VOL. 19, NO. 7, APRIL 2024 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

REAL-TIME LANE DETECTION USING RASPBERRY PI FOR AN AUTONOMOUS VEHICLE

Umamaheswari Ramisetty[1], M. Grace Mercy[2], V. Nooka Raju[2], N. Jagadesh Babu[2], P. Ashok Kumar[3] and Vempalle Rafi[4]
[1]Department of ECM, Vignan's Institute of Information Technology, Visakhapatnam, India
[2]Department of EECE, GST, GITAM, Visakhapatnam, India
[3]Department of ECE, Vignan's Institute of Engineering for Women, Visakhapatnam, India
[4]Department of EEE, JNTUA College of Engineering, Pulivendula, India
E-Mail: vempallerafi@gmail.com

ABSTRACT

The latest developments include the smart world, smart cars, and other technologies. The development of intelligent vehicles must be able to detect and identify traffic signs to ensure traffic safety. To control the speed of an autonomous vehicle, environmental perception is essential. The traffic regulations listed on traffic signs must be fed as input to autonomous vehicles. However, traffic regulation is one of the essential factors in autonomous vehicles, but many more factors need to be taken into consideration. In this paper, machine learning techniques for stop sign detection, traffic sign detection, and object detection with Obstacle avoidance and distance calculation play a crucial role in regulating the longitudinal velocity of an autonomous vehicle. The stop sign disappears from the camera's field of view as the car approaches it, making it challenging to stop the car at the desired distance from the sign. To know exactly where to stop the vehicle, knowledge of the location of the stop line is crucial. Obstacle avoidance and detection of the object are other challenging factors that analyse the potential. The Haar cascade classifier method is the optimizer approach utilised here. Features of both the Hue Saturation Value grey scaling space have faster detection capability in speed and low lighting suffering. The proposed technique is evaluated using an Indian Traffic Sign that has set a benchmark. The proposed method provides an accuracy of almost 80%.

Keywords: lane tracking object and sign identification, machine learning, image processing, Haar cascade, control of self-driving autos.

Manuscript Received 19 February 2024; Revised 24 April 2024; Published 30 May 2024

1. INTRODUCTION

The rapid advancement of hardware and software products mentioned by Bresson [3] has led to a surge in research into autonomous cars in recent years. It is a major step forward in the development of safe autonomous transportation. The biggest technology companies, including Google, Uber, Tesla, Samsung, and others, are all interested in this subject. Under certain circumstances, autonomous navigation systems, particularly when using image processing methods, perform well. The processing of driving situations can be flexible, focused on the real target, and independent of conditions like lighting and background. This has been proven by several recent developments in machine learning. The most significant issue is that identical traffic signs come in various sizes.

When training a Convolutional Neural Network, ZhitongXiong *et al* [1] suggested using different feature sizes and training images, which leads to problems with detection and recognition. Problems arise when dealing with enormous datasets because the training data grows in direct proportion to the data type. Although the accuracy rate was great, the processing time was also very high when Yuan *et al*. [2] used Adaboost with SVM for road traffic sign recognition. To speed up the processing, we employ the Haar cascade classifier. Algorithms for vehicle tracking and lane recognition using cameras are crucial to many autonomous systems. The navigational processes of these systems mostly focus on the outcomes of detecting algorithms. However, lane detection techniques necessitate additional computational power and time spent on pre-processing. Environmental factors also affect them, and they need constant modification. Since traffic signs are relatively small items in comparison to the image size, they can be difficult to identify from other, less obvious things in complicated street scenes that lack contextual information. To address this issue, Qi Wang et al. [4] provide an alternative method. They suggest a model that uses a convolutional encoder and a recurrent neural network stacked one on top of the other. With its built-in attention mechanism, the recurrent neural network can improve its detection performance by actively focusing on specific local regions. A distributed filtering approach was also suggested by Gao *et al*. [5] to reduce the impact of partial and noisy observations in vehicle data. In a nutshell, deep learning has shown state-of-the-art, human-competitive, and occasionally better-than-human performance in solving numerous computer vision problems, including object detection, image classification (as shown by Linyu [8]), retrieval, and semantic segmentation (discussed by Long [9]), according to Wu Libing [6] and Qian [7]. The MATLAB Automated Driving Toolbox is one example of a virtual testing environment for open- and closed-loop simulations; Kang [10] summarizes pertinent datasets for testing DL-based AD systems. Virtual testing is a significant area, particularly for deep learning algorithm development (e.g., convolutional neural networks for perception tasks). To address DL-specific deficiencies, such as robustness limits

