracy.

IV. WORKING OF THE MODEL

The idea is that the model toy car is to act as a stand in for a car in real life situations. We need the car to do three basic things: a) Try to remain inside lane or road for cases where the car accidently leaves the road due to human error b) Follow traffic light rules which is stopping at red light and moving at green c) Stopping at the presence of stop sign for situations where stop signs are placed on road in case of constructions or restricted areas.

The data collection for such is done from a single versatile input source which is the camera. The camera used here is the pi camera module which is interfaced to the raspberry pi board [4]. The raspberry pi board will act as the stand-in client computer that which will be attached to the car for video streaming. The windows pc will act as the stand in server. All the video streamed from the pi camera is processed by the server computer due to the server computer's superior processing speed and power. Similar server client model of ADAS can be found in Google's self driving car for reference. The car is steered by remote radio signals from its radio transmitter which is interfaced to the arduino uno microcontroller for intelligent automation [5-7]. Based on input video stream data, the car is intelligently controlled through serial output from arduino, which by itself is controlled through pygame library of python at the software level. The pygame library is what interfaces the software model to its hardware part. Object detection such as traffic light detection or the stop sign detection is done through trained neural network classifier. These are .xml files that employ machine learning to enable systems to differentiate between objects in images and identify them. The lane detection is done by training the car on track with different categories of path images labeled by different direction keys on the pygame.

V. BASIC SOFTWARE MODEL AND WORKING

The software model to be implemented on the proposed hardware uses programming language Python 2.7 in the windows PC system for all the image processing and computer vision work [8]. It also uses ANN to achieve its objectives for intelligent detection of path. Object detection in the form of traffic light detection and stop sign detection is done through trained neural classifiers. TCP networking and socket programming was used for connecting the windows PC application process with the raspberry Pi video streaming application, which ran in python 3. The hardware was controlled physically through arduino microcontroller which received inputs from the running from the running software through through serial input.

VI. SOFTWARE IMPLEMENTATION, TESTING AND TRAINING

A. Uploading Arduino Sketch program

The arduino program for keyboard control of RC Car is uploaded into the arduino board using the Arduino

IDE. The program makes sure that the digital I/O pins are kept at HIGH for every corresponding key press on the keyboard and LOW after the key is released. For example when the forward key is pressed on the keyboard, the corresponding arduino pin which controls the forward pin of the radio controller goes HIGH. This is done as the arduino receives the serial input data from the computer (which is the keyboard key press). Thus by keeping the digital I/O pins HIGH/LOW, the corresponding direction pins of the TX-2B IC is kept at LOW/HIGH through the optocoupler channel circuit.

B. *Keyboard control*

The testing of keyboard control of car is done by implementing pygame library functions in python. The key presses are registered by the python program and the resultant output is passed serially to the arduino board, which is still attached to the server windows PC. Each key press causes the corresponding pin in the arduino to go HIGH and releasing the key causes it to go LOW.

C. *Camera video streaming*

Connecting Pi to PC using VNC viewer: The Raspberry Pi microcomputer is switched on and connected to the windows PC server through WiFi using mobile hotspot [9]. The first step is to determine the ip address of the Pi. For the VNC viewer to connect to the ip address of the raspberry pi, we need its ip address. The ip address of the raspberry pi is found by using the command *sudo ifconfig* statement in the terminal of the pi [10].

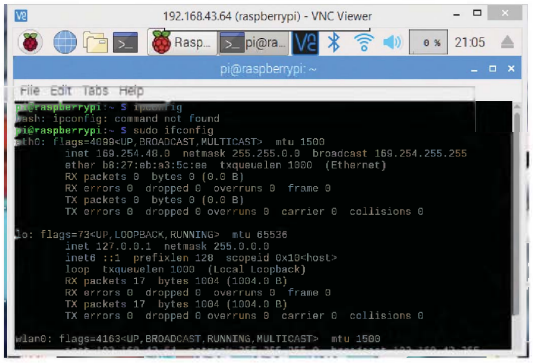

Figure-6.1 Using the sudo ifconfig command

Figure 6.1 shows how the *sudo ifconfig* command is used to the know the wlan0 ip address of the ip. After the ip address was found we opened the VNC viewer application to write the ip address of the raspberry pi for it to connect to the computer.

