ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/351247579

Autonomous Bot Using Machine Learning and Computer Vision

Article *in* SN Computer Science · July 2021

DOI: 10.1007/s42979-021-00640-6

CITATIONS
6

READS
283

2 authors, including:

Thejas Karkera

1 PUBLICATION **6** CITATIONS

SEE PROFILE

All content following this page was uploaded by Chandra Singh on 27 February 2022.

The user has requested enhancement of the downloaded file.

SN Computer Science (2021) 2:251
https://doi.org/10.1007/s42979-021-00640-6

ORIGINAL RESEARCH

Autonomous Bot Using Machine Learning and Computer Vision

Thejas Karkera[1] · **Chandra Singh**[1]

Received: 19 December 2020 / Accepted: 8 April 2021
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd 2021

Abstract
Self-driving vehicles have the potential to revolutionize urban mobility by providing sustainable, safe, and convenient transportability. In recent years several companies have identified automation as their major area of research and also are investing a huge amount of their financial resources in automating vehicles. This is the period of time where autonomous vehicles are very close to being capable of transporting us to destinations without the aid of drivers in the very near future. In the current generation, the main focus is to make vehicles more automated to provide a better driving experience. These vehicles are designed to drive without or with little human assistance by sensing it's the environment. This can be achieved by a combination of sensors and processing the data with the help of computer vision technology and machine learning. The vehicle autonomy needs to be conducted with care, keeping in mind the challenges that can be faced during the process. Recognizing the traffic signals, understanding the signs, identifying the lane markings are some of the basic functions that it needs to perform. After gathering all this information, the next task is to understand the predefined protocols and follow them without any fault. This problem can be solved stepwise using some functions from image processing and computer vision technology such as Haar transform, perspective mapping, perspective transformation, canny edge detection, and histogram equalization. This solution is further enhanced by including machine learning, which improves performance with experience, making it more reliable. It should be noted that, although the vehicles promoted by the companies ensure 80% reliability, we are not yet ready to completely adapt to the idea of automated vehicles. This paper hence focuses on the negative of current ideology and makes it reliable enough to pave a way for its immediate implementation. In this paper, the authors have used a microcontroller and a microprocessor, to Arduino uno is used as a microcontroller and Raspberry pi B+ model is used as the microprocessor. To detect the lanes the authors have used image processing using a library called OpenCV. For detecting the traffic signs the authors have used supervised machine learning technique, to capture the images authors have used raspberry pi version 2 cam, using cascade training to classify the positive images from the negative images.

Keywords Raspberry Pi · Haar transform · Self-driving vehicles

Introduction

A recent survey says that nearly 1.25 million deaths are caused by road crashes every year, that is an average of 3287 deaths per day. In addition, around 20–50 million people get injured or disabled in accidents. Road accidents are ranked 9th for being the cause of death and it is responsible for 2.2% of the deaths around the globe. Safety has become a major concern these days. Self-driving vehicles could help in reducing this number. This technology could help the individuals who are unable to drive by themselves, such as the elderly, disabled, and the ones who are phobiatic to driving. Currently, several technological firms and universities have started investing immense technical and financial resources in the field of autonomous vehicles because of their high growth scope. This concept of self-driving vehicles is on the edge of becoming the mainstream, provided it overcomes the practical challenges, along with economic, social and legal acceptance. In this project, we have made an attempt to realise this ideology in a very simple and cost-effective way, by

This article is part of the topical collection "Data Science and Communication" guest edited by Kamesh Namudri, Naveen Chilamkurti, Sushma S J and S. Padmashree.

✉ Thejas Karkera
thejkrk7@gmail.com

Chandra Singh
chandrasingh146@gmail.com

[1] Department of ECE, Sahyadri College of Engineering and Management, Mangalore, India

Published online: 30 April 2021

exploring the concepts and effects of some basic functionalities in machine learning, computer vision and other such fields. The final vision is to ideally accomplish all the necessary tasks that a basic self-driving vehicle needs to perform. Ideally here refers to a methodology that is uncomplicated to understand, easy to modify, and open to improvisation. The report details on how computer vision technology along with some image processing functions further dealt with machine learning help to study the environment of the vehicle and enables the vehicle to find out a path and travel through it in the prescribed way.

