© IJCIRAS | ISSN (O) - 2581-5334
August 2019 | Vol. 2 Issue. 3

DESIGN AND IMPLEMENTATION OF SERVER BASED SELF-DRIVING CAR

Thiri Kywe[1], Thae Thae Ei Aung[2], Hnin Ngwe Yee Pwint[3]

[1]Assistance Professor, Department of Electronic Engineering, Technological University (Meiktila), Myanmar
[2]Lecturer, Department of Electronic Engineering, Technological University (Meiktila), Myanmar
[3]Lecturer, Department of Electronic Engineering, Technological University (Meiktila), Myanmar

Abstract

Self-driving car is a vehicle that can guide itself without human interaction. The car is capable of reaching the given destination safely and intelligently thus avoiding the risk of human erroring this paper aims to build server-based self-driving car prototype using server for processing of data. The car is designed to autonomously navigate through appropriate paths by detecting lanes, stop sign and obstacles through real-time high-definition videos from Pi camera. Raspberry Pi is used to control L298N motor driver and driver module controls DC motors for the movement of the car. The motor driver also controls the car's speed based on the distance calculation result between the car and front objects. All the computations are handled by server and python is used for image processing. Lane detection, stop sign detection and object detection algorithms are combined together to provide the necessary control to the car.

Keyword: Raspberry Pi, server, L298N motor driver, DC motors

1.INTRODUCTION

Self-driving car is promising solution to enhance road safety, traffic issues and passengers. Such car requires vision algorithms that demands computers with high-speed processing capabilities. Control system including camera and server are capable of analyzing data to identify appropriate paths, obstacles and relevant signs. This self-driving car is very useful in transportation areas where the paths are defined. The car will be able to navigate with no human intervention to predetermined destination over roads that have not been adapted for its use. It will also provide safety of human lives, and leisure time instead of driving. In this system, the server-based self-driving car is designed to autonomously navigate through appropriate paths by detecting lanes and stop sign and obstacles.

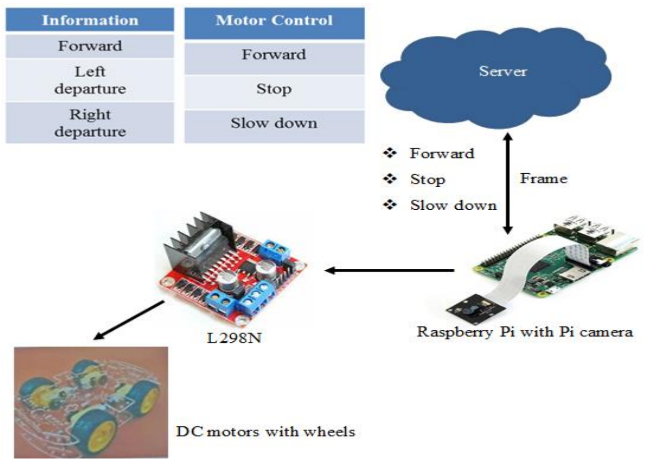


Figure 1. Overall Diagram of Server-Based Self-Driving Car

Figure 1 depicts the overall diagram of the server-based self-driving car. The proposed system is designed for self-driving car prototype that can autonomously detect and identify lanes, stop sign and obstacles using image processing techniques. Raspberry Pi camera is used to capture the real-time high-definition videos to the car. The motion JPEG (MJPEG) image frames streamed from Pi camera are sent to the server through the Raspberry Pi. Then, the server applies the image processing techniques to MJPEG streaming frames and provides information to Raspberry Pi in order to move forward, slow down or stop. L298N motor driver module is controlled by Raspberry Pi according to the commands

sent from server. The driver module is used to control the movement of the car and it also controls the car's speed based on the distance calculation result from Pi camera to front objects.
When the lanes are detected, the car will be moved forward along with the lanes. In Lane detection, the image frames taken from Pi camera are color segmented and then computed the edge detected image using Canny edge detection. Next, Hough transform is used to detect lines and then the car will be moved forward along the lanes according to the detected lines. If the vehicle is detected, the car will be slowed down else the stop sign and person are detected, the car will be stopped. In stop sign detection, Haar cascade algorithm is used. In Object detection, the image frames taken from Pi camera are predicted according to trained data from database by using the convolutional neural network (CNN) algorithm. This proposed system especially focuses on lane detection, stop sign detection and motor control.

