Self-Driving robotic car utilizing image processing and machine learning

Husam A Almusawi, Mohammed Al-Jabali, Amro M Khaled, Korondi Péter, Husi Géza

Department of Mechatronics, University of Debrecen, Debrecen 4028, Hungary

E-mail: husam@eng.unideb.hu

Abstract. The major goal of this paper is to build and represent a prototype of a fully autonomous car that employs computer vision to detect lanes and traffic signs without human intervention using limited computing capacity. The project contains an embedded system represented by a Raspberry Pi 3 which serves as the image processing and machine learning unit. This method requires a stream of images as input for the computer vision using OpenCV2 library with C++ programming language along with Haar Cascade Classifier for the detection of traffic signs. The Raspberry Pi will send binary signals to the Arduino UNO which is responsible for merging those signals with the ones from the ultrasonic sensor and producing new signals which are sent to the motor driver to control the direction and speed of the dc motors. The system was able to detect the lane and respond to changes in lane direction, as well as to detect traffic signs and give appropriate responses.

1. Introduction

Driven vehicles play an essential role in human life both in terms of daily life and economics [1]. However, humans can make mistakes or errors that can endanger people inside or outside the vehicle. According to the National Highway Transportation Safety Administration (NHTSA), the majority of these mistakes are avoidable or preventable [2] and they are primarily caused by speeding, distraction, or Lack of traffic knowledge [3]. Recent research has shown that autonomous vehicles or driver self-assistance can reduce accidents or crashes caused by human errors [4], [5]. There's also the concern of driver shortage to consider. According to the Financial Times web page, truck drivers' numbers shrink over time in Europe and that can cause serious economical consequences [6]. Therefore, Autonomous vehicles can contribute not only to eliminating human errors and traffic control but also can be considered as a solution to the drivers' shortage issue [7].

A self-driving car that can travel without human intervention has to consider various situations such as obstacles, road conditions, and traffic signs in real-time utilization depending mainly on the hardware, software, and computer vision. According to SAE International [8], there are mainly 6 levels of autonomous vehicles leveled from 0 (No automation) to 5 (Full automation) [9]. The main aim of this paper is to build a level 5 self-driving car model that can adapt to real-time traffic conditions as well as detect objects, obstacles, and traffic signs in its surroundings and respond to them accordingly. This model detects the vehicle's track by utilizing image processing to recognize the lane's white lines, and it employs machine learning approaches to detect various traffic signs. Additionally, Ultrasonic sensors to determine the end of the lane are implemented to ensure the vehicle's safety. The system receives a

stream of images coming from the MIPI CSI-2 camera [10] which is going to be processed by the embedded system (Raspberry Pi 3) and sent to the microcontroller (Arduino UNO) [11]. The Arduino UNO sends the output signals to the motor driver to control the DC motors.

2. Related work

2.1. Approaches that utilize data collected from previous stages

Autonomous vehicles have ushered in a technological revolution, particularly in the field of computer vision, which includes image processing, machine learning, and deep learning algorithms [12]. A methodology called "Supervised learning" was used by Chishti et al., where a huge amount of data was collected by a driven car with high accuracy [13]. Another method was used by Memon et al., which is called the "Target Car" method in which the vehicle receives it is data by following the direction and location of another car [14]. These methods rely on the data collected from previous stages.

2.2. Lane detection approaches

It is critical for an autonomous car to detect the lanes of the road on which the vehicle is driven. One of the techniques used to detect the lanes is "Hough Transformation" which implements a set of algorithms for feature extraction, and SVM (Support Vector Machine) [15] for machine learning algorithms which can be implemented with lines, edges, and the region of interest [16]. The same research could get high accuracy considering computing capability and the difference in light intensity that can affect the system. According to Satozda et al. [17], the accuracy of the same system has increased by improving the Hough Transformation algorithm in lane detection and by introducing the (HAHT) which means hierarchical additive Hough Transformation. Another approach [18] for solving the task is the use of YOLO (You Only Look Once) [19] and CNN (Convolutional Neural Network) [20] algorithms which detect the lane and its objects on highways but not on urban roads. According to Daigavane et al. [21]. Lane detection algorithm could be applied by using Canny Edge detection and Ant Colony algorithm along with Hough transform. The advantage of this system is the ability to work under painted and straight roads [22].

2.3. Traffic signs detection approaches

Traffic sign detection is applied by machine learning. By applying the CNN network and by converting the inputs images into grayscale, the traffic sign can be detected [23]. Using deep CNN architecture has a higher accuracy rate compared to other methods. A similar approach came with sign detection using a classification based on CNN-SVM with a 98.6% accuracy rate which takes a raw image of a sign converted to grayscale and then trains the image for feature extraction and classification. This model is tested before processing the data and if the process failed, it should be trained again for optimal outcomes [24].