and black-box features, Jie M. *et al*. [11] summarize research on acceptable testing methodologies and highlight these discrepancies. A gradual framework for traffic sign identification, tracking, and recognition from driving films is proposed by Hang Wan-Nan *et al.* [12]. The authors increase the overall identification performance by separating the tracking algorithm into two parts: motion and appearance models. Then, they use a scale-based weighted classifier that gives larger votes to signs observed on a wide scale to complete the pipeline.

The design and development of autonomous vehicles, their social and economic impacts, and the legal and regulatory frameworks needed to govern their use are among the many aspects of the technology that researchers are interested in, according to a literature survey on autonomous vehicles. Even though numerous studies have shown effective methods for traffic sign recognition, these algorithms do have certain limitations. For example, they aren't very adaptable to different environments and can't handle disturbances such as inclement weather, fading, or changes in lighting. As a result, the recognition rate will decrease.

In this research, we present the development of an autonomous system that may be controlled by a pre-programmed, integrated framework using visual input acquired from its environment. Machine learning techniques or rule-based control could be used to train this type of control system. Since the latter would make the system more adaptable and provide a more robust decision-making mechanism across a variety of scenarios, we opted for it. The suggested system's primary goals are lane tracking and following a target vehicle.

2. RELATED WORK

One of the basic requirements for autonomous car systems is algorithms that can recognize lanes using cameras. The identification method can be time-consuming and computationally intensive because of the extensive pre-processing that is necessary for these algorithms to effectively recognize lane markers on typical roadways. In their Deep Lanes investigation, the researchers utilized a deep learning algorithm to recognize lane markings. Frames are taken by mounting two downward-facing cameras on either side of the vehicle. A deep neural network is trained to analyse captured frames and determine whether the lane is present.

Additionally, the model output is the estimated lane position. On the other hand, it's crucial to find other cars and roadside objects to keep these autonomous systems safe. There are numerous methods for completing this task, but each method has trade-offs. Huang considered three detectors known as "meta-architectures": Faster R-CNN, R-FCN, and SSD, which stands for Single Shot Multi Box Detector. Using different convolutional neural network (CNN) models such as VGGNet, ResNet, and Inception, we compare the detection models based on picture resolution and other parameters. Both the feature extractors' and the comparison models' mean average precision are shown. We also propose an alternative approach to vehicle detection that makes use of artificial neural networks (ANNs) trained using properties similar to those of Haar networks to identify the vehicle. These traits are extracted from both positive and negative photos of cars. Cars can vary in size and features, yet they all share certain characteristics if each car has its own unique set of characteristics. To train ANNs, these features are first turned into features that are similar to Haar features.

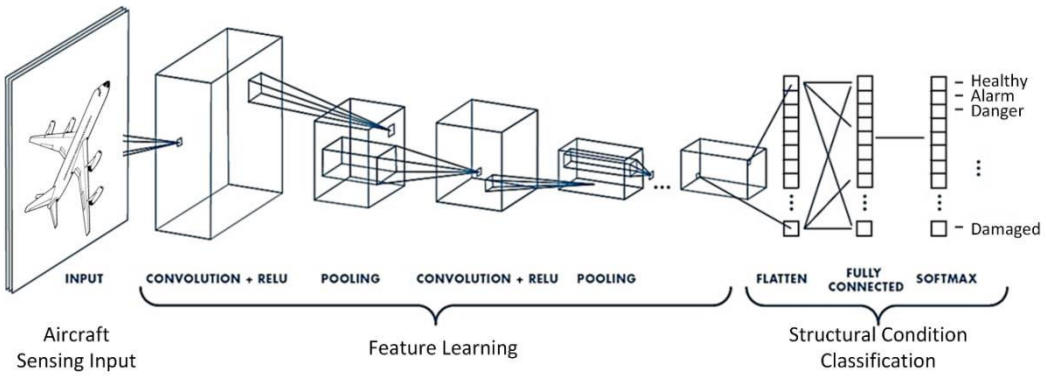

Figure-1. CNN Architecture.