2.OPERATION OF SYSTEM DESIGN

The design of the system is divided into two parts:

- ❖ The system workflow of self-driving car
- ❖ The procedure of the system

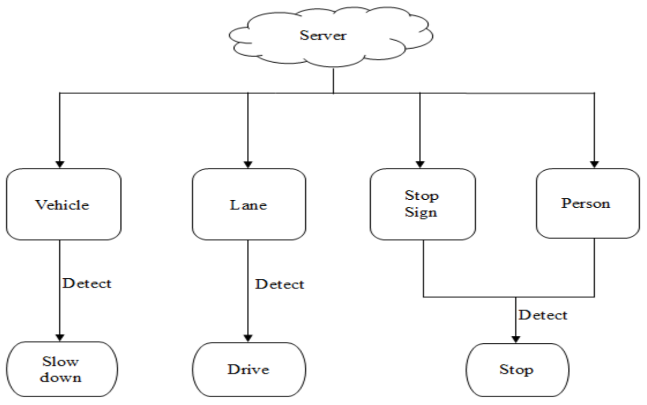


Figure 2. The Workflow Block diagram at the Server

The system workflow is Raspberry Pi camera firstly captures the real-time videos and then sends the image frames to the server by MJPEG streaming. MJPEG format compresses each video input frame as a JPEG image and streams JPEG frames over IP-based network from Pi camera to server. At the server, MJPEG image frames are applied the image processing algorithms and the respective algorithms are carried out. When the input image frames are lanes, color segmentation, edge detection and Hough line detection algorithms are processed to track the lane. If the image frames are stop sign, Haar cascade algorithm is applied else the frames are the person and vehicles, convolutional neural network algorithm is carried out.
This system uses the server for processing of images processing algorithms and provides the information and commands to Raspberry Pi. The interaction between the server and Raspberry Pi enables by using the WebSocket Protocol. The WebSocket protocol provides the full-duplex communication channels and enables the two-way ongoing conversion between the Raspberry Pi and the server to be instantly distributed with lower overheads, facilitating real-time data transfer from and to the server.
Raspberry Pi controls L298N motor driver module in order to move forward, stop or slow down according to the commands received from the server and the driver module controls the DC motors for the movement of the car. When the driver module receives the forward commands, the car will be moved forward along the lane. If the driver attains the stop commands, the car will be stopped else the module obtains slow down commands, the car will be slowed down.

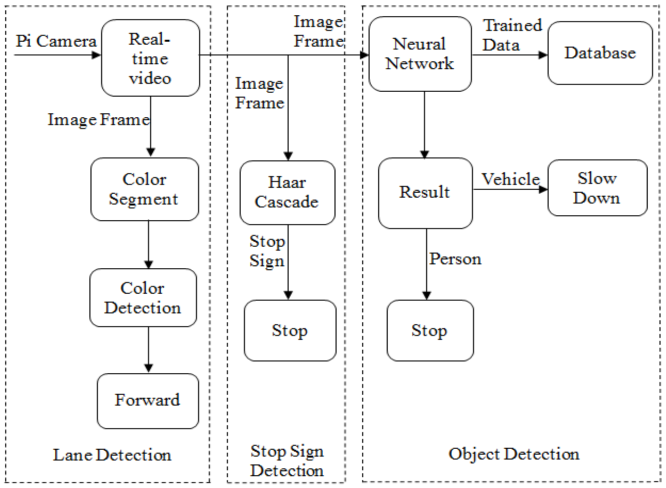


Figure 3. Procedure Block Diagram of Server-Based Self-Driving Car

3.INSTALLATION FOR COMPLETE DESIGN

At first, pi camera captures the real-time video and compresses the input video as image frames. Next, distance between the car and object on its way is calculated and then the threshold value is set to compare with calculated distance value in order to determine whether the lane or Haar cascade algorithm is applied to the input image frames. The front object distance is less than the threshold value, the image frames are applied to Haar cascade algorithm whereas the distance is greater than the threshold value, the frames are provided to the lane detection algorithm. The distance calculation will be discussed in next article. After processing the algorithm, generate the forward or stop commands according to respective algorithm to control the movement of the car.The four wheels of the chassis are connected to four separate motors and the other hardware components are fixed on the chassis. The following Figure 4 shows the complete design of server-based self-driving car prototype.