2.4. Contribution of the paper

This paper will contribute to a self-driving automobile algorithm that can be processed with limited computing capacity. The primary goal is to develop a completely autonomous model that can drive safely in real-time. Considering the hardware requirement which contains a Raspberry Pi 3, lane detection will be applied by converting the input images into grayscale instead of Hough transformation and then to Canny edge detection which makes it easier for detecting the edges of the lines.

Haar classifier can be used for signs or object detection with C++ programming language along with neural network principles [25]. This method requires positive and negative samples (images), and the result of the training is a cascade file that can be imported to the code as the OpenCV2 library supports these types of files [26].

3. Literature review

The approaches of this study can be divided into two groups, controlling DC motors, and utilizing basic concepts of computer vision and image processing.

3.1. Controlling DC motors

- DC motors are working under Faraday's law of electromagnetism and there are many ways of controlling their speed and rotation direction [27].
- For this project, PWM (Pulse-width modulation) is used to control the speed of the motor which levels from 0 to 255 in the Arduino IDE platform.
- Reversing the polarity of the DC motors will cause the direction of rotation to be reversed and for this purpose, an H-bridge is used [27].
- The motor driver receives the signals from the Arduino UNO and according to these signals, it varies the speed and determines the direction of rotation of the DC motors.
- The Arduino UNO gets four signals from the Raspberry Pi that processes the data of image processing and machine learning [28].

3.2. Computer vision

- The main goal of computer vision is to reconstruct images and videos captured by digital cameras or the optical sensor array-based natural scene. Every scene consists of pixels, edges, shape, color, and texture and they are processed in artificial intelligence algorithms [29].
- OpenCV library is used for image processing and machine learning which converts the image from one format to another according to the type of data needed such as converting to grayscale, RGB, BGR, and thresholding [30].
- Thresholding in OpenCV is the technique of replacing pixel values. Each pixel value is compared with the threshold value. when the pixel's value is smaller than the threshold, it will be set to 0 (black). If the value is greater than the threshold it will be set to be 1 (white) as shown in (1) [31].

$$g(x,y) = \begin{cases} 0, & \text{if } g\,(x,y) \leq \mathrm{T} \\ 1, & \text{if } g\,(x,y) > T \end{cases} \quad (1)$$

- Canny edge is one of the most popular edge detectors. It can detect the edges with a high accuracy rate and catch as many edges on an image as possible. Also, it has the least amount of noise compared to other detectors. [32].
- A histogram function gives the distribution of pixels in a graphical representation. the x-axis of the histogram represents the different color values between 0 and 255 while the y-axis is a representation of the number of times a specific color value occurs in the image [26] the values of the histogram function can be obtained by using pointers which are variables that store memory addresses. They are used for storing other variables, addresses, or memory items. It is essential, particularly for dynamic memory allocation [33] and they are used to detect the white lines of the lane.
- Haar cascade classifier is used for machine learning and training, it's utilized for sign detection. This method requires 40 positive samples (traffic sign images) and around 300 negative samples (environment or surroundings of the traffic sign).

The task can be solved by using the OpenCV library with C++ programming language for image processing and machine learning where the input of the system is an Image or stream of frames, and the output of the system is four binary signals sent to the Arduino UNO which converts those signals and merge them with the ultrasonic signals to produce six output signals that control the rotation direction and the speed of the DC motors. The solution of the system is illustrated in 'Figure 1' which shows a closed loop-controlling system with the feedback coming from the camera and the ultrasonic sensors.

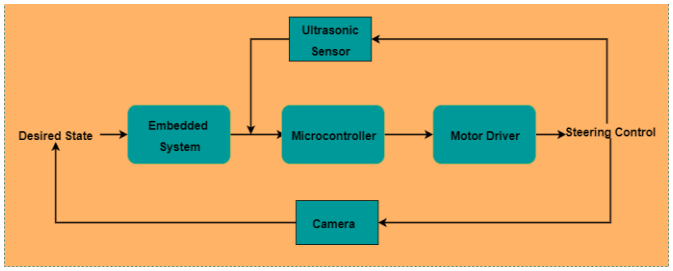

Figure 1. Closed loop controlling system

4. System assembly

The components used in this study are an embedded system (Raspberry Pi 3 B+) and a microcontroller (Arduino UNO). It also consists of an MIPI CSI-2 camera, two Ultrasonic sensors, two power units, an H-bridge DC motor driver, car chassis, and 4 DC Motors. The assembly is shown in 'Figure 2'.