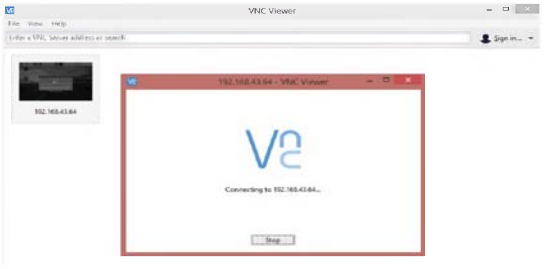

Figure-6.2 Connecting Pi to PC through VNC viewer

When the connection is complete the raspbian OS of the Pi loads into the VNC viewer as shown in Figure 6.3

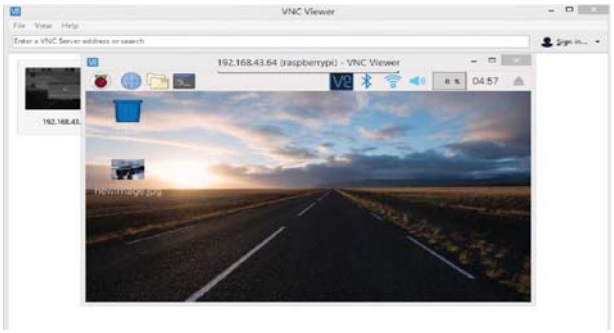

Figure- 6.3 Accessing Raspberry Pi OS with windows PC using VNC viewer

The raspberry pi OS [11-12], which is the raspbian, is accessed using the VNC viewer. VNC or Virtual Network Computing is graphical desktop sharing software used for remotely controlling another computer from one's computer. To access the raspberry Pi through VNC viewer of windows, the corresponding VNC server application should be activated in raspbian OS of the raspberry Pi as shown in Figure 6.4

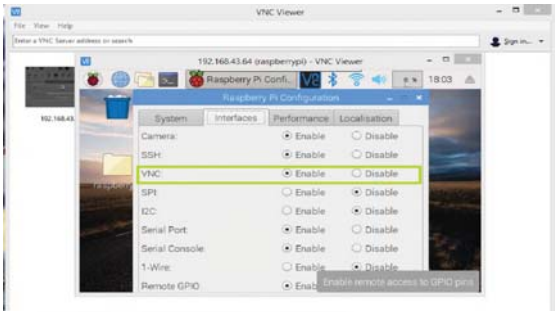

Figure 6.4- Enabling VNC server on Raspbian OS Pi

D. *Camera streaming*

The streaming is done through the wifi connection. The server camera streaming python program run on windows PC while the client streaming program is run on raspberry Pi usinf the VNC viewer. The camera streaming is done in 320x240 resolutions. Streaming is done in both grayscale and color format. The grayscale format is for training whereas the color format is used for user use and viewing.

Table-1 IP addresses of the server and the client

Serial Number	Server WLAN IP address	Client WLAN IP address
1	192.168.43.183	192.168.43.64

The table 1 outlines the WLAN ip addresses found for the server and client systems that are needed to connect the systems through wifi connection. The ip address of the server which is the PC is known by writing the *ipconfig* command at DOS terminal.

VII. TRAINING OF THE MODEL FOR INTELLIGENT MANEUVERING

The software model [13-14] for intelligent maneuvering is trained through manual runs of car along the path by collecting images streamed when doing those test runs. Key presses are needed for the car to move. During the same time, image data samples is collected using an algorithm such that when the user presses the keys to move the car, the camera takes images of the path. For each direction key press, a sample image is taken. The images fames are stored and labelled with their corresponding direction key labels. Successful training is determined when the car is able to turn automatically when it is at the edge of the path. The software model training work of the collected labelled images is done internally after the data collection is complete. For this we use an ANN algorithm and related functions of Python.