Methods for Haar Features One machine learning approach that has been used to recognize objects based on their edge properties is the cascade classifier. Neural networks can improve detection performance but slow down execution speed, according to researchers. The detection rate is good for the Haar-like feature, but the false positive rate is considerable. Deep learning is a machine learning technique that uses examples from the training process to teach computers new tasks. You can learn in an unsupervised, semi-supervised, or supervised fashion. An MLP is a perceptron with three or more layers. To classify data that cannot be linearly separated, a nonlinear activation function is assessed. The network was completed by connecting the nodes of each layer to those of the layer below it. Convolutional neural networks (CNNs) typically consist of dense layers followed by one or more layers of convolution. As illustrated in Figure-1, the primary objective of convolutional neural network (CNN) design is to exploit the two-dimensional geometry of an input, usually an image or audio file. In a

convolutional neural network (CNN), fully connected layers usually come after subsampling and convolutional layers.

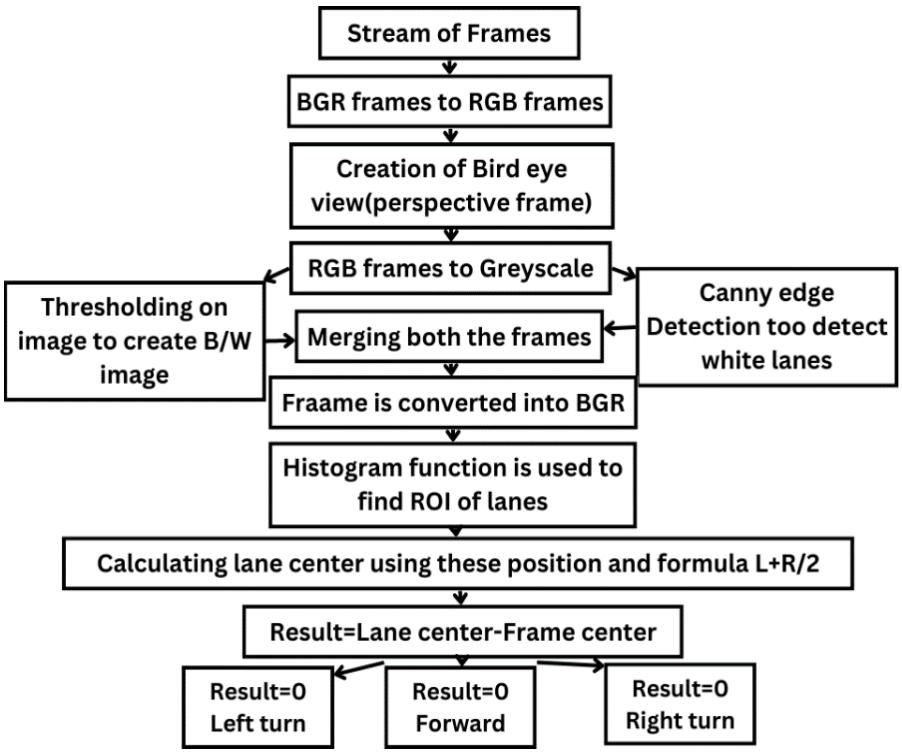

Figure-2. Flow chart for image processing and lane detection.

3. IMPLEMENTATION

3.1 Lane Detection Methods

To determine the lane detection ROI's distance from the left side, we first processed the image and then measured the distance of the white pixels (255). To do this, their locations are determined by processing the pixels like vectors. Here, the histogram comes into play. The Arduino UNO is instructed to rotate in the correct direction if the distance between white pixels increases. As soon as the distance decreases, the Arduino UNO ought to turn left. The required degree of rotation is adjusted about the measured distance. The developed model was tested in simulation before being tested in the real world. Figure-2 shows the testing of an autonomous vehicle in various settings. The necessary RCS and ACS parameters were established before we began the sending and listening processes. The Raspberry Pi 3 transmits frames to the computer that were captured by the camera module. The computer utilizes these frames and pre-processes them so it can fulfill the input requirements of the model and make predictions using the model. The two projected values that the steering and speed models provide are identical. The features derived from the input picture affect the tracking and velocity prediction. On a straight road, speed rises; however, on turns to the right or left, speed falls.