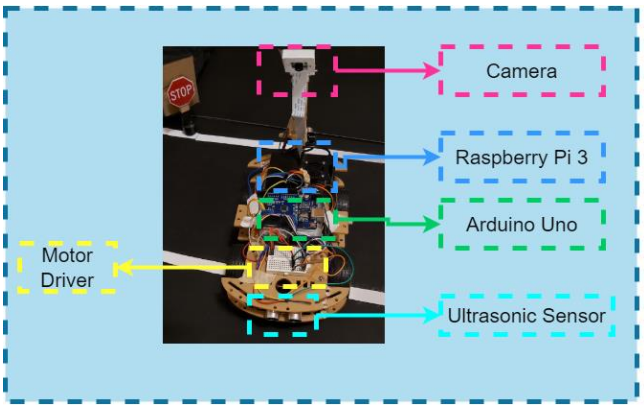

Figure 2. Components of the project

The connection diagram is shown in 'Figure 3'. An MIPI CSI-2 camera is connected to the Raspberry Pi board which is the input of the system. The Raspberry Pi and the Arduino UNO are fed from the same power unit. The serial communication between Raspberry Pi and the Arduino UNO is done by connecting the GPIO which is notated by Wipi21 to Wipi24 on the "WiringPi" [34] library, as the most significant bit (MSB) and the least significant bit (LSB) to any four Arduino Digital Inputs. The motor driver and the Arduino UNO are connected by six jumper wires in which four wires are controlling the rotational direction of the DC motors and two wires for controlling the rotational speed of the DC motors by PWM. Another power supply is used for the motor driver to supply the required voltage for the motors, all hardware equipment is installed on the car chassis.

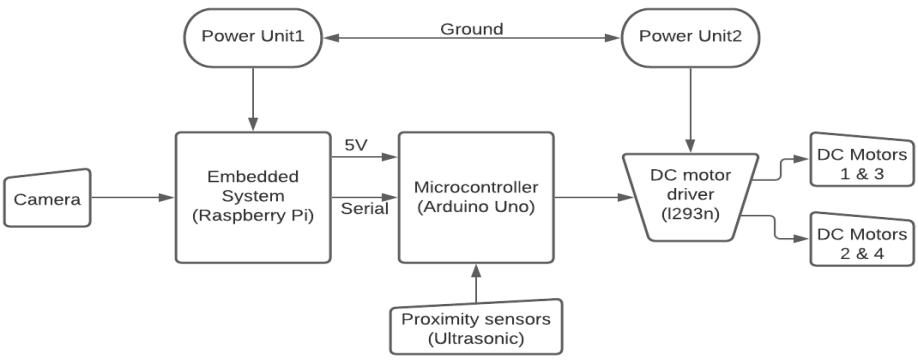

Figure 3. Connection diagram

5. Hardware and software architecture

5.1. Embedded system (Raspberry Pi 3 B+)

Raspberry Pi 3 is the main processing unit in this project, it has a Quad-core processor (BCM2837) 64-bit with 1 GB DDR2 RAM and a clock Frequency of 1.2GH [35]. It requires a 5.1 VDC and 2A power source and contains built-in Bluetooth and Wi-Fi modules. 32GB MicroSD is used with it to install the necessary software requirements and libraries. Most importantly, the board has a "15-pin MIPI Camera Serial Interface (CSI-2)" which is required to provide the system's input. The Raspberry pi 3 contains 40-GPIO pins which have two 3.3VDC pins, two 5VDC pins, and nine Ground pins. The rest of the pins are I/O pins that can send or receive either High signals (3.3VDC) or Low signals (0VDC). PWM signals can only be obtained through four pins which are GPIO18, GPIO19, GPIO12, and GPIO13 [36]. Raspberry Pi is the unit responsible for image processing and machine learning algorithms as shown in 'Figure 4'.

5.2. Microcontroller (Arduino UNO)

It is a board based on "ATmega328P" equipped with 6 analog inputs and 14 digital I/O pins (6 of them are used as PWM) that can be integrated with other circuits and boards [37]. It is connected to the motor driver that controls the steering of the vehicle. It can be powered with the same source as the Raspberry Pi which requires (5.1VDC). Arduino UNO is used in this study for the purpose of dividing the process between it and the Raspberry Pi to reduce the lagging error. It is used to merge the binary 4-bit signals coming from the Raspberry Pi with the signals from the ultrasonic sensors to produce a new signal to the motor driver shown in 'Figure 4'.

5.3. Motor driver (L298N)

The Full-Bridge circuit is commonly used to change the polarity and control the speed of DC motors. It can also accept TTL logic levels [38] coming from the Arduino UNO. The supply voltage can be up to (46 VDC, 4A) and (4.5-7VDC) as a logic supply voltage. Because the Arduino UNO does not provide enough power to run the motors on its own, it requires an external circuit that connects an external power supply to the DC motor and receives signals from the Arduino UNO as shown in 'Figure 4'.

5.4. Power supplies

In this study, there are two power supplies. The first one has approximately 5(VDC) to supply the necessary power to the Raspberry Pi and the Arduino UNO. The second supply has approximately 8(VDC) to supply the necessary power to the motor driver and DC motors. Both of them have the same ground. The Arduino UNO is not connected directly to the power supply, but it is connected to the Raspberry Pi GPIO. The reason for having two power sources is to protect the Raspberry Pi from the reverse current of the DC motors and to give the motors extra power than other components. Hence, the control and power circuits are isolated.