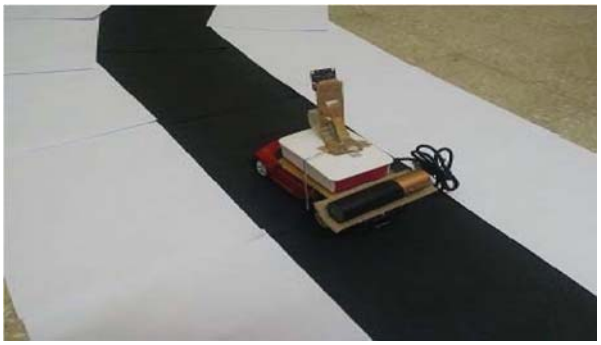

Figure-6.5 Training of the model for forward direction

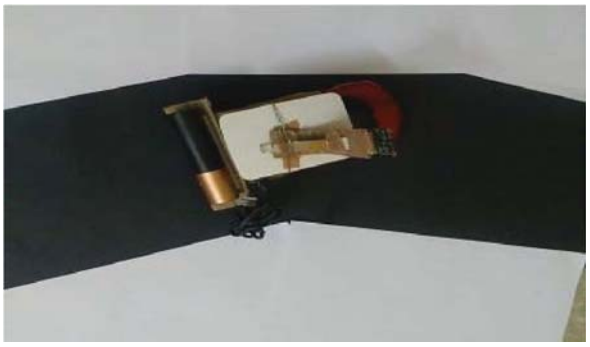

Figure-6.6 Training of the model for corners.

Figure 6.5 and Figure-6.6 shows the different situations the car experiences. At extreme corners, the car needs to turn in either left or right direction for it to remain within the road or path.

The basic philosophy when training the model is letting the model differentiates between situations where the car is at the extreme edge or corner of the path or turn and where the car is at the middle of the said path. When the model is at the extreme corner of the black path, the camera captures only the white surrounding area and not the path it runs on, and thus the path occupies only 20% or less of the image in such situations. For straight paths, the path occupies more than 20% of the image. The key presses accordingly make the model understand when to turn forward and when to turn in a different direction. Similar differentiation between left and right directions can be done with further training. Every label of direction is attached with the sample image and neural training is done based on that. The neural network algorithm in the training code causes the 38400 pixels of the image to correspond to one of the three direction labels. The predicted direction labels on the testing samples are compared with the actual ones, letting the algorithm to calculate the training and the validation accuracy .

VIII. TRAFFIC LIGHT AND STOP SIGN DETECTION

After the trained intelligent maneuvering is done, neural network classifiers [15-17] of stop sign and traffic light is used for the detection of stop sign, red light and green light using pi camera. The car stops or moves based on the signs it detects. For the detection of the traffic light, a traffic light module was used as given figure

Figure-6.7 Traffic light module

Figure-6.8 Stop Sign

The stop sign was made from a taking a photocopy of a standard stop sign image taken from google images and pasting it on cardboard.

The neural network classifiers for these two signs are haar cascade classifiers. These haar cascade classifiers are trained by taking 50 or more positive sample images if the sign that needs to be trained and about ten times the number of negative samples which can be anything other than the sign itself. The trained haar cascade classifier is then used for the detection of the traffic lights and the stop sign according to which the car stops and moves accordingly.

Haar cascade classifier was also used for differentiating between the green and red light. Since the traffic lights have a fixed position starting from red at the top to green at the bottom, the haar cascade classifier was made according to their position on the traffic light module rather than their color. This was important as only a grayscale input image was enough causing less load on the training machine and quick processing. When detecting the traffic light, the camera first detects the shape of the traffic light module and its color black, which is totally consistent with grayscale

input image. After the detecting the traffic light, the individual lights are then detected leading to detection of the red light and the green light.

IX. RESULTS

A. *Training and Validation accuracy of the data collected for intelligent maneuvering*

Table 2 Model training of the collected input data in three different times

Sl No.	No of image samples	Training Accuracy	Validation Accuracy	Training Duration
1	169	88.98%	90.20%	91.54s
2	205	91.61%	93.55%	124.22s
3	196	33.58%	42.57%	104.80s

The sample images are inputted in array form of dimensions (no. of image samples, 38400), where 38400 stands for number of pixels on the image. The neural classifier array is outputted in the form (no. of image samples, 4) where 4 correspond to the different four directions. The total number of sample images was divided into training set and testing set for determining the training and validation accuracy. The number of sample images is the number of images captured by the pi camera registered with a label whenever a key press event took place on the keyboard during the training of the car. A key press to move the car also led to capturing of the image with a label the same as the key that was pressed. For example, if forward key was pressed, an image would be taken of the track and that image would be labeled with the label 'forward'. The same way image would be taken with right and left key. The total samples were then divided into two parts, the first part is the training samples, which are the samples from which is the training model is created. When this trained model is applied on the training samples, the training accuracy is determined. The other parts are called the testing sample which is used to test the model created during training. When the trained model is applied on the testing samples, validation accuracy is calculated. For good results, the validation accuracy should always be more than the training accuracy. The training duration is nothing more than the time the model took to train itself.