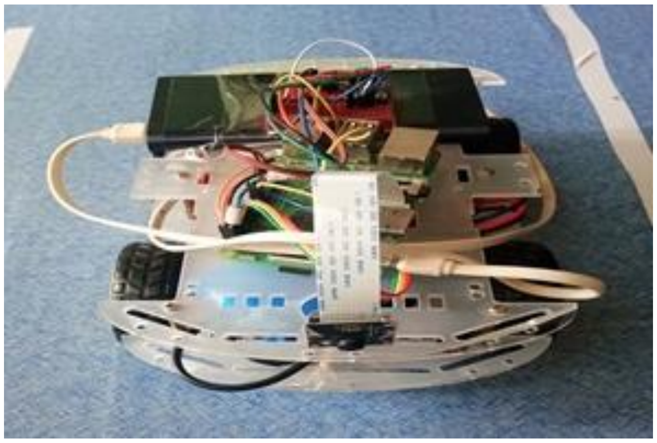


Figure 4. Complete Design of Server-Based Self-Driving Car Prototype

4.TESTS AND RESULTS OF LANE DETECTION AND STOP SIGN DETECTION

The simulation results, prototype results and real-time video testing are also included with respective figures.

4.1. Atom IDE

The server-based self-driving car system is tested by using Atom Software IDE and python image library is used to compute multi-dimensional array of incoming images. Atom is a text editor that is modern, approachable, yet hackable to the core. A tool can customize to do anything but also use productively without ever touching a config file.

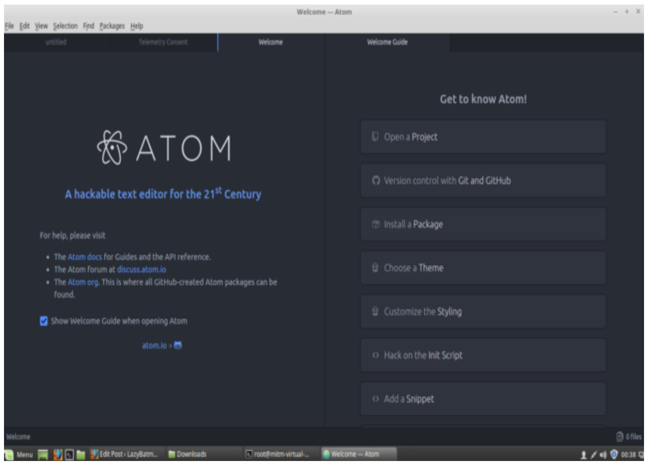


Figure 5. Atom IDE

4.2 Simulation Result for Lane and Stop Detection

Before testing the prototype, the simulations of lane and stop-sign detection are carried out by using the lane detection algorithm and Haar cascade algorithm respectively. In simulation, testing with stop-sign detection in real-time indoor condition and testing with the lane detection in input image frames. Figure 6 shows input image frames of original images.

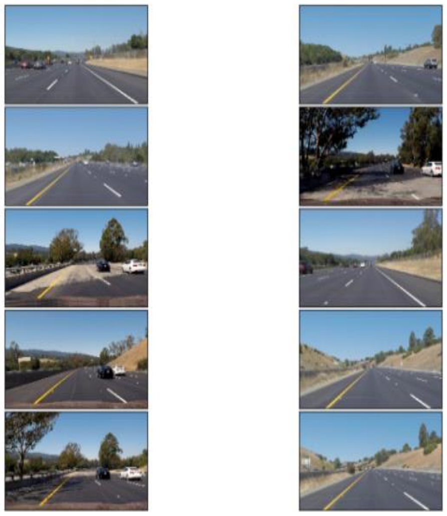


Figure 6. Original Images

The images are loaded in RGB color space. In the images, the white and yellow lines are lanes which need to track.