5.5. Vehicle Safety

The safety of the vehicle and the objects surrounding can be achieved by stopping the vehicle once it's necessary and that can be in two ways:

5.5.1. Finding the lane ends. Because it estimates the value of pixels in the white line, the histogram function aids in determining the end of the lane. The aggregate of these values is calculated, and the end line values are obtained by reading the terminal while the robot is in the middle of the track. The values change according to white lines as the car moves. When placed near the end of the lane, however, it returns the smallest or largest value, which can be used to tell the car to stop when it reaches the end of the track.

5.5.2. Using proximity sensors. Two ultrasonic sensors are used to sense if there is an object in front or behind the vehicle 'Figure 5' in the range of 20 cm. When an obstacle is detected, the microcontrollers give the order to stop the vehicle

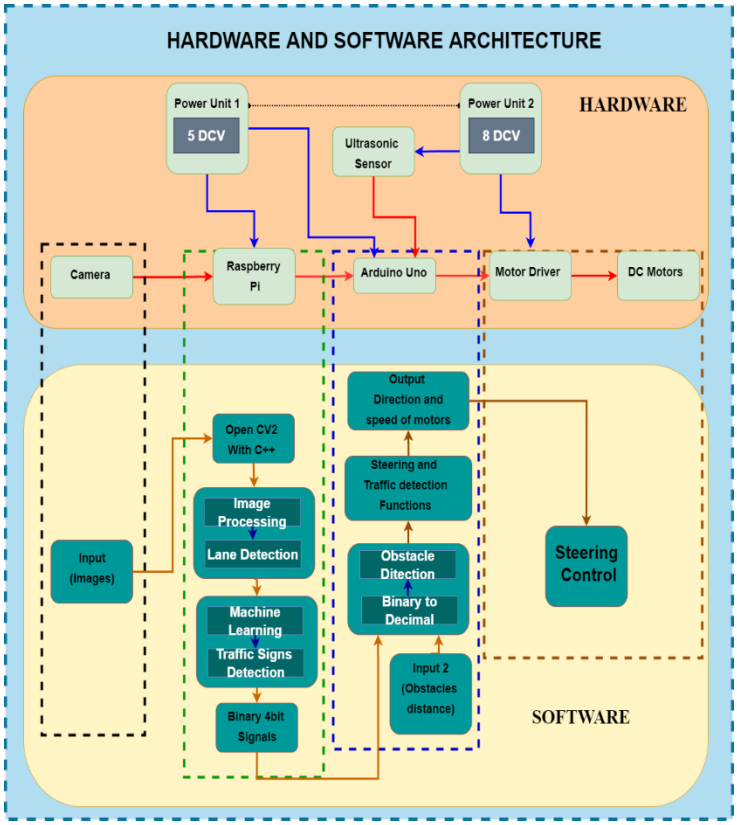

Figure 4. Hardware and software architecture

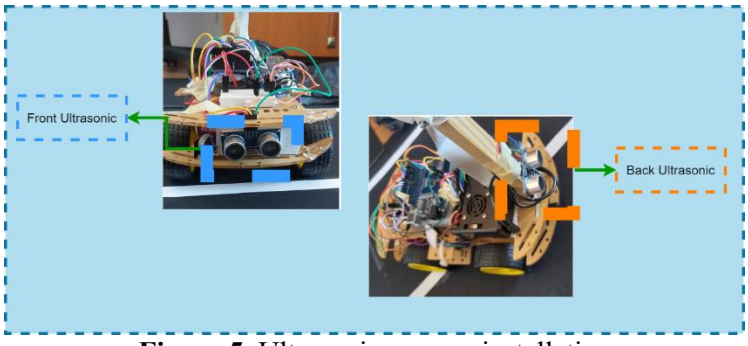

Figure 5. Ultrasonic sensors installation

5.6. Communication between Raspberry Pi and Arduino

The communication between Raspberry Pi and Arduino UNO is carried by four digital pins. The signals sent by Raspberry Pi are in the form of binary signals e.g., 0000 (forward). For this purpose, it is important to identify the most significant bit and least significant bit. Four GPIO would be chosen which are Wipi 21(LSB) to Wipi 24 (MSB) and connecting them to the digital pins on the Arduino UNO as INPUTS. The binary signals are converted into decimals by using equation (2) below:

$$Decimal = (Wipi24 \times 2^3) + (Wipi23 \times 2^2) + (Wipi24 \times 2) + Wipi\ 21 \quad (2)$$

If the digital signal of Wipi is high, then Wipi = 1, and if it is low, then Wipi =0. Digital pins 9 and 10 can be utilized as PWM to control the speed of the dc motor, while pins 2 to 5 control the polarity. The decimal numbers are used to give the orders to the motor driver are the following:

```
if (decimal == 0){ Forward();  }
else if (decimal == 1){ R_lvl1();}
else if (decimal == 2){ R_lvl2();}
else if (decimal == 3){ L_lvl1();}
else if (decimal == 4){ L_lvl2();}
else if (decimal == 5){No_Entry2();}
else if (decimal == 6){Stop_SignFunction();}
else if (decimal == 8){Stop();}}
```