Literature Survey

The process of automation of vehicles is carried out by various methods like sensor fusion, computer vision, path planning, actuator, deep learning, and localization [1]. Computer vision deals with the process of making computers acquire a high level of understanding from digital images [2]. Sensory fusion deals with the process of combining sensors and analyzing the obtained sensory data as a combined result of two or more sensors that would yield a better understanding of the environment of observation [3]. Deep learning can be seen as a wider family of machine learning, that includes various types of data representation unlike task-specific algorithms [4, 5]. Path planning is a primitive step that identifies the path where the vehicle is allowed to pass through. An efficient path planning can be done by plotting the shortest path between two points [6]. An actuator helps in moving or controlling the vehicle [7]. Navigation is the vehicle's capability to determine the position of the vehicle within its frame of reference and plan the most effective path towards the destination. To navigate in its environment, the vehicle requires a representation of the plot, i.e. a map showing the environment and the capability to interpret its representation. Edge detection comprises a set of mathematical equations that identify the points within a digital image where there is a rapid change or where the image brightness has discontinuities [8]. Since the usual color of the road is black, which is the least intensity color and that of the lane markings is either white or yellow both of which falls under the region of high-intensity colors, it is easier to differentiate the two regions, thus making the task of identifying the region of interest easier. The points where the image brightness changes sharply are grouped together and stored as a set of curved line segments [9, 10]. These line segments compose the edges of the region of interest [11]. Edge detection is a fundamental tool in image processing, machine learning, and computer vision, after which the functions are performed on the image [12]. The process of detection is done in three simple steps since all the traffic signals are designed in a common way to be understood easily by everyone. Since all three colors (red, yellow, and green) have high contrast ratios, this feature itself is used to separate the traffic signal from the rest of the objects in the image frame [13]. This separates the region of interest from the rest of the surroundings. Then to identify the colors individually, their RGB pixel values are considered and then the colors are classified [14]. For more precise performance, this process is carried out in six different steps. Input frame from the video, color filtering, edge detection, contour detection, detect bounding rectangle of contours, and then save candidate images for recognition. The data is then sent to the processor which then performs data set exploration and the required action is taken [15]. The process of detection is followed by the process of recognition, which involves recognizing various regions of interest on which the functions need to be performed. The first step in recognition is data set exploration. The data set used for training is GTSRB. Approximately 1000 images are taken for each class from different perspectives and different sizes. Twenty percentage of the training data set is stored for the validation process, hence it helps to increase the data set size artificially by a method called the augmentation process. Random images are chosen from the existing images and random rotations and translations are performed on the [16]. The transformed set of pixels is then added to the original set of pixels. The next step is training and model performance. Stochastic gradient descent is used as the optimizer. Instead of stochastic gradient descent other optimizers can also be used to increase the performance, since the work not only focuses on the optimizer. Another important performance indicator is batch size tuning because small batch sizes results in slow convergence whereas large batch sizes cause memory problems. Usually, middle batch sizes are preferred. To conclude, the paper includes two main phases: detection and recognition [17]. The first sign is detected from the real-time video stream using a CNN model. The detected sign is classified with an accuracy of 97.42%. However, when the video obtained by the RC Car is streamed online, the accuracy rate instantly decreases to 87.36%. The reason behind this rapid fall is because of the low sensitivity of the color filtering method used to the lighting and other objects. From the results, the classic image processing methods are eliminated and recurrent neural networks are used for detection as well as recognition phases. Hence, the result consists of each object in the whole picture [18]. By this, the decrease in the performance can be contradicted. The theory of neural networks, autonomous vehicles, and the process of how a prototype with a camera as its only input can be used to design, test, and evaluate the algorithm capabilities [19, 20]. The ANN is an efficient algorithm that helps in recognizing