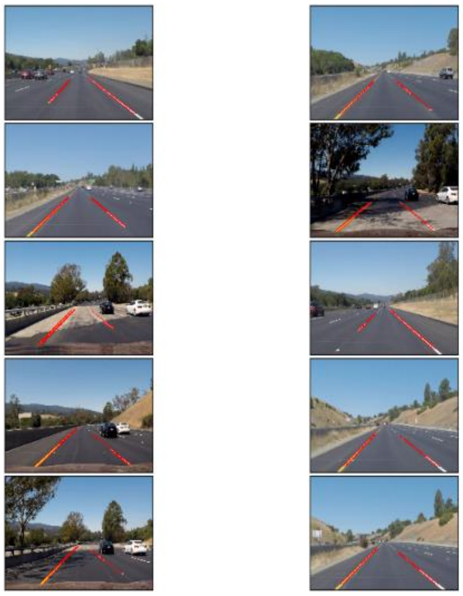


Figure 7.Simulation Result of Line Detected Images

Figure 7 describes the line detected image result tested from image frames input. The final line detected images have two lane lines, the left lane line and right lane line. The car will move along the lane according to the detected lane lines.

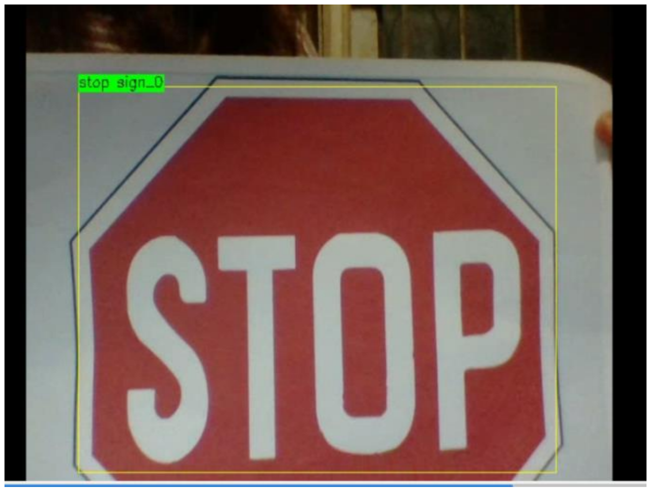


Figure 8.Simulation for Stop Sign Detection

Figure 8 shows test and result of neural network server result in indoor security to test accuracy for stop sign detection.

4.3. Prototype Result for Lane and Stop Sign Detection

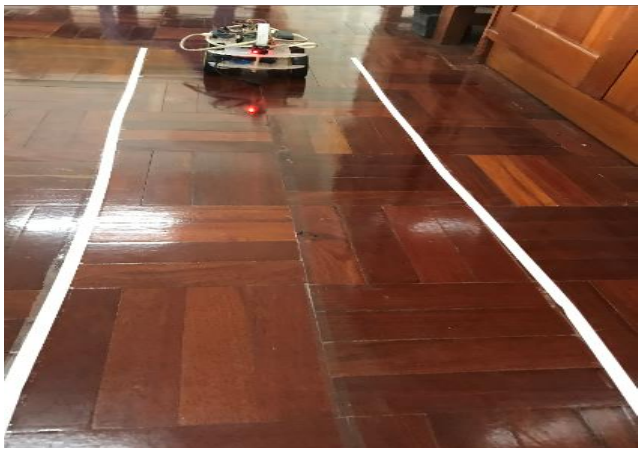


Figure 9.Testing Prototype for Lane Detection

Figure 9 shows the testing of prototype for lane detection. The prototype will move forward along the lane until it detects the obstacles on its way.