3.1.1 Region of interest

The developed model was tested in simulation before being tested in the real world. Figure-3 shows the testing of an autonomous vehicle in various settings. The necessary RCS and ACS parameters were established before we began the sending and listening processes. By Figure-3 Raspberry Pi 3 transmits frames to the computer that were captured by the camera module. To meet the demands of the model input and be able to predict using the model, the computer uses these frames and pre-processes them. Models for steering and speed send two identical prediction values. Steering prediction and speed prediction are affected by the features that are extracted from the input image.

Region of Interest = Rect (x, y,x+a,y+b) (1)
X = x axis coordinates point
Y = y axis coordinates point
X + A = width of the ROI
Y + B = Height to ROI

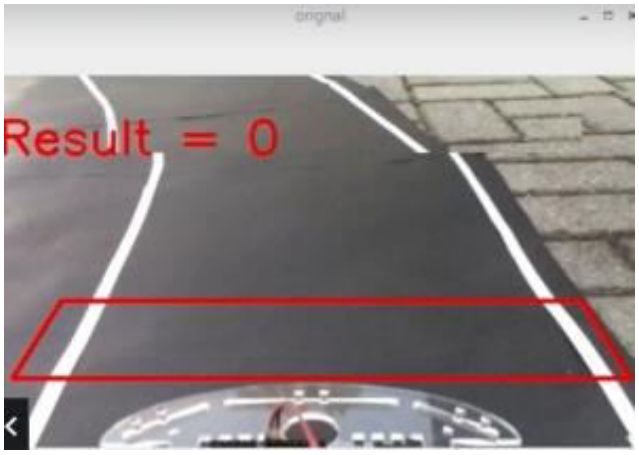

Figure-3. Testing vehicle.

3.1.2 Perspective view

ROI is derived from Figure-4 to define a perceptive warp four coordinates are needed to define. A bird's-eye view perspective is a view or perspective of an object or location from high above as if seen from the viewpoint of a bird flying in the sky. With this viewpoint, you can see the region's general structure and layout as well as any patterns or connections between various elements.

An aerial perspective is frequently used to better understand intricate systems or procedures, such as urban planning, traffic flow, or landscape design. It can also be used in photography or film to convey a sense of scale and grandeur or to produce eye-catching visual effects.

In general, having a bird's-eye view perspective helps provide a more comprehensive view and understanding of the world. So, this perspective is shown in Figure-4.

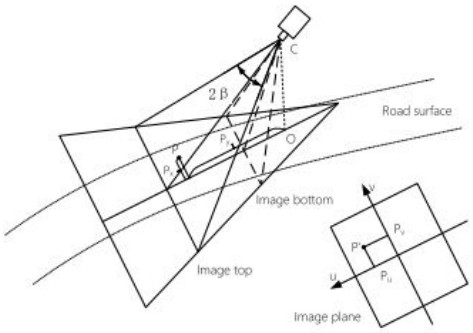

Figure-4. Birds eye perspective.

3.1.3 Lane tacking

Using the Canny edge identification algorithm, we can eliminate the edge points that aren't good candidates [13]. The Raspberry Pi 3 transfers photos taken by the camera module to a computer. The computer utilizes these frames and performs pre-processing on them so that it can match the requirements of the input parameters and provide predictions using the model. Both the steering and speed models provide the same forecast numbers. Input picture feature extraction has an effect on both steering and speed prediction. The opposite is true for curves: speed increases on straight roads and decreases when navigating corners. The presence of a detection edge and the successful maintenance of a threshold are inferred from Figure-5. The opposite is true for curves: speed increases on straight roads and decreases when navigating corners.

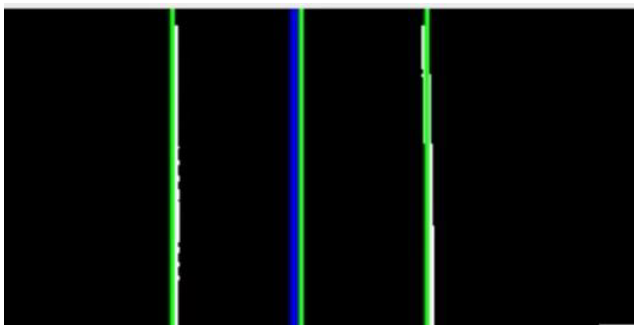

Figure-5. Lane detection.