the patterns within an image with the help of a training set that nearly contains 2000 images. The result thus obtained is 96% accurate. The main convenience of this project is that the design has successfully accomplished the tasks to be performed just by using one camera of average cost for navigation, which might as well be used for obstacle avoidance. In the end, the project concludes that the precision and accuracy of the output is directly proportional to the number of input images fed to the self-learning system [21]. The approach used here successfully meets all the requirements needed for autonomous navigation. In fact, the neural network is here used as the exact means to operate and control an autonomous vehicle in order to provide the user with a high accuracy rate reaching 96%. The perspectives for continuation of the work are in improvising the learning algorithm and enhancing the process of the testing the experiments. One of the key tasks is on-road obstacle detection and classification in the region ahead of the self-driving vehicles [22]. Since the key function of vehicle automation involves vehicle tracking or locating and associating vehicles in the frames, vehicle detection and classification becomes necessary. Due to the cost-effectiveness of vision-based approaches, they are given a higher priority over other approaches available for this task. This system uses a deep learning system accompanied by the convolutional region-based neural network. PASCAL VOC image dataset is used to train these dataset. The algorithm is advanced to the extent that it automatically identifies and classifies the obstacles like animals, pedestrians, and other vehicles with a time-dependent increased performance [23]. Using Titan X GPU to implement the system can help us achieve a frame rate of 10 frames per second or above during processing for an image frame of VGA resolution. The high frame rate thus obtained simplifies demonstration to an extent that it then becomes suitable for driving automated vehicles even on highways. During the performance testing, the results showed invariant performance under different textures of road and various climatic conditions that make the design well reliable for Indian rural roads as well.

Methodology

Slave Setup

As raspberry pi cannot handle both image processing and machine learning operations at the same time, a separate controller is used to control the motors of the bot. The motors are driven by using a L298N H bridge motor driver. Enable pins are connected to the PWM-enabled pins to control the speed of the motor. The powers to the positive and negative terminals of the motor are handled by the digital pins of the microcontroller. Figure 1 shows the setup required.

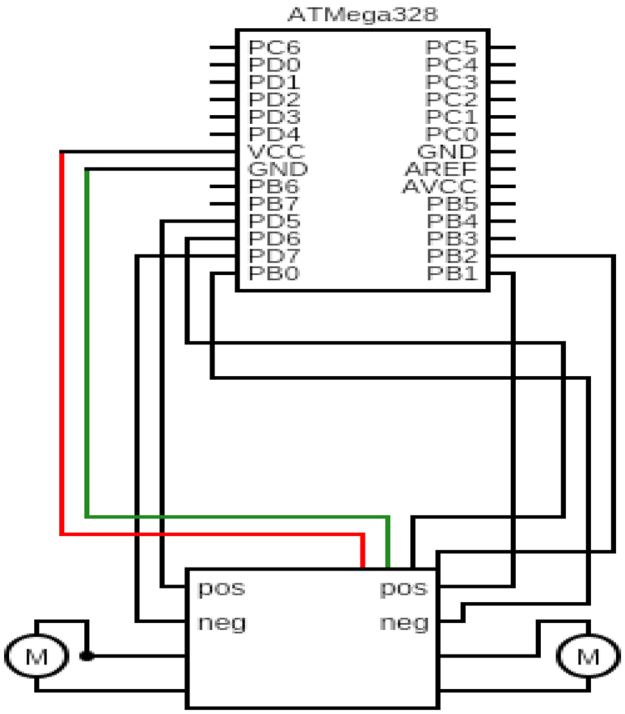

Fig. 1 Slave setup

OpenCV Using C

We have used raspberry pi as our microprocessor. To perform image processing on the objects we have used an open platform called OpenCV. There are other platforms like tensor flow but as OpenCV provides smooth performance we have decided to use OpenCV. We have used pi cam 2 to capture video. We have used this camera because we are performing processing with an image resolution of 480×360. Pi cam perfectly supports this resolution and is cheaper than other cams. After capturing the required frame we first convert the image into signature, before that we have to change the raspberry pi default BGR format of the image to RGB format. To detect the lanes we have to apply a perspective wrap on the image. To analyze an image 5 frames of references are needed, object co-ordinate frame, word coordinate frame, camera coordinate frame, image coordinate frame, image coordinate frame, and pixel coordinate frame. By applying transformations to all these 5 frames we get a perceptive wrap of an image. To apply perspective wrap first we have to create a region of interest around the working region. Then a perspective transform is taken over the region to get a bird's eye view of the image. A fresh frame of the same image is taken and a canny edge detection algorithm

is applied to it. Now the wrapped image is added with this frame and the lanes can be detected accurately. The actual distance of the lanes are found out by dividing the region of interest equally and finding the maximum intensity levels of each element. Again the array is divided into two parts to detect the left and right part of the lanes. The array elements having the max intensity corresponds to the lane position. After finding the distance of the lanes the midpoint is taken. Taking the center of the camera frame as then we have used raspberry pi as our microprocessor. To perform image processing on the objects we have used an open platform called OpenCV.