The functions are made as needed and they depend on many factors such as the power supply, the weight of the vehicle, and the type of DC motors

6. Methodology

6.1. Lane detection

After installing the necessary operating system and libraries for the Raspberry Pi, the input is processed by the system. The images captured by the camera are converted to 400 × 240 pixels and also to RGB color format. As shown in 'Figure 6', a region of interest is chosen to be straightened out [33] and converted to grayscale format, therefore, thresholding and canny edge detection can be applied to it for the histogram function. Dynamic arrays or vectors can be used to store the values of pixels of the histogram function. Those values can be called by C++ programming by using pointers which are going to search for the values beginning from the lift side to the right side [26]. For both lines, the detected white color values will be combined to draw a yellow line 'Figure 7'. Another line is drawn which is the average of both lines and moves in the same direction as the vehicle 'Figure 7'. This is the light blue line. The purple solid line doesn't move with the direction of the vehicle, and it's drawn to calculate the difference distance between it and the average line 'Figure 7'. The steering orders would be sent to the Arduino UNO in the form of binary signals 'Figure 8' according to the difference between the solid line and the average line.

6.2. Traffic sign detection

As shown in 'Figure 10' the traffic sign detection is achieved by a machine learning algorithm which depends mainly on Haar Cascading Classifier [33] and to make the process easier the following steps are followed:

- A program called Cascade-Trainer-GUI [39] is used for the training of traffic signs.
- It requires approximately 50 positive images for each traffic sign and 300 negative images for the surroundings.
- The output of the training process will be a file with the extinction of (.xml) [26].
- The file can be imported to the C++ program by an individual frame and the difference between height and width can be calculated by (*px*).

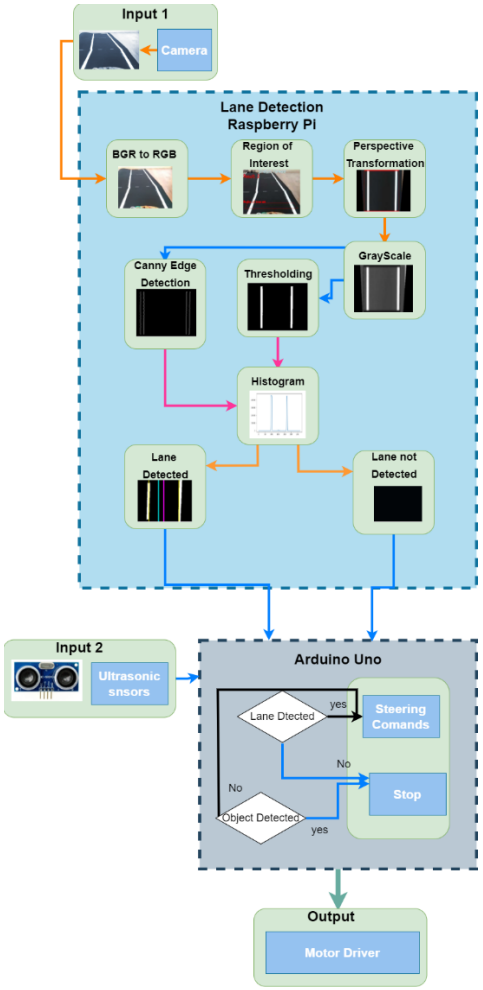

Figure 6. Lane detection flowchart

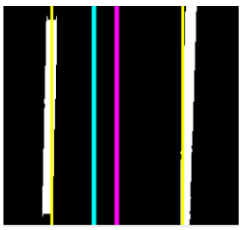

Figure 7. Line detection

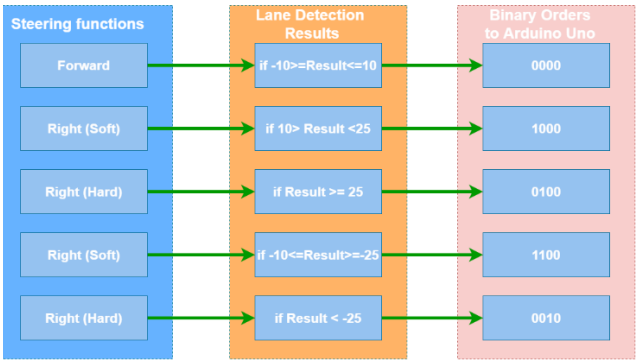

Figure 8. Steering command

6.3. Getting the distance in centimeters

- To get the distance between the vehicle and the traffic signs in cm dynamic equation (3) should be used:

$$y = mx + c \tag{2}$$

- Where (*y)* is the precalculated distance between the vehicle and the sign in (cm) and (*x)* is the difference in (px) between different points (height and width) of the detected sign. Two equations are needed in different positions to find the full equation of (*y)* by finding *m* and *c* and as shown in 'Figure 9' the distance between the stop signs and the vehicle is shown in cm.
- After the distance calculation, commands can be given to the Arduino UNO in form of binary signals to do the required operation such as making a U-turn or parking the vehicle.

