Design of an IoT Based Autonomous Vehicle with the Aid of Computer Vision

Mohammad Rubaiyat Tanvir Hossain*, Md. Asif Shahjalal†, Nowroz Farhan Nur‡
Department of Electrical and Electronic Engineering
Chittagong University of Engineering and Technology
Chittagong-4349, Bangladesh
Email: *mrthossain@cuet.ac.bd, †asif.cuet93@gmail.com, ‡nowroz97@gmail.com

***Abstract*—A web controlled and partially autonomous vehicle system is presented in this paper. It highlights the idea to develop a remote controlled car which can be driven from anywhere using internet over a secured server. This car will also have limited automation features like traffic light detection, obstacle avoidance system and lane detection system so that it can drive itself safely in case of connectivity failure. The main goal here is to minimize the risk of human life and ensure highest safety during driving. At the same time the car will assure comfort and convenience to the controller. A miniature car including the above features has been developed which showed optimum performance in a simulated environment. The system mainly consists of a Raspberry Pi, an Arduino, a Picamera, a sonar module, a web interface and internet modem. The Raspberry Pi was mainly used for the Computer Vision algorithms and for streaming video through internet. The proposed system is very cheap and very efficient in terms of automation.**

***Index Terms*—Internet of Things(IoT), Raspberry PI, Arduino Uno, Computer Vision, OpenCV, Sonar Module, Picamera, Machine Learning, Python, Apache Web Server.**

I. Introduction

With the ever-growing technological advancement, human civilization is looking for automation in every sphere of life. Automated car is one of the latest trends which has been massively recognized by people all around the world as they want maximum security and comfort during driving. Nowadays, road accident is one of the prime concerns for the people. It became very frequent and uncertain. Most of the road accidents occur due to lack of abidance of the traffic rules. Most of the time, the drivers become drowsy or distracted during driving and eventually hit objects ahead of them. If the driving process can be handled with the aid of Computer Vision and efficient sensors then the risk of human mistakes can be highly reduced. Besides, sometimes it gets necessary to access the car from a remote location in order to reduce hassles. In this case, it would be a lot more convenient if the car could be viewed from a remote computer and driven by interaction through the computer keyboard. This could be as easy as playing a computer game. Our work is based on Internet of Things technology and Computer Vision to remotely control our vehicle and automation features.

Since 1920 the research for vehicle automation has been conducted on, although first promising trials took place around 1950s. During 1980 with Carnegie Mellon University's Navlab and ALV [1], [2] the first ever autonomous car has been seen. This has paved the way for the companies to work on autonomous vehicle research. In July 2013, Vislab demonstrated BRAIVE a vehicle that moved autonomously on a mixed traffic route. Cities like Belgium, France, Italy and the UK are planning to operate transport systems for driver less cars. Germany, Netherlands and Spain have allowed testing robotic cars in traffic.

Google self-driving car is a recent trend. Sebastian Thrun, professor of Stanford University and his team have developed the algorithm and led the development of the Google self-driving car [3]. Google self-driving cars are designed to navigate safely through city streets. They have sensors designed to detect objects as far as two football fields away in all directions, including human and vehicles. It will be marketed by 2020. Tesla motors will be fully driver less within two years and will compete with the Google car. So far these prototypes also have a live driver inside them to acquire test data. If a live driver can be substituted with an online driver then the risk of human life damage can be avoided.

Society of Automotive Engineers (SAE) has classified automated vehicles into six categories from level-0 (No automation) to level-5 (Full automation). In this paper we have worked with the level-3 (Conditional automation). In this level the driver can safely turn their attention in a familiar place and good weather condition. This is by far the most secure driving system as we can not put confidence into fully automated vehicles yet. Besides the cost behind Google car or Tesla wheels is supposed to be out of reach for most people.

The driving becomes a lot boring during traffic jam. In this situation traffic light detection system and obstacle detection comes in handy. Researches have been done regarding traffic light detection using heuristic models and color segmentation [4], [5]. However in this paper a Haar Cascade Classifier is developed with respect to the working environment which can easily be extended to work in real life environment by collecting a lot more frames from the environment and by using powerful computer.