There are many other platforms for image processing we have used open CV as it is a open source platform. We have used pi cam 2 to capture video. We have used this camera because we are performing processing with image resolution of 480 × 360. Pi cam perfectly supports this resolution and is cheaper than other cams. After capturing the required frame we first convert the image into signature, before that we have to change the raspberry pi default BGR format of the image to RGB format. To detect the lanes we have to apply a perspective wrap on the image.

To apply perspective wrap first we have to create a region of interest around the working region. Then a perspective transform is taken over the region to get a bird's eye view of the image. A fresh frame of the same image is taken and canny edge detection algorithm is applied to it. Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function. By convolving G and f we for a smoothed image, which is given by fs. After this it is followed up by calculating the gradient magnitude and direction. The gradient magnitude is computed at every point and direction to estimate the edge strength and direction at every point, which is called edge gradient.

The equations of the process involved in canny edge detection are from Eqs. (1–7) Thresholding of the image is done and is added with canny edge detected output. Thresholding is done to extract or enhance the image. To extract an object from the image one way is to separate the object and background by using a threshold. At any point (x, y) in an image $f(x, y) > T$ is called an object point, otherwise called as a background point. Equation (8) gives the mathematical equation of the process. As our image is a grayscale image we set T = 0.5. Now the wrapped image is added with this frame and the lanes can be detected accurately. The actual distance of the lanes is found out by diving the region of interest equally and finding the maximum intensity levels of each element. Again the array is divided into two parts to detect the left and right part of the lanes. The array elements having the max intensity corresponds to the lane position. After finding the distance of the lanes the midpoint is taken. Taking the center of the camera frame as a reference the bot has to adjust its position. If the value of the distance is negative then the bot has to move left. The magnitude of the turn depends on the distance from center-lane.

$$G(x, y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}, \tag{1}$$

$$f_s = G(x, y) \otimes f(x, y), \tag{2}$$

$$G_x = \frac{\partial f_s}{\partial x}, \tag{3}$$

$$G_y = \frac{\partial f_s}{\partial y}, \tag{4}$$

$$\text{Edge Gradient}\,(G) = \sqrt{G_x^2 + G_y^2}, \tag{5}$$

$$\text{Angle}\,(\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right), \tag{6}$$

$$\text{Edge Gradient} = \left|G_x\right| + \left|G_y\right|, \tag{7}$$

$$t(x, y) = \begin{cases} 1, & \text{if } f(x, y) > 0.5 \\ 0, & \text{if } f(x, y) \le 0.5 \end{cases}. \tag{8}$$

Master/Slave Communication

Here parallel communication is setup between the microcontroller and raspberry pi using GPIO pins of raspberry pi and four digital pins of microcontroller. Conditions are applied for different distances between the frame center and the lane center. Depending on the conditions the bot is moved left or right towards the frame center (Figs. 2, 3).

Machine Learning

To detect the traffic signals, obstacles, and traffic signs labelled machine learning is used, i.e. the data set used by the authors will be labelled, these labelled images are compared with the real-time images taken by the cam. To classify the images we need a machine learning model. To implement a machine learning model, we require a data set which sufficient enough for the model to classify between the images. The authors take 400 samples of the object to be detected these are called the positive images and then 300 negative images are taken, that is those areas which do not belong to the object to be detected. Histogram equalization is applied to all the images after converting the RGB image to a grayscaled image. If we consider values of continuous intensity and if r is the intensities of the image to