Figure 9. Stop sign distance in cm

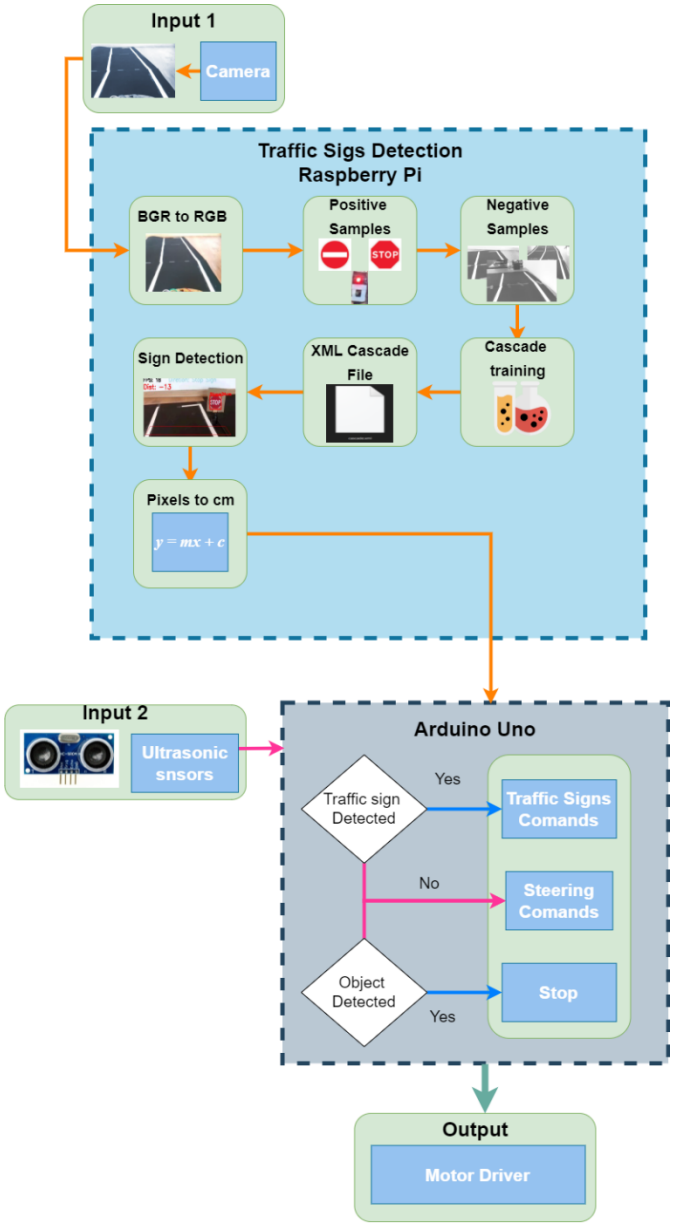

Figure 10. traffic sign detection flowchart

7. Tests and results

By applying the principles mentioned previously for the connection of the hardware units and the software methods which are image processing and machine learning algorithms using OpenCV2, the lane could be detected as shown in 'Figure 7' by the program as well as the traffic signs (Stop Sign, Traffic Light, and No entry Sign as shown in Figures 9, 11, and 12. Furthermore, the car and its surroundings were kept safe by stopping at the lane's end and using the ultrasonic sensor to stop the vehicle at a safe distance from obstacles. The following are the results of the study:

- The distance could be obtained in cm between the vehicle and the traffic sign.
- The vehicle stops at the end of the track or when an object or an obstacle appears in front of the car.
- The car responds to the stop sign and stops for five seconds then continues driving.
- The car responds to the traffic light and stops if it is red and continues driving if it is green.
- The car responds to the no entry sign and makes a U-turn.

Figure 11. Traffic light detection

Figure 12. No entry sign detection

However, the accuracy of the detection of either the lane or the traffic sign is being affected by external light sources as shown in 'Figure 13'. The experiment was carried out in different light conditions (sunny, cloudy, and Artificial light) 10 times for each condition with adjusting the thresholding and Canny edge detection manually. That is, the system's accuracy is found to be around 88%.