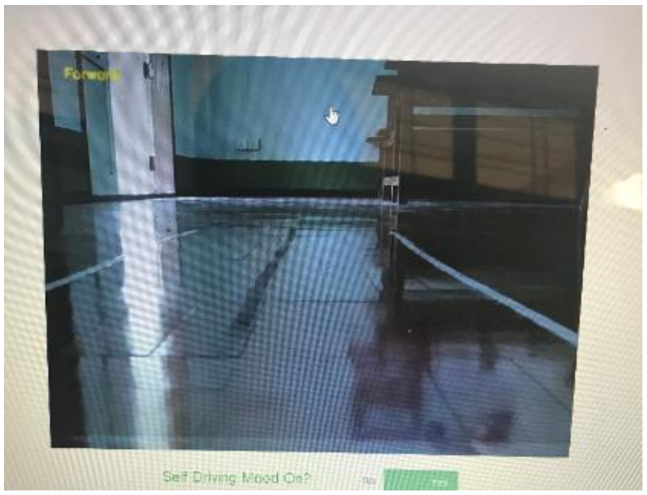


Figure 10. Result of Moving Forward Viewed from User Interface

Figure 10 shows the result of lane detection and lane detected information. The neural network server sends lane information via web based user interface (UI) during self-driving mode on.

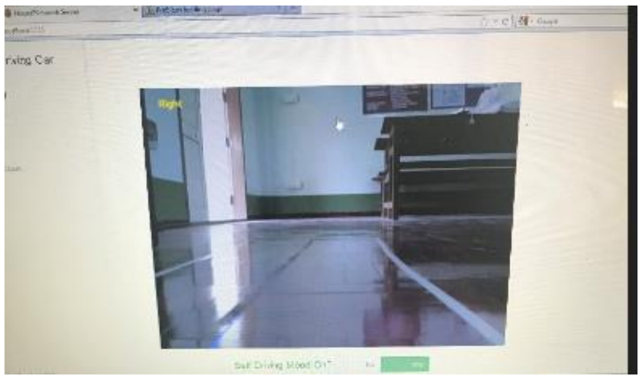
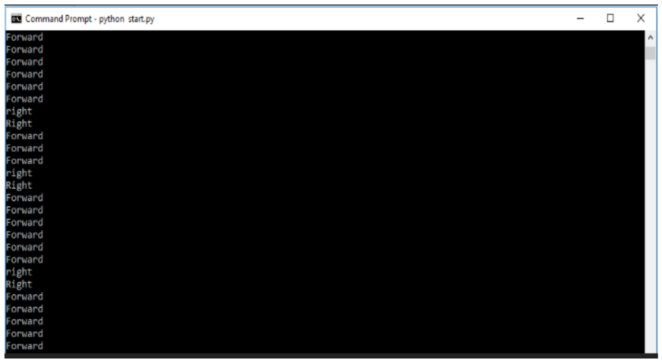


Figure 11. Result of Right Lane Information Viewed from UI

Figure 11 shows the result of detection of right lane and show information via web based user interface. In this system, detected lane path shows to vehicle as information and the car will be moved forward when it detects the lane.



```
Command Prompt - python start.py
Forward
Forward
Forward
Forward
Forward
Forward
right
Right
Forward
Forward
Forward
right
Right
Forward
Forward
Forward
Forward
Forward
Forward
right
Right
Forward
Forward
Forward
Forward
Forward
```

Figure 12. Viewing Result from Command Prompt of the Server

Figure 12 depicts the result of lane detection and information viewed form command prompt of the server.

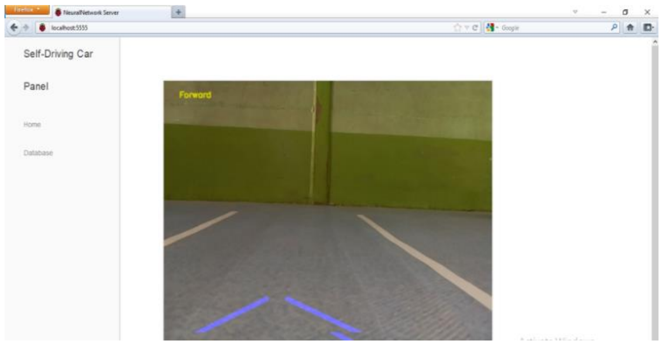


Figure 13. Result of Line Detection Viewed from UI

Figure 13 shows the result of line detection and lane detected information viewed from user interface (UI). The prototype will be moved forward between the detected lines.