$$\text{Lane centre} = \frac{\text{Right Lane Pos} - \text{Left lane pos}}{2} + \text{Left Lane pos} \quad (2)$$

3.2 Object Detection Methods

When doing object detection, machine learning is involved. We start by collecting samples of the target object. The positive samples are what we refer to as. Afterward, we snap several pictures without the thing we're trying to detect. The negative samples are those. We used Cascade Classifier software for both training and detection. After that, the code that detects the target object uses various open CV techniques to identify it. Afterward, we determined the camera-to-object distance by plugging the measured distance and the object's detected pixels into the distance formula (y=mx+c). We later added a second method to use the distance as a parameter to instruct an Arduino UNO board, which contains an algorithm for stopping the vehicle within a given range.

The Faster Haar classifier was used to train this model. We required more precise results, so we went with the center because the difference between the left and right could be small. Classifier using the RESNET Inception v2 feature extractor would provide better accuracy, as mentioned in the related works section since there is no issue with processing power. Within three hours, the model has learned nine thousand steps to identify vehicles. Object detection and object distance calculation are shown in Figures 6 & 7.

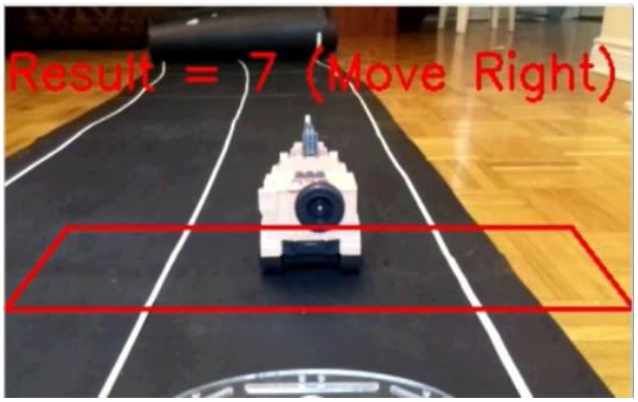

Figure-6. Object detection.

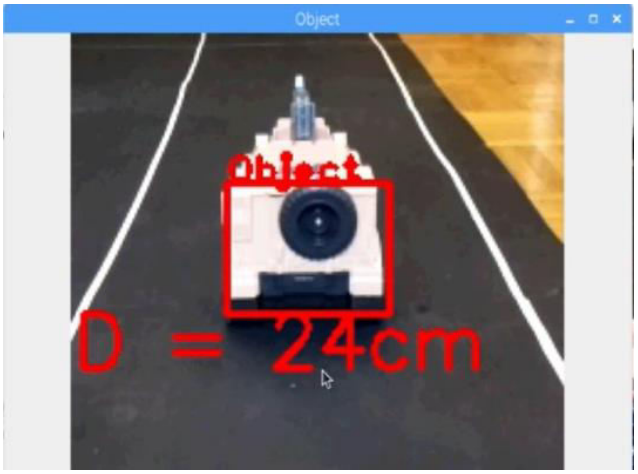

Figure-7. Object distance calculation.

The method we suggested for labeling vehicles was insufficient and only effective under certain circumstances. Since the model does not know where it is about the x-axis, it gives erroneous results if the leading RC car is tilted to the right or left of the image. For instance, if the front car is tilted to the right in the camera's field of view and the camera sees the left side of the front car, it may not be the case that the front car is moving to the left. Therefore, to make a precise move, positional arguments are required. Thresholds are required to make precise movements.

3.3 Sign Detection Method

In this paper, we used shape-based classifiers to detect stop signs and traffic lights, along with cascade classifiers based on Haar features. A cascade classifier detector and trainer are both available in the OpenCV library. A trainer needs both positive and negative samples to properly train a Haar feature-based cascade classifier. Cropped images of the target object are used as positive samples. Negative samples are any additional images in which the intended subject is obscured. Images of both positive and negative samples are shown in Figure-8.

Figure-8. Sign detection.

The traffic light's location is first ascertained by processing the grayscale image using the learned Haar cascade classifier. After this identification, the ROI is used to draw the bounding box. The second step is to lower the ROI's noise level using a Gaussian Filter. The third step is to check the ROI to find the brightest point. When we saw the stop sign or traffic light, we measured how far away the camera was from the object. The camera is fine-tuned using the photos and the associated distance measurements. To find the optimal focal length, we averaged all the focus length measurements taken from the supplied calibration photos.