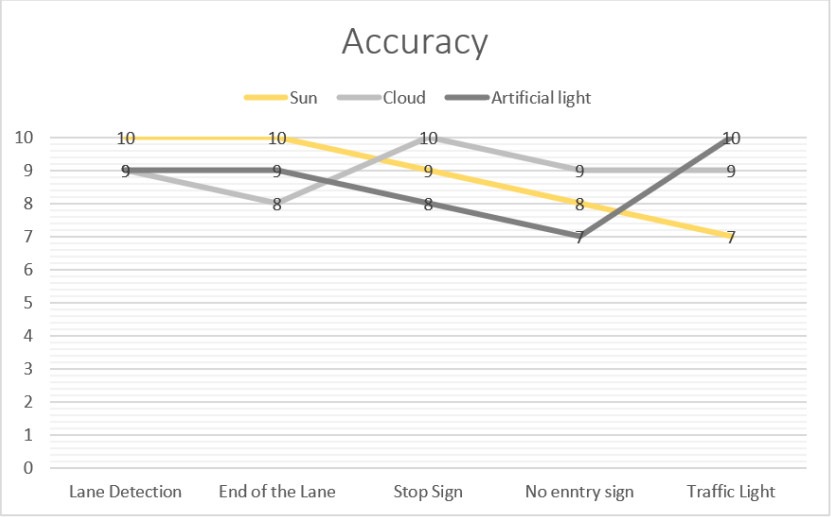

Figure 13. System accuracy results

Errors occur mainly due to the noises caused by light, or the similarity of the traffic sign shapes in addition to the speed and the direction of the vehicle. Therefore, the error can be reduced by increasing the positive and negative samples of the Haar cascading classifier or by using a higher-quality camera designed to capture images in low light conditions. The light reflected from the traffic signs can also affect the results. Hence, the traffic light could be better detected in lower light conditions, however, because of the reflection of the sun the error increased.

8. Conclusion

The study aimed to create a fully autonomous car that can detect the lane and can detect some traffic signs. The goals could be achieved by using a Raspberry Pi for image processing and machine learning. The system could detect the lane and respond to the change of direction of the lane and detect the traffic signs and give responses accordingly as shown in 'Figure 14'. Responses were sent to the Arduino UNO as binary signals. Then, these signals were merged with the output signals of the ultrasonic sensor to deliver the final signals to the motor driver.

However, the system has some limitations, such as the entire process being easily affected by light intensity or weather conditions, resulting in image processing errors. The computing unit of the system is relatively weak compared with other more expensive computing units.

Further improvements are possible by using faster computing units such as NVIDIA Jetson Nano [40] which has a dedicated graphics card and a bigger RAM size [40] as well as a better camera and Python with TensorFlow [41] and NumPy [42] libraries, allowing for even more enhancements. Mapping the location with a GPS is also a possible enhancement.
The system can be implemented in several applications e.g., it can be applied to a fully autonomous car, Parking assistance, or a driver assist system. It can be used for self-driving trucks to transport goods between cities on private highways, as well as for moving products within warehouses by defining roadways for small robots.

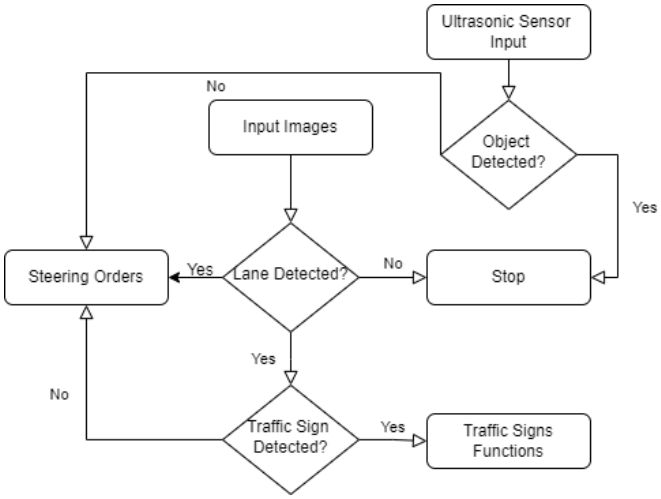

Figure 14. System working algorithm

9. List of abbreviations

MIPI CSI-2: Mobile Industry Processor Interface, Camera Serial Interface 2
PWM: Pulse Width modulation
CNN: Convolutional Neural Network
SVM: Support Vector Machine
NHTSA: The National Highway Traffic Safety Administration
SAE: the Society of Automotive Engineers
HAHT: Hierarchical Additive Hough Transformation
YOLO: you only look once
RGB: Red, Green, Blue
BGR: Blue, Green, Red
GPIO: General purpose Input/Output
I/O: Input/Output
TTL: Transistor-Transistor Logic
MSB: Most Significant bit
LSB: Least Significant bit
RAM: Random Access Memory
GPS: Global Positioning System

References

[1] A. Forrest and M. Konca, "Autonomous Cars and Society," p. 54.
[2] "What Percentage of Car Accidents Are Caused by Human Error? | Pittsburgh Law Blog." https://www.cbmclaw.com/what-percentage-of-car-accidents-are-caused-by-human-error/ (accessed Jun. 01, 2022).
[3] "Driving Statistics: The Ultimate List of Car Accident Statistics [2022]." https://driving-tests.org/driving-statistics/ (accessed Jun. 01, 2022).