X. DETECTION OF TRAFFIC SIGNS

A. *Stop sign detection*

The trained neural classifier of stop sign was used for its detection. In addition, distance of the stop sign from the car was measured after it is detected: The model car stops after the stop sign is detected remains stopped till it keeps on detecting the stop sign. As soon as the stop sign was removed, the car started to move which is what is required.

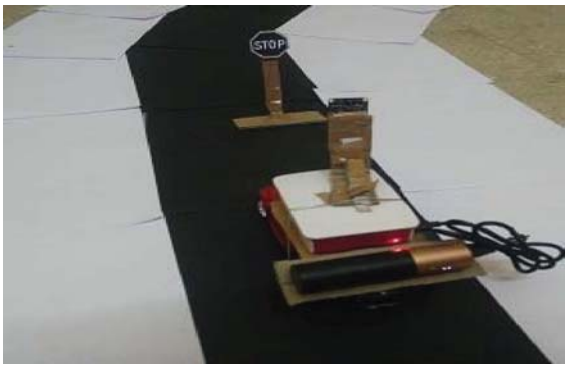

Figure-6.9 Stop sign kept in front of the model car

Figure-6.10 Detected stop sign with distance measurement

B. *Traffic Light Detection*

Trained neural classifier was used for the traffic light detection. Differentiation between red light and green light was done based on this neural classifier. For detection purposes a traffic light module was used. Figure 6.11 shows the traffic light module used for the detection.

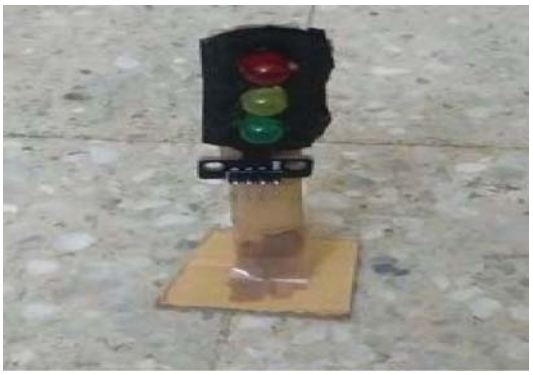

Figure-6.11 Traffic light module

A traffic light module was used for the detection purposes. The module was powered using 5V power supply and kept in upright position using cardboard backbone.

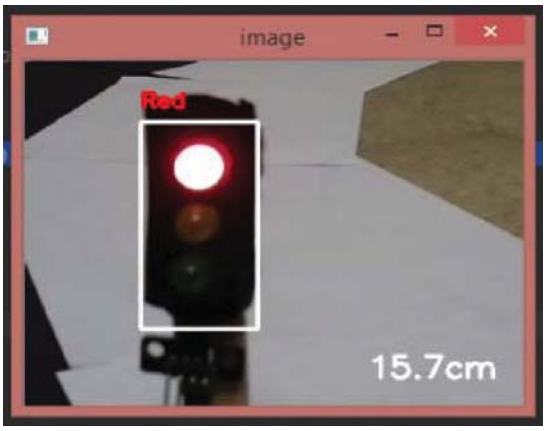

Figure-6.12 Red Light Detection

The detected red light output image is given in Figure 6.12. The car stops as soon as it detects the red traffic light.

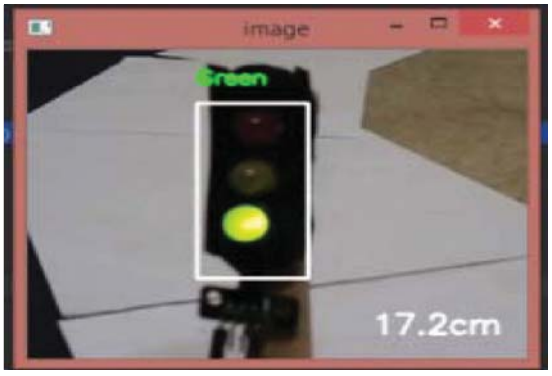

Figure-6.13 Green Light Detection

The green light detected output image is shown in Figure 6.13. When the green light is detected car keeps on moving. The minimum distance for detection is kept at 30 cm due to keeping in mind the delay produced when streaming through mobile hotspot wifi, which has greater latency and not as reliable as a standard router one.