Various lane detection techniques have been observed. Lane detection techniques using OpenCV based on Receiver Operating Characteristic curve and Detection Error Trade-off curve [6] and using perspective image [7] have already been worked on. In this paper lane detection is done using canny edge algorithm and Hough line transformation which has shown good rate of success in the working condition. So far many related works are done involving remote controlling an autonomous car using Bluetooth with android or iPhone. Several papers [8], [9] have been published regarding autonomous car and obstacle avoidance system which lack either versatile control over internet or live video streaming. Concepts from papers [10], [11], [12] like home surveillance system, automatic toll collection, obstacle avoidance system are combined to further develop the idea. A sample car was built for the purpose of testing in a created environment. This car successfully achieved its goal.

II. Proposed Method

The overall work can be divided into two major categories. The methodology is briefly shown in Fig. 1.

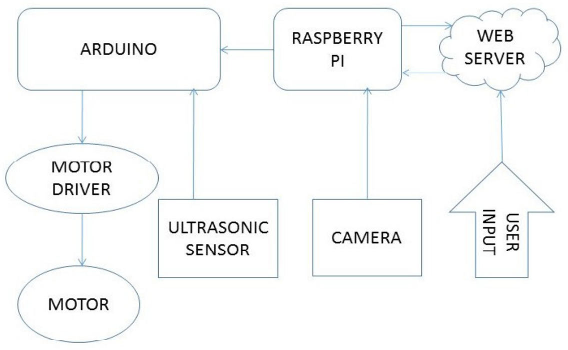

Fig. 1. Working procedure

Firstly the car can be remotely controlled through Internet using a web browser. In case of connectivity failure it can act autonomously in a good weather condition. The proposal consists of complex Computer Vision algorithms and video transmission with Internet. Raspberry pi and Arduino are the main devices to implement the prototype. The Raspberry Pi streams the video to internet. A user can access the streaming using a web browser. It takes a lot of processing power for simultaneously working on video streaming and running Computer Vision. The Raspberry Pi 2 model B is a single-board computer with a powerful processing unit and serial and camera interface (CSI). The Raspberry Pi camera module can be used to take high-definition video. It can be accessed through the V4L (Video for Linux) APIs, and there are numerous third-party libraries built for it, including the Picamera Python library which will be beneficial to the live streaming purpose. Apache is a popular web server application that was installed on the Raspberry Pi to allow it to serve web pages. Apache can serve HTML files over HTTP, and with additional modules can serve dynamic web pages using scripting languages such as python. A web page was hosted that shows the video streaming sent from the Picamera. To access into the web page one only need to know the IP address of the Raspberry Pi and a user name and password to log in. From the web page the car can be fully driven.

For the connectivity failure the car needs to work on its own. It needs to keep itself safe from collision and abide by the traffic rules. The Arduino controls the motor driver circuit. It is connected with sonar, an ultrasonic sensor which evaluates the attributes of a target by interpreting the echoes from radio waves. It is used to detect the distance of obstacles from the car. If an obstacle is detected then the Arduino stops the motor from running operation. Meanwhile the Raspberry Pi uses computer vision algorithms to detect the lane and traffic light signals. Python Open Source Computer Vision (OpenCV) is a library of programming functions mainly aimed at real-time computer vision. It has over 2500 optimized algorithms which can be used for image processing, detection, object identification, classification of actions, traces and other functions. The Raspberry Pi is interfaced with the Arduino with serial communication. It controls the Arduino to run the car accordingly.

A. *Streaming Video and Remote Access*

The Raspberry Pi works on Linux operating system. It can host web pages through Apache server. It responds to requests to serve up web pages, which can be simple HTML or sophisticated web-based apps. To make the Pi capable of hosting websites Apache is installed on it. Apache is a free, open-source HTTP (Hypertext Transfer Protocol) web server application. A website was built for hosting the streamed video and for controlling the car remotely.