To identify stop signs and traffic lights, we employed a shape-based technique in conjunction with cascade classifiers trained on Haar features. The OpenCV package provides several useful tools, including a trainer and a cascade classifier detector. Training a Haar feature-based cascade classifier requires both positive and negative samples to be provided to the trainer. Images that have been selectively cropped to highlight the target object are called positive samples. Any supplementary pictures that don't depict the target are considered negative samples. Figure-9 shows positive samples and Figure-10 shows negative samples.

Figure-9. Positive samples.

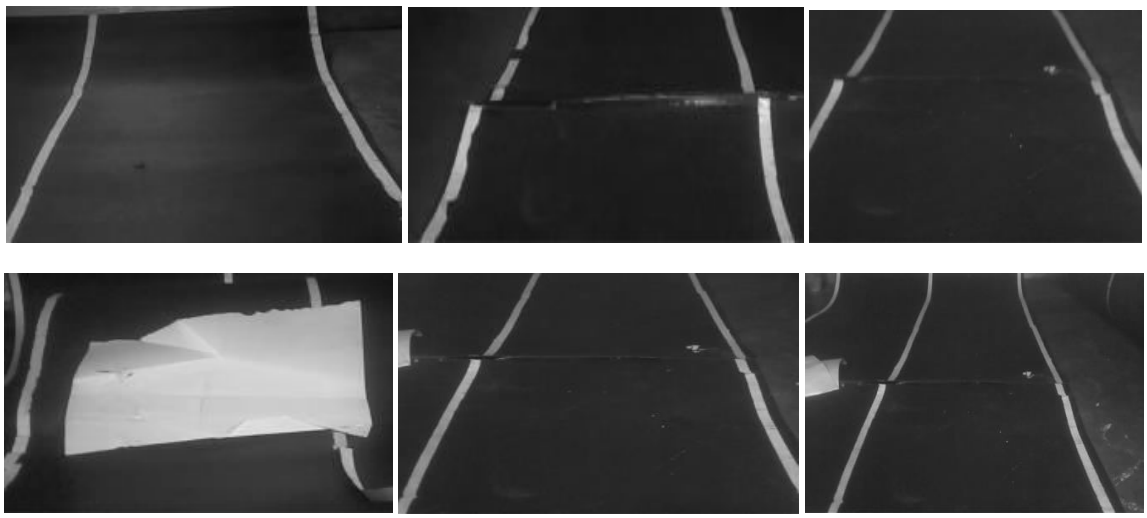

Figure-10. Negative samples.

Table-1. Training samples.

Type	Positive samples	Negative samples	Sample size (in pixels)
Stop Sign	30	300	30x30
Object detection	40	300	40x42

Training samples including positive and negative frames are displayed in Table-1.

4. RESULTS AND DISCUSSIONS

Three mechanisms are being used by the autonomous system: The first mechanism uses a Camera to detect obstacles. The second mechanism is the ability to identify lanes and traffic signals, such as stop signs and lights. A graphic user interface (GUI) controller or self-operating mode for systems. The computer then evaluates the frames based on the chosen model after receiving them from the Raspberry Pi. The autonomous car then moves independently because of the prediction and obstacle conditions after the predicted values have been sent to the Raspberry Pi. The majority of the trials are focused on identifying distinct road types under varied environmental circumstances. To do this, we ran a battery of experiments on a variety of road photos. The needs of the autonomous system have been satisfied through configurations. Hardware was successfully implemented. Both systems were started after ACS and RCS had been set up as described in the Implementation section. According to what was stated in the section on implementation, the autonomous mode has been successful. The camera module can identify traffic signals including stop signs and lights, as well as follow a vehicle's lane. Figures 11 and 12 show the hardware. Stop sign detection is indicated in Figure-13.

Figure-11. Autonomous car.

Figure-12. Autonomous car.

The final output of the project that successfully detected the object and sign boards by the model using the haar cascade classifier. Object detection is indicated in Figure-14.