[4] C. Filiz, “Can Autonomous Vehicles Prevent Traffic Accidents?,” in *Accident Analysis and Prevention*, M. Darçın, Ed. IntechOpen, 2020. doi: 10.5772/intechopen.93020.
[5] Q. Zhang, T. Du, and C. Tian, “Self-driving scale car trained by Deep reinforcement Learning,” p. 6.
[6] “Europe’s trucker shortage becoming ‘extremely dangerous’ | Financial Times.” https://www.ft.com/content/e8ca2a08-308c-4324-8ed2-d788b074aa6c (accessed Jun. 01, 2022).
[7] “This Year, Autonomous Trucks Will Take to the Road With No One on Board - IEEE Spectrum.” https://spectrum.ieee.org/this-year-autonomous-trucks-will-take-to-the-road-with-no-one-on-board (accessed Jun. 01, 2022).
[8] “SAE International.” https://www.sae.org/ (accessed Jun. 01, 2022).
[9] T. Inagaki and T. B. Sheridan, “A critique of the SAE conditional driving automation definition, and analyses of options for improvement,” *Cogn Tech Work*, vol. 21, no. 4, pp. 569–578, Nov. 2019, doi: 10.1007/s10111-018-0471-5.
[10] “MIPI-CSI2 Camera Sensors - TechNexion.” https://www.technexion.com/products/embedded-vision/camera-sensors/ (accessed Jun. 01, 2022).
[11] I. Ciganović, A. Pluškoski, and M. D. Jovanović, “Autonomous car driving - one possible implementation using machine learning algorithm,” p. 7.
[12] Chaocheng Li, Jun Wang, Xiaonian Wang, and Yihuan Zhang, “A model based path planning algorithm for self-driving cars in dynamic environment,” in *2015 Chinese Automation Congress (CAC)*, Wuhan, China, Nov. 2015, pp. 1123–1128. doi: 10.1109/CAC.2015.7382666.
[13] S. OwaisAli Chishti, S. Riaz, M. BilalZaib, and M. Nauman, “Self-Driving Cars Using CNN and Q-Learning,” in *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, Karachi, Nov. 2018, pp. 1–7. doi: 10.1109/INMIC.2018.8595684.
[14] Q. Memon, M. Ahmed, S. Ali, A. R. Memon, and W. Shah, “Self-driving and driver relaxing vehicle,” in *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, Rawalpindi, Pakistan, Nov. 2016, pp. 170–174. doi: 10.1109/ICRAI.2016.7791248.
[15] “SVM | Support Vector Machine Algorithm in Machine Learning.” https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ (accessed Jun. 01, 2022).
[16] M. V. G. Aziz, A. S. Prihatmanto, and H. Hindersah, “Implementation of lane detection algorithm for self-driving car on toll road cipularang using Python language,” in *2017 4th International Conference on Electric Vehicular Technology (ICEVT)*, Sanur, Oct. 2017, pp. 144–148. doi: 10.1109/ICEVT.2017.8323550.
[17] R. K. Satzoda, S. Sathyanarayana, T. Srikanthan, and S. Sathyanarayana, “Hierarchical Additive Hough Transform for Lane Detection,” *IEEE Embedded Syst. Lett.*, vol. 2, no. 2, pp. 23–26, Jun. 2010, doi: 10.1109/LES.2010.2051412.
[18] B. T. Nugraha, S.-F. Su, and Fahmizal, “Towards self-driving car using convolutional neural network and road lane detector,” in *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro--Mechanical System, and Information Technology (ICACOMIT)*, Jakarta, Oct. 2017, pp. 65–69. doi: 10.1109/ICACOMIT.2017.8253388.
[19] “Autonomous Driving – Car detection with YOLO Model with Keras in Python | sandipanweb.” https://sandipanweb.wordpress.com/2018/03/11/autonomous-driving-car-detection-with-yolo-in-python/ (accessed Jun. 01, 2022).
[20] “Self-Driving Cars With Convolutional Neural Networks (CNN) - neptune.ai.” https://neptune.ai/blog/self-driving-cars-with-convolutional-neural-networks-cnn (accessed Jun. 01, 2022).
[21] P. M. Daigavane and P. R. Bajaj, “Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization,” in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, Goa, Nov. 2010, pp. 76–80. doi: 10.1109/ICETET.2010.128.
[22] I. Oršolic, “BUILDING A SELF-DRIVING RC CAR,” p. 125.

[23] D. A. Alghmgham, G. Latif, J. Alghazo, and L. Alzubaidi, "Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN," *Procedia Computer Science*, vol. 163, pp. 266–274, 2019, doi: 10.1016/j.procs.2019.12.108.
[24] Y. Lai, N. Wang, Y. Yang, and L. Lin, "Traffic Signs Recognition and Classification based on Deep FeatureLearning:," in *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods*, Funchal, Madeira, Portugal, 2018, pp. 622–629. doi: 10.5220/0006718806220629.
[25] A. Mohamed, A. Issam, B. Mohamed, and B. Abdellatif, "Real-time Detection of Vehicles Using the Haar-like Features and Artificial Neuron Networks," *Procedia Computer Science*, vol. 73, pp. 24–31, 2015, doi: 10.1016/j.procs.2015.12.044.
[26] A. Kaehler and G. R. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*, First edition, Second release. Sebastopol, CA: O'Reilly Media, 2017.
[27] T. Wildi, *Electrical machines, drives, and power systems*, 6th ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2006.
[28] J. Cicolani