MJPG streamer was used to stream video from Raspberry Pi. The easiest way to install it is by using subversion. There is a facility in linux operating system named daemon which runs the selected programs automatically during system boot up. The scripts for MJPG streamer,traffic light detector and lane detector are all run through daemon. So whenever the Raspberry Pi is powered up it automatically keeps streaming the video from its camera to its web server. Now if we type ̈http://(Raspberry Pi's IP address):port number ̈then the streaming data can be viewed from any web browser.

The web page hosting the video streaming was developed using python flask framework. From the web page, a python script is used to handle keyboard interruption from the user. This keyboard interruption can be processed and sent through the internet to the remote Raspberry Pi which is located inside the car. The Pi in turn sends signal to Arduino to control the motor through serial communication.

B. *Obstacle Avoidance*

A sonar sensor (HC-SR04) has been used for this purpose. It emits very high frequency (40 KHz) of sound. It has two transducers - a transmitter and a receiver. The "Transmit" transducer sends out a short burst of (8 cycles) of pulse train. The sonar module timing diagram is shown in Fig. 2[13].

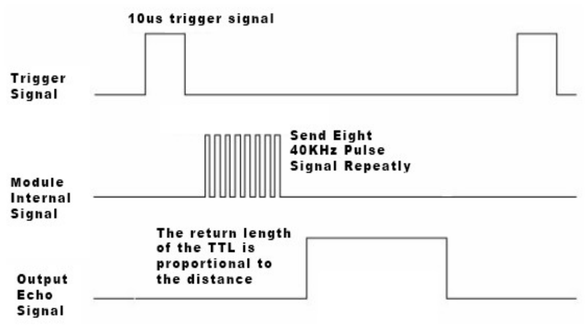
10us trigger signal
Trigger Signal
Send Eight 40KHz Pulse Signal Repeatly
Module Internal Signal
The return length of the TTL is proportional to the distance
Output Echo Signal

Fig. 2. Timing diagram of a sonar module

The "Receive" transducer in turns wait for an echo. If an object exists on its perimeter an echo bounces back to the "Receive" transducer. Distance of the object is calculated from the following equation.

$$d = v * (t/2) \tag{1}$$

Here d is the distance from object, t is the total time from transmission to reception and v is the velocity of sound which is typically 340 m/s in room temperature.

The sonar is connected with an Arduino which calculates the distance and controls the motor rotation accordingly. The sonar is placed in front of the vehicle and is mounted on a servo motor which can rotate upto 180 degrees. This way if an obstacle comes ahead, then it rotates the sonar and check if the road is clear around. If no obstacle is found then it turns the car and picks an alternative way.

C. Traffic Light Detection

The traffic light detection procedure can be briefly described with Fig. 3.

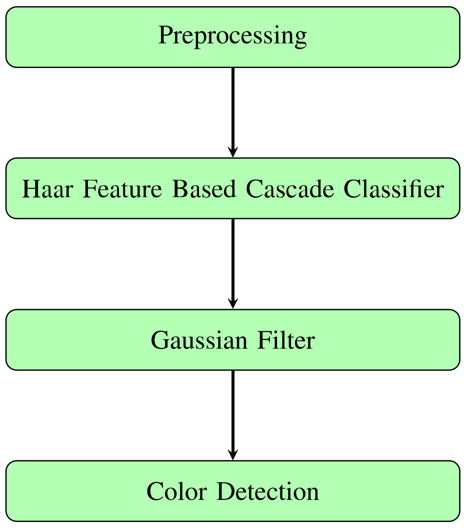

Fig. 3. Traffic light detection process

1) Preprocessing: The image frames captured from the video were converted into gray scale images.

2) Haar Feature-Based Cascade Classifier: Haar feature-based cascade classifier was chosen for traffic light detection. This feature was highly popular for its success on face detection. It has two parts-training and detection. We can generate both using OpenCV. To build a Haar Cascade it was needed to generate some positive images and some negative images. The result is better as much as sample image is generated. However for the processing power limitation of Raspberry Pi only 1000 positive samples and 1000 negative samples were taken.