XI. DISCUSSIONS

The latency during the video streaming of the pi camera greatly affects the accuracy of the results. Since the project uses mobile hotspot as the medium of WiFi, a not so satisfactorily latency definitely has some impact on the accuracy and proper working of the model. For our case, the latency was not too great for it to be unfeasible and using mobile hotspot worked just fine, but using standard WiFi medium of connection such as from powerful routers will be much preferable. It could be possible that the poor training accuracy in the third row of table might be because of this factor but it may not be the only.

The speed of the model RC car is not variable for models that require variable car speed for certain ADAS applications. Using expensive variable speed RC Car or making your own variable speed donkey car can be an option, but they both are expensive and so for the work needed to be done in this project, a simple RC Car was enough.

XII. CONCLUSION

The model designed for the detection of the signs and pathway had a lot of ups and downs. We used a basic four direction RC car that didn't have speed control because of which the accuracy of training and the accuracy of the results were affected. Excessive speeds caused the model to go too fast before it could detect corners, turns, or signs. Using a resistor solved the problem to some extent but using an expensive speed control RC controller would be a much better option. Donkey car projects are highly efficient too but they are also too expensive to be applicable. Even with these problems the model worked satisfactorily in detecting traffic signs and path. Latency of the video streaming could be improved using a router instead of mobile hotspot. Processing speed of the server PC could be improved by using a high end computer.

REFERENCES

[1] Aditya Kumar Jain, "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino", 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018).

[2] Ahmad Adamu Galadima, "Arduino as a learning tool", 2014 IEEE

[3] Arpad Takacs, Imre Rudas, Dominik Bosl, and Tamas Haidegger, "Highly Automated Vehicles and Self-Driving Cars", IEEE Robotics and Automation Magazine December 2018.

[4] Bhaumik Vaidya, Ankit Patel, Anand Panchal, Rangat Mehta, Krish Mehta and Parth Vaghasiya, "Smart home automation with a unique door monitoring system for old age people using Python, OpenCV, Android and Raspberry pi", International Conference on Intelligent Computing and ControlSystems ICICCS 2017

[5] Brian Markwalter, "The Path to Driverless Cars", IEEE Consumer Electronics Magazine April 2017

[6] Chris Urmson and William "Red" Whittaker, Carnegie Mellon University, "Self-Driving Cars and the Urban Challenge", IEEE intelligent systems published by IEEE computer society2008.

[7] David Singleton, "How I built a neural network controlled self-driving (RC) car!", https://blog.davidsingleton.org/

[8] Erik Coelingh and Jonas Nilsson, "Driving tests for self- driving cars", March 2018 spectrum.ieee.org

[9] Fatma Salih and Mysoon S.A. Omer, "Raspberry pi as a Video Server", International Conference on Computer, Control, Electrical, and Electronics Engineering 2018.

[10] Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan and Matthew Johnson Roberson, "Failing to Learn: Autonomously Identifying Perception Failures for Self-driving Cars", IEEE Robotics and Automation Letters. Preprint Version, Accepted July, 2018

[11] Mehmetcan Guleci and Murat Orhun, "Android Based WiFi Controlled Robot using Raspberry Pi", 2nd International Conference on Computer Science and Engineering, 2017.

[12] Mike Daily, Swarup Medasani, Reinhold Behringer and Mohan Trivedi, "Self-Driving Cars", IEEE magazine 2017.

[13] Ruturaj Kulkarni, Shruti Davalikar and Sonal Bangar, "Traffic Light Detection and Recognition for Self Driving Cars using Deep Learning", Fourth International Conference on Computing Communication Control and Automation, IEEE 2018.

[14] Souhail Guennouni, Anass Mansouri, Ali Ahaitouf Sidi Mohammed and Ben Abdellah, "Multiple Object Detection using OpenCV on an Embedded Platform", IEEE 2014 .

[15] Yihuan Zhang, Jun Wang, Xiaonian Wang, and John M. Dolan, "Road-Segmentation-Based Curb Detection Method for Self-Driving via a 3D-LiDAR Sensor", IEEE transactions on intelligent transportation systems.

[16] Zhang Lei, Zhang Xue-fei and LIU Yin-ping, "Research of the Real-time Detection of Traffic Flow Based on OpenCV", International Conference on Computer Science and Software Engineering, IEEE 2008.

[17] Vu Truong Thanh and Yoshiyori Vrano, "Mobile TCP socket for secure applications", The 12th International Conference on Advanced Communication Technology (ICACT) 2010.