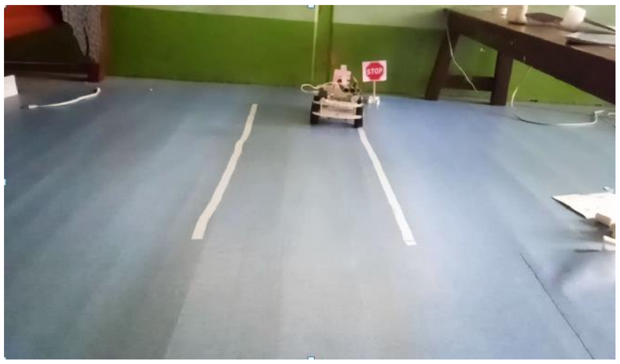


Figure 14. Testing Prototype for Stop Sign Detection

Figure 14 shows the testing of prototype for stop sign detection by using Haar cascade algorithm. The prototype will be stopped when it detects the stop sign on its way.

The prototype will be stopped when it detects the stop sign on its way.

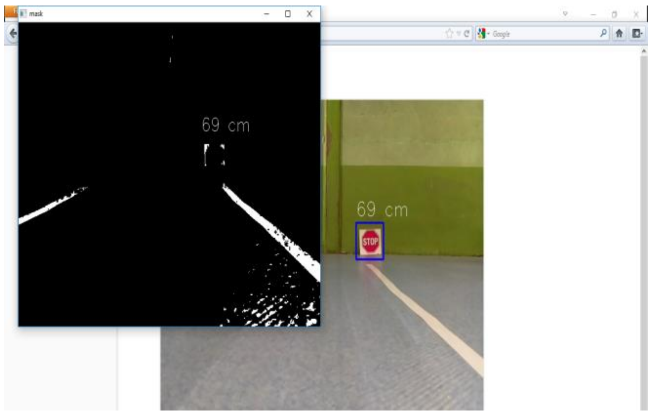


Figure 15. Result of Stop Sign Detection with Mask Viewed from UI

Figure 15 depicts prototype result of stop sign detection with mask viewed from UI.

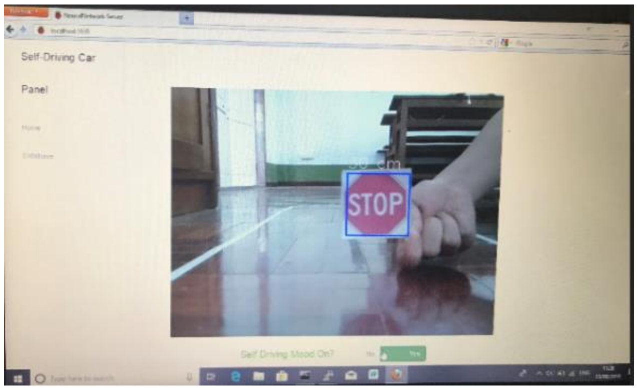


Figure 16. Result of Stop Sign Detection with Distance Viewed from UI

Figure 16 shows the result of stop sign detection with distance viewed from web page UI during self-driving mode on. In this system, when the distance between prototype and stop sign is 30cm, the car will be stopped.

4.4. Result from Real-time Video

In this system, user can choice manually self-driving mode with manual on/off mode. This prototype can work well in real world and simulation environment.

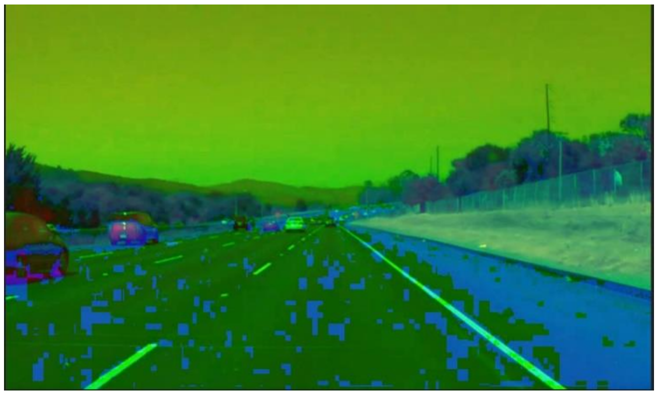


Figure 17. Result of Color Spaced Image from Real-Time Video

Figure 17 depicts the color space image result of the real time video, so the lane detection algorithm used in prototype also works in real-time conditions.