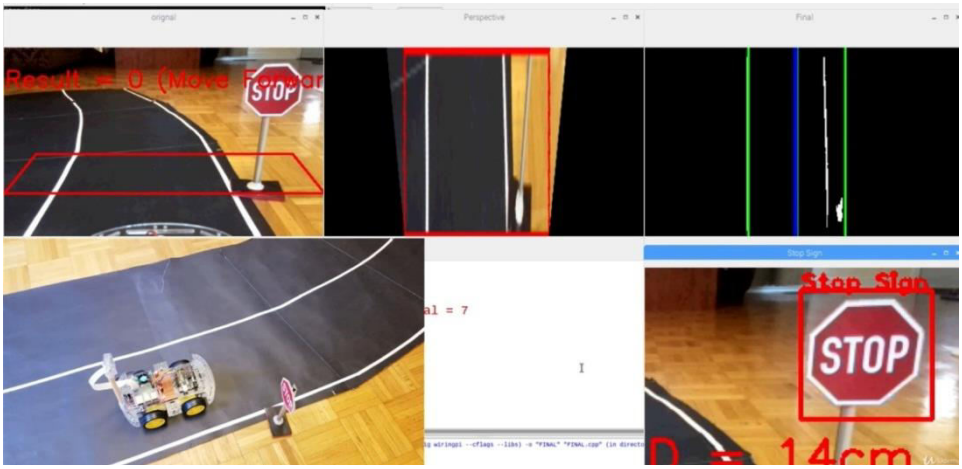

Figure-13. Stop sign detection.

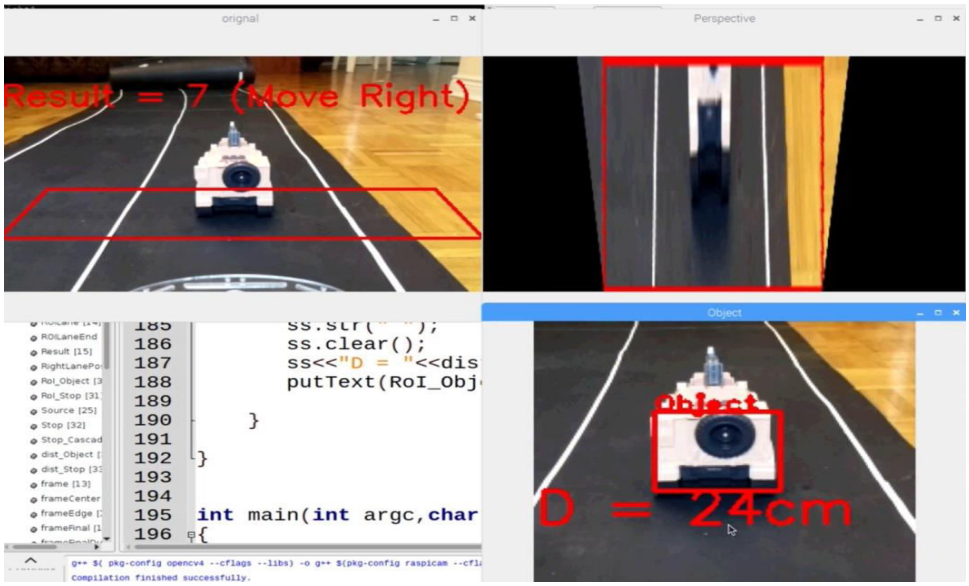

Figure-14. Object detection.

5. CONCLUSIONS

This paper describes a self-driving car system that can travel to a specific location. Additionally, any obstacles in its path are efficiently detected and avoided by the system. The suggested system has a generic design that can be installed on any vehicle, regardless of size. This more accessible substitute for the current generation of self-driving modules will open the door to a promising future for autonomous vehicles. Using information gathered from sensors and models developed through machine learning, an autonomous vehicle can drive itself. Several scenarios meant to reflect the actual world were used to test the system. After testing on single-lane rails, we found that our approach worked effectively. Because the feature model needs to be trained on a huge dataset that contains many of these features, the system is not entirely independent of the environment and light conditions. Adding more cameras and covering broader angles of the vehicle's front would be necessary if the autonomous car system were to be employed in a real-world self-driving car situation, among other significant challenges. In a subsequent study, the system can be upgraded with the following features: backup cameras, automated parking, safe lane changes, communication between vehicles, advanced Wi-Fi encryption, and support for path planning with image processing and sensors for the autopilot. Unmanned Aerial Vehicles (UAVs) can also benefit from the technology, which allows them to navigate autonomously in both indoor and outdoor settings.