The positive samples were the images of different traffic light signals in different angle. The negative samples were collected from the related environment where no traffic signals were present. With the utilization of OpenCV_createsamples command many other positive samples were randomly generated superimposing on the negatives. A vector file was created merging all the positive samples. This training part was done using OpenCV_traincascade command. This cascade classifier is used to detect the traffic light post that is the region of interest.

3) Gaussian Filter: To reduce the image noises gaussian blur filter was used.

4) Color Detection: The BGR image was converted to HSV, because it gets much easier to represent the colors in HSV. Then the threshold value for green and red colors were selected individually for the image. And finally the green or red part was extracted.

D. Lane Detection

For the lane detection technique the popular canny edge detection and Hough line transform was used. This algorithm is highly efficient for a road with clearly visible lane marker. In edge detection algorithm the boundaries of an image are generally detected. Canny algorithm is selected for its very low error rate, good localization and minimal response that is only one detector response per edge. For good efficiency several steps are needed to be maintained. Figure. 4. shows the process in a nutshell.

1) Gaussian Filter: Suitable masking was done to filter out the noise from the original image using gaussian filter.

2) Finding Intensity Gradient: After smoothing was done Sobel kernel was used in both horizontal and vertical direction to get the first derivative in both directions. Gradient is the change of brightness in a series of pixels.

3) Removing non Edge: Pixels that were not part of an edge were removed. Thus an image with thin edge is observed.

4) Hysteresis: Canny uses two thresholds. If a pixel gradient is higher than the upper threshold, it is accepted as an edge. Otherwise it is rejected. If a pixel gradient is between the thresholds then it is only accepted if it is connected with a pixel of upper threshold.

5) Hough Line Transformation: After canny edge detection Hough line transformation is applied. Hough transformation is very efficient for detecting any shape if it can be mathematically expressed even if it is a little distorted. To determine the Hough line two parameters are needed- ρ, the perpendicular

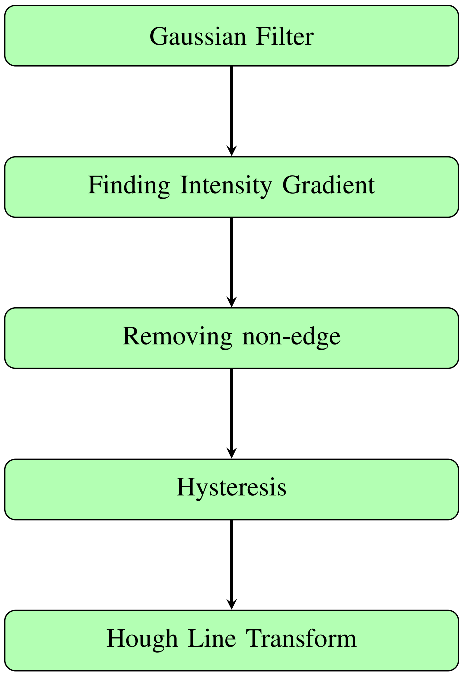

Fig. 4. Lane detection process

distance from origin to the line and θ, the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise. These can form the parametric line equation.

$$\rho = x cos\theta + y sin\theta \tag{2}$$

There exists an OpenCV function for doing this named cv2.HoughLines(). This function takes the ρ,θ as argument. It also takes an extra argument which determines the threshold for allowing a pixel as a line. From the perspective of Bangladesh the drivers always drive through the left side of the road. So it needs to detect the left lane marker.

III. Result

A miniature car including the above features has been developed which showed optimum performance in a simulated environment. The sonar sensor is set up in front of the car. The camera is fixed on the window of the car and the processing units are set within. Fig. 5. shows the implemented model.