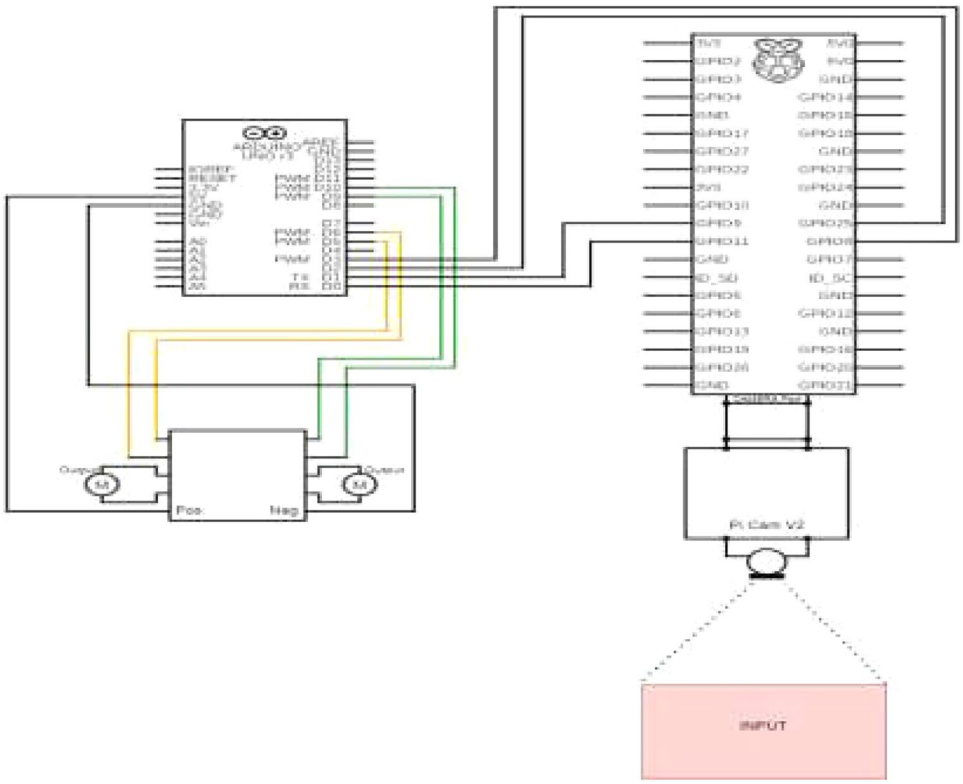

Fig. 2 Master–slave communication setup

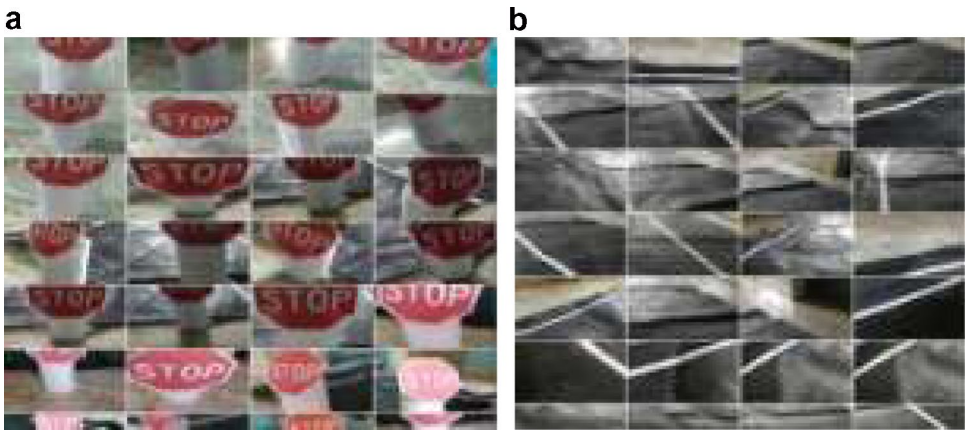

Fig. 3 **a** Positive image. **b** Negative image

be processed, we focus attention on intensity mappings of the form $s = T(r)$. The purpose of using histogram equalization is to uniformly distribute gray value, by making the probability distribution function the image intensity uniform. By creating a info. file of the images we store the exact location of the image and also the number of objects in each image. The info. file is created by using the OpenCV integrated annotation tool. By using these images a training system is developed to recognize the object. This training method involves cascading and then a XML file format of the learned method is created. This file has to be uploaded to the program to apply the remaining operations. After creating the info. file cascade training is done on the image using opencv_traincascade. For given an training example, $(x_1, y_1) \ldots (x_n, y_n)$, where $y_i = 0, 1$ respectively, the cascade transform initializes the weights for y_i respectively. For training examples from $0 - N$, the transform normalizes the weights, so that the weights are of the form of probability distribution. For each feature, we train the classifier which is restricted to use a single feature, the errors are evaluated and classifier with the lowest error is taken and the weights are updated. Finally a classifier strong enough to classify between is build, which is given by Eqs. (9–11). Where h_t is a classifier, where alpha and beta are used for updation of the weights. These values are chosen randomly. The best results of cascade classifier will consist of 38 stages. To train the