REFERENCES

[1] Yuan, Yuan, Zhitong Xiong and Qi Wang. 2016. An incremental framework for video-based traffic sign detection, tracking, and recognition. IEEE Transactions on Intelligent Transportation Systems. 18(7): 1918-1929.

[2] Huang Zhiyong, Yuanlong Yu, Jason Gu and Huaping Liu. 2016. An efficient method for traffic sign

recognition based on an extreme learning machine. IEEE transactions on cybernetics. 47(4): 920-933.

[3] Bresson Guillaume, Zayed Alsayed Li Yu and Sébastien Glaser. 2017. Simultaneous localization and mapping: A survey of current trends in autonomous driving. IEEE Transactions on Intelligent Vehicles. 2(3): 194-220.

[4] Yuan Zhitong Xiong and Qi Wang. 2019. VSSA-NET: Vertical spatial sequence attention network for traffic sign detection. IEEE transactions on image processing. 28(7): 3423-3434.

[5] Gao Jian and Hamidou Tembine. 2018. Distributed mean-field-type filters for traffic networks. IEEE Transactions on Intelligent Transportation Systems. 20(2): 507-521.

[6] Wu Libing, Jingxiao Yang, Man Zhou, Yanjiao Chen and Qian Wang. 2019. LVID: A multimodal biometrics authentication system on smartphones. IEEE Transactions on Information Forensics and Security. 15: 1572-1585.

[7] Wang Qian, Minxin Du, Xiuying Chen, Yanjiao Chen, Pan Zhou, Xiaofeng Chen and Xinyi Huang. 2018. Privacy-preserving collaborative model learning: The case of word vector training. IEEE Transactions on Knowledge and Data Engineering. 30(12): 2381-2393.

[8] Zhang Jinyu, Yifan Zhang and Mengru Shen. 2019. A distance-driven alliance for a p2p live video system. IEEE Transactions on Multimedia. 22(9): 2409-2419.

[9] Long Jonathan, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431-3440.

[10] Kang Yue, Hang Yin, and Christian Berger. 2019. Test your self-driving algorithm: An overview of publicly available driving datasets and virtual testing environments. IEEE Transactions on Intelligent Vehicles. 4(2): 171-185.

[11] Zhang Jie M., Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes, and horizons. IEEE Transactions on Sofware Engineering. 48(1): 1-36. Yuan, Yuan, ZhitongXiong and Qi Wang. 2016. An

[12] Incremental framework for video-based traffic sign detection, tracking, and recognition. IEEE Transactions on Intelligent Transportation Systems. 18(7): 1918-1929.

[13] Zhang Wannan. 2020. Combination of SIFT and Canny edge detection for registration between SAR and optical images. IEEE Geoscience and Remote Sensing Letters. 19: 1-5.

[14] Vempalle Vempalle Rafi P. K. Dhal, M. Rajesh, D. R. Srinivasan, M. Chandrashekhar, N. Madhava Reddy. 2023. Optimal placement of time-varying distributed generators by using crow search and black widow - Hybrid optimization, Measurement: Sensors. Vol. 30.

[15] Rafi Vempalle & P. K. Dhal. 2020. Loss Minimization by Reconfiguration along with Distributed Generator Placement at Radial Distribution System with Hybrid Optimization Techniques, Technology, and Economics of Smart Grids and Sustainable Energy. Vol. 5.

[16] Rafi, Vempalle and P. K. Dhal. 2020. Maximization savings in distribution networks with the optimal location of type-I distributed generator along with reconfiguration using PSO-DA optimization techniques. Materials Today: Proceedings. 33: 4094-4100.

[17] Ramisetty, Uma Maheswari, and Sumanth Kumar Chennupati. "Optimization of number of base station antennas in downlink massive MIMO and analysis of imperfect channel state information by perfection factor." Engineering Science and Technology, an International Journal 23, no. 4 (2020): 851-858.

[18] Ramisetty, Uma Maheswari, and Sumanth Kumar Chennupati. "Performance Analysis of Multi User Mimo System with Successive Hybrid Information and Energy Transfer Beamformer." Wireless Personal Communications (2021): 1-19.