Whenever an obstacle was placed in front of the car it reduced its speed and stopped. Some echoes were overlapped and gave back garbage values for a very few time. For better performance multiple sonars can be used. For real life application much efficient and powerful sensor can certainly minimize the hassle.

Python Flask framework was used to host the video sent from the Raspberry Pi. A web page was developed to show the video stream and to provide user interface for supporting the remote control from web page. The MJPG streamer streamed data flawlessly except for that it suffered ms of delay. The video data streamed was 100 ms slow and the commands

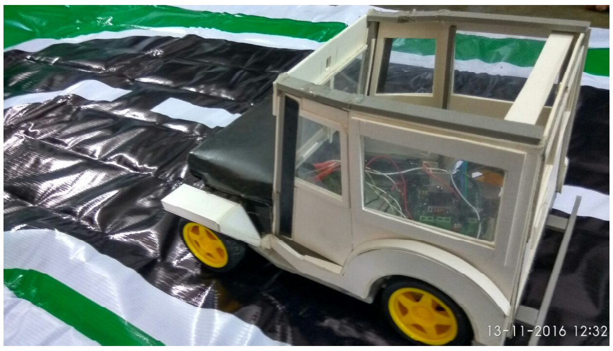

Fig. 5. Implemented model

sent from web page consumed another 200 ms. So it overall suffered through 300 milliseconds of delay.

The traffic light detection system was very accurate for the given environment. Although to make it work in the real life environment a lot more training data will be needed. Here 1000 positive and 1000 negative samples have been used to create the classifier. The car could successfully detect the location and interpretation of traffic signal. Fig. 6 shows the traffic light detection obtained from our implemented model.

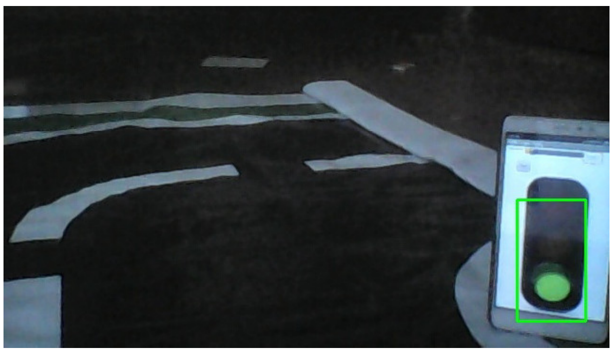

Fig. 6. Traffic light detection using Haar cascade classifier

The lane detection algorithm worked flawlessly and was able to detect its lane without much of an error. It successfully drove itself in a printed road. Fig. 7, Fig. 8, Fig. 9 show the results of the image processing for detecting lane.