detector of image size 240 × 240 a total of 30 min was taken. After detecting the image, the next step is to stop before the sign, for this a distance should be known from the bot to the detected sign. The authors have used Haar cascade transformation to implement the model. To find the distance we use linear equations. A linear equation for eg: $y = mx + c$, where x is a weight of the equation. This weight and the intercept are found manually. After getting the distance from the sign after the sign is detected, a threshold is set to stop the bot when the distance is reached. The whole working flow of the system is shown in Fig. 4.

$$h(x) = \begin{cases} 1 \sum_{t=1}^{N} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{N} \alpha_t, \end{cases} \quad (9)$$

$$\text{Where, } \alpha_t = \log \frac{1}{\beta_t}, \quad (10)$$

where, update weights is given by:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad (11)$$

Results and Discussion

Results obtained from this paper are as follows. The first result is lane detection. From Fig. 5 it has detected the lane and the lane center is at a distance of − 18 from the frame center which indicates us to take a left turn and in Fig. 6 it is giving a value zero, which is a condition for forward direction.

Next part of our paper was to detect the signs in sides of the roads, we have taken 500 negative images of the stop

Fig. 5 Lane detection center is − 18 from the frame center

Fig. 4 Work flow of the bot

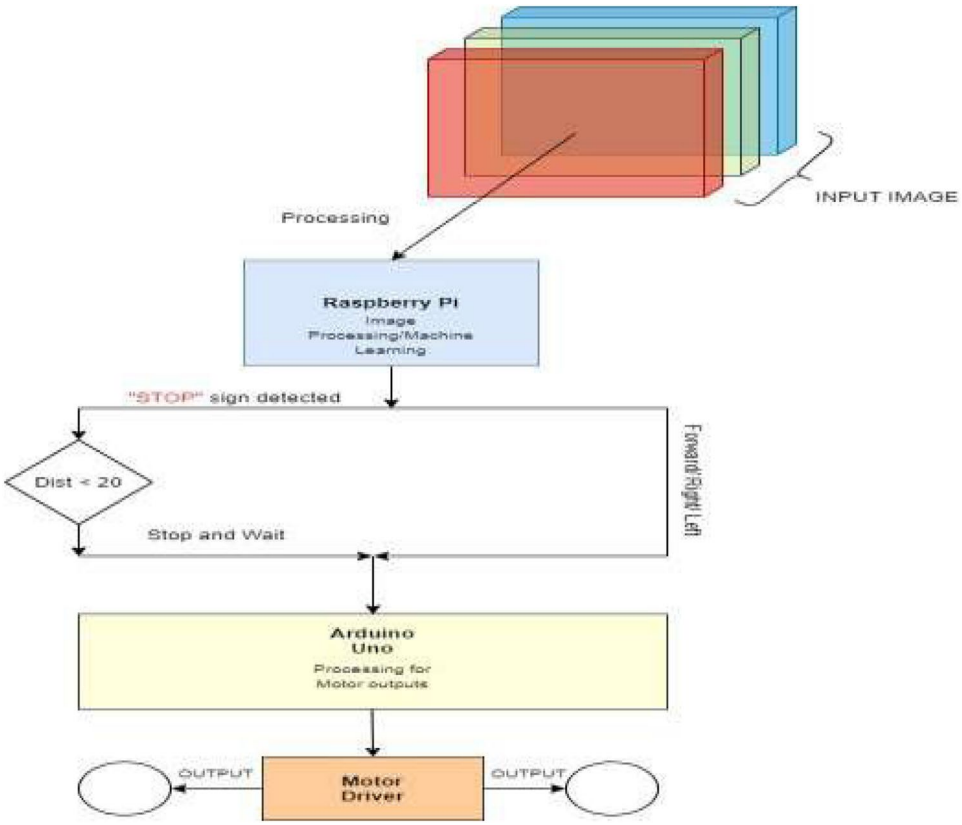

Fig. 6 Lane detection with distance value 0

sign surroundings and 60 positive images of the stop sign. By applying Harr cascade transform we have trained our system to detect the stop sign. From Fig. 7 we have detected the sign by marking it with a rectangular box.