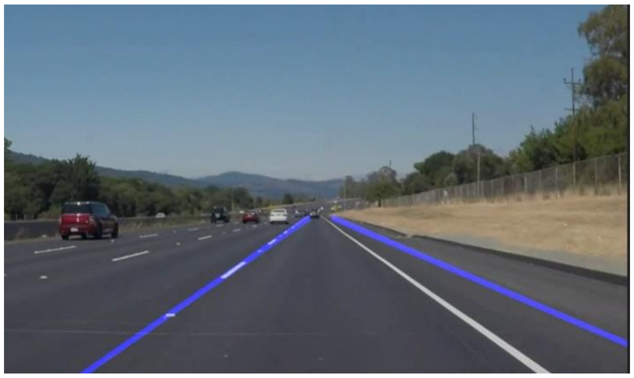


Figure 18. Result of Line Detection from Real-Time Video

Figure 18 shows that the result of line detection in real-time conditions. The self-driving car will be moved forward between the detected lines.

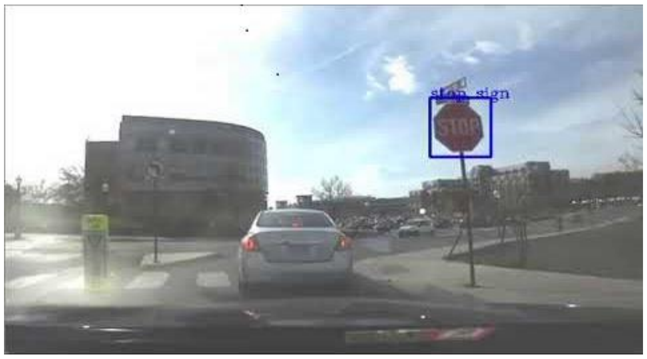


Figure 19. Result of Stop Sign Detection from Real-Time Video

Figure 19 depicts the result of stop sign detection in real-time video. The self-driving car will be stopped when it detected stop sign on its way.

5.CONCLUSION

The algorithms mentioned in implementation of lane and stop sign detection has been successfully implemented on a server-based self-driving car prototype. This system provides intelligence server-based self-driving car system based on image processing techniques. Although there are various kinds of the autonomous car with different technologies such as GPS, sensors, voice-controlled and so on, this self-driving car prototype is implemented using the image processing algorithms. That is why the advanced image processing techniques play an important role in man's everyday life. The car can capable of reaching the given destination safety and intelligently thus avoiding traffic accidents caused by human error. The self-driving car is very useful in transportation areas where the paths are defined. So, self-driving car can reduce number of car accidents, save lives and make highways, road and streets a safer place to travel through.

REFERENCES

[1] Image Segmentation Using the Color Thresholder, (2018). http://www.mathworks.com.
[2] Dronebot,Workshop:DC motor L298N Motor Driver, (2017).
[3] Adrain Rosebrock, Object Detection using Haar Cascades, (2016).
[4] Abraham Michelen, Michael Davis, Jonathan Ashdown, Raspberry Pi Training, (2015).
[5] Anonymous: RGB Color Code Chart, (2015). http://www.rapidtables.com/web/color/RGBColor.htm.
[6] Dave Jones, Picamera Documentaton Release 1.10, (2015).
[7] Adrian Rosebrock, Find distance from camera to object/maker using Python and OpenCV, (2014).
[8] Percy Jaiswal, Finding Driving Lane Line live with OpenCV (2014).
[9] Introbotics: How to choose DC Motor for robotic project, (2013). http://ggo.gl/images/LdaKk.
[10] http://commons.wikimedia.org/wiki/File:Hslmodels.svg, (2013).