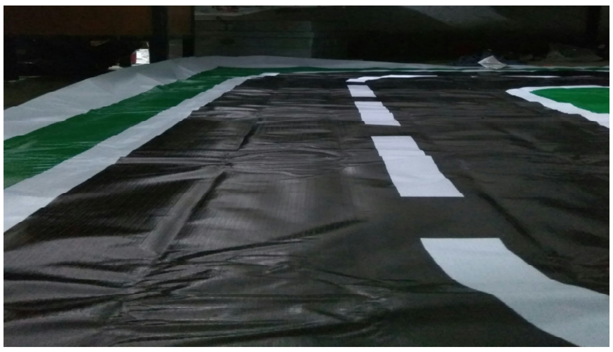

Fig. 7. Original image

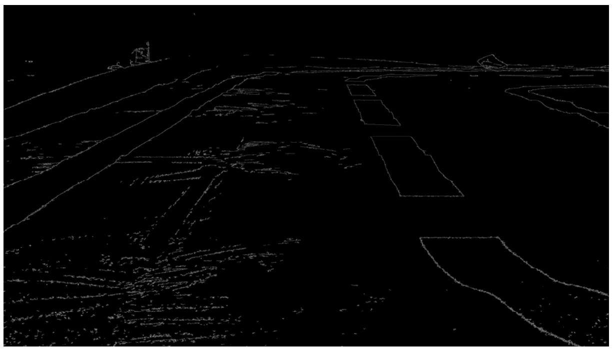

Fig. 8. After canny edge detection

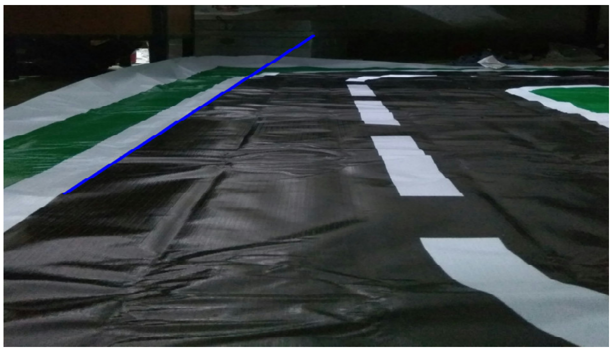

Fig. 9. After Hough transformation and drawing a Hough line

IV. Conclusion

In this paper a method to implement some automation feature in a regular car is described. Utilizing this a small prototype is designed and built. This prototype successfully achieved the goals. However Raspberry Pi maybe powerful but yet we need a much powerful computing machine if we need to implement it on a real car. Or multiple Raspberry Pi could be cascaded to perform different tasks.

As for the cascade classifier we have created for the traffic light detection we need a lot more positive and negative samples from different streets and different weather and light condition if we want to implement it on real life. As the whole system relies heavily on Internet so it would be very convenient in a region where 4G data is available. A fast Internet connection is one of the limitations of this project.

The implemented model is a Level-3 automated car. However if it is to be provided with Level-5 automation a lot more work is to be done. It can not navigate its way to a given location. So a navigation system can be built on top of it. A much better classifier for traffic light detection can be designed to get better performance in a real life scenario.

References

[1] T. Kanade, C. Thorpe, and W. Whittaker, "Autonomous land vehicle project at cmu," in *Proceedings of the 1986 ACM fourteenth annual conference on Computer science*. ACM, 1986, pp. 71–80.

[2] R. Wallace, A. Stentz, C. E. Thorpe, H. Maravec, W. Whittaker, and T. Kanade, "First results in robot road-following." in *IJCAI*. Citeseer, 1985, pp. 1089–1095.

[3] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 163–168.

[4] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles." in *Intelligent Vehicles Symposium*, 2010, pp. 431–435.

[5] M. P. Philipsen, M. B. Jensen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Traffic light detection: A learning algorithm and evaluations on challenging dataset," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2341–2345.

[6] S. K. Vishwakarma, D. S. Yadav *et al.*, "Analysis of lane detection techniques using opencv," in *2015 Annual IEEE India Conference (INDICON)*. IEEE, 2015, pp. 1–4.

[7] M. P. Batista, P. Y. Shinzato, D. F. Wolf, and D. Gomes, "Lane detection and estimation using perspective image," in *Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014 Joint Conference on*. IEEE, 2014, pp. 25–30.

[8] N. Ollukaren and K. McFall, "Low-cost platform for autonomous ground vehicle research," in *Proceedings of the 14th Early Career Technical Conference*, vol. 13, 2014.

[9] J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 213–218, 1988.

[10] D. S. K. Dixit and M. S. Dhayagonde, "Design and implementation of e-surveillance robot for video monitoring and living body detection," *International Journal of Scientific and Research Publications*, vol. 4, no. 4, pp. 2250–3153, 2014.

[11] A. Suryatali and V. Dharmadhikari, "Computer vision based vehicle detection for toll collection system using embedded linux," in *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*. IEEE, 2015, pp. 1–7.

[12] P. B. Rao and S. Uma, "Raspberry pi home automation with wireless sensors using smart phone," *Int. J. Comput. Sci. Mob. Comput*, vol. 4, pp. 797–803, 2015.

[13] E. Freaks, "Ultrasonic ranging module hc-sr04," *linea]. Disponible: http://www. micropik. com/PDF/HCSR04. pdf. Accedido*, 2011.