From the test results the efficiency of detecting the lane in dark environment is very low. But during sufficient brightness around it was able to distinguish between the sign and the surroundings properly. The efficiency of detection can be improved by using good cam's light night vision camera's of raspberry pi is available. During the motor test the bot was able to move in the centre with a total efficiency of 75%. Due to the lower frame rates of capture and low processing speed of our microprocessor the efficiency was reduced to 75%. If we use a processor with RAM of 4 GB the data processing and frame rates can be increased i.e. increasing the efficiency. Iterations were performed for different samples and number of stages, and evaluation metrics like accuracy, precision, and recall of detection of the traffic signals where calculated, Table 1 depicts the results given for different video samples. Video sample 4 was taken in indoor conditions and all the other samples from Videos 1–3 were taken from outdoor. In each video sample, the signs were shown to the bot 12 times. The formula for calculating is given in Eqs. (12–14) from the confusion metric. Our best results were from a sample size of 400:300 positive and negative samples respectively (Fig. 8).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{12}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{13}$$

Fig. 7 Stop sign detection

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{14}$$

Conclusion

The objective of the proposed work was to come up with a cost-effective and a very efficient automatic driving car, with a learning algorithm that is easy to understand and further can be improvised. Our bot was successful in following the straight path when the distance measured was zero, it was able to detect the stop sign successfully in the outdoor environment. But we found that there was a delay in the communication between the raspberry pi and the Arduino. This was because of the 1 GB RAM of raspberry pi, which was not enough to perform image processing, machine learning and also send signals to the Arduino board for motor control simultaneously. The visual results indicated that our bot performed to the expectations. To get more smooth performance, the solution is to r use a 4 GB RAM processor. As the machine learning technique we used required many

Table 1 Evaluation table for different samples/sample size

Video_Samples	Number of positive samples	Number of negative samples	Accuracy	Precision	Recall
Video_1 (Out-Door)	150	120	0.831	0.800	0.796
Video_2 (Out-Door)	150	120	0.808	0.785	0.785
Video_3 (Out-Door)	150	120	0.792	0.698	0.769
Video_4 (In-Door)	150	120	0.452	0.328	0.56
Video_1 (Out-Door)	250	170	0.851	0.804	0.774
Video_2 (Out-Door)	250	170	0.827	0.783	0.752
Video_3 (Out-Door)	250	170	0.810	0.775	0.798
Video_4 (In-Door)	250	170	0.505	0.398	0.598
Video_1 (Out-Door)	400	300	**0.943**	0.900	0.870
Video_2 (Out-Door)	400	300	**0.948**	0.878	0.922
Video_3 (Out-Door)	400	300	**0.946**	0.887	0.900
Video_4 (In-Door)	400	300	**0.523**	0.425	0.659

The bold numbers specify the highest value of accuracy for a particular value of Number of positive samples and Number of negative samples

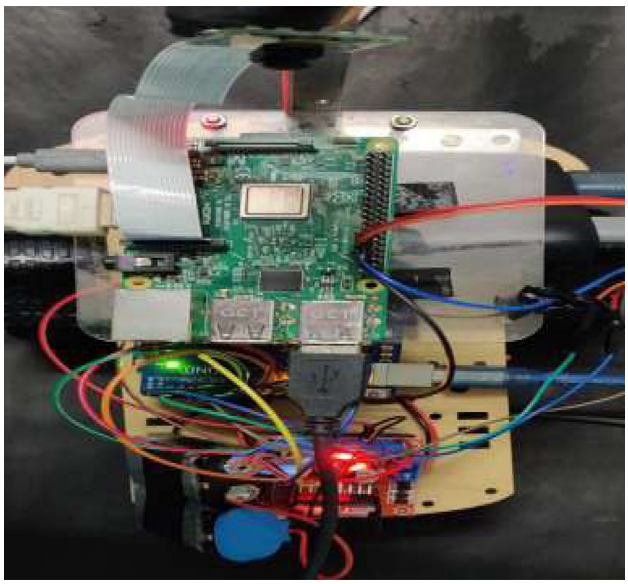

Fig. 8 Hardware connection

samples of the picture predefined for training. This makes the process semi-autonomous, so we will have to figure out some algorithm which will learn by itself without any predefined label for each subject. The camera which we used was not high-end, so as per the evaluation from Table 1 it clearly was not able to detect the lanes in the indoor environment. We have to change the setup when we shift from an outdoor environment to indoor or vice-versa.

References

1. Ballard DH, Brown CM. Computer vision. 1st ed. Prentice Hall; 1982.
2. Huang T, Vandoni CE, editors. Computer vision: evolution and promise. In: 19th CERN School of Computing. Geneva: CERN; 1996. pp. 21–25. https://doi.org/10.5170/CERN-1996-008.21. ISBN 978-9290830955.
3. Elmenreich W. Sensor fusion in time-triggered systems. PhD Thesis (PDF). Vienna, Austria: Vienna University of Technology; 2002.
4. Haghighat MBA, Aghagolzadeh A, Seyedarabi H. Multi-focus image fusion for visual sensor networks in DCT domain. Comput Electr Eng. 2011;37(5):789–97.
5. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell. 2013;35(8):1798–1828. https://doi.org/10.1109/tpami.2013.50. ArXiv: 1206.5538 freely accessible.
6. Schmidhuber J. Deep learning in neural networks: an overview. Neural Netw. 2015;61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003. PMID 25462637. ArXiv: 1404.7828 freely accessible.
7. Umbaugh SE. Digital image processing and analysis: human and computer vision applications with CVIP tools. 2nd ed. Boca Raton: CRC Press; 2010. (**ISBN 978-1-4398-0205-2**).
8. Barrow HG, Tenenbaum JM. Interpreting line drawings as three-dimensional surfaces. Artif Intell. 1981;17(1–3):75–116.
9. Lindeberg T. Edge detection. In: Hazewinkel M, editor. Encyclopedia of mathematics. Springer Science Business Media; 2001. (**ISBN978-1-55608-010-4**).
10. Lindeberg T. Edge detection and ridge detection with automatic scale selection. Int J Comput Vis. 1998;30(2):117–54.
11. Anggraini D, Siswantoko W, Henriyan D, Subiyanti DP, Aziz MVG, Prihatmanto AS. Design and implementation of system prediction and traffic conditions visualization in two dimensional map (case study: Bandung city). In: 2016 6th International conference on system engineering and technology (ICSET); 2016.
12. Shapiro L, Stockman G. Computer vision. Prentice Hall Inc.; 2001.
13. Duda RO, Hart PE. Use of the Hough transformation to detect lines and curves in pictures. Commun ACM. 1972;15:11–5.
14. Hough PVC. Method and means for recognizing complex patterns. U.S. Patent 3,069,654, Dec. 18, 1962.

15. Hough PVC. Machine analysis of bubble chamber pictures. In: Proc. Int. Conf. High Energy Accelerators and Instrumentation; 1959.
16. Nunes E, Conci A, Sanchez A. Robust background subtraction on traffic videos. In: 2011 18th International conference on systems, signals and image processing (IWSSIP); 2011. pp. 1–4.
17. Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: IJCAI81; 1981. pp. 674–679.
18. Pang CCC, Lam WWL, Yung NHC. A novel method for resolving vehicle occlusion in a monocular traffic-image sequence. IEEE Trans Intell Transp Syst. 2004;5:129–41.
19. Chiu C, Ku M, Wang C. Automatic traffic surveillance system for vision-based vehicle recognition and tracking. J Inf Sci Eng. 2010;26:611–29.
20. Gordon RL, Tighe W. Traffic control systems handbook. Washington, DC, USA: U.S. Department of Transportation Federal Highway Administration; 2005.
21. Hsieh J-W, Yu S-H, Chen Y-S, Hu W-F. Automatic traffic surveillance system for vehicle tracking and classification. IEEE Trans Intell Transp Syst. 2006;7(2):175–87.
22. Jung Y-K, Ho Y-S. Traffic parameter extraction using video based vehicle tracking. In: 1999 IEEE/IEEJ/JSAI international conference on intelligent transportation systems, proceedings, pp. 764–769.
23. Cheung S-CS, Kamath C. Robust background subtraction with foreground validation for urban traffic video. EURASIP J Apple Signal Process. 2005;2005:2330–40.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

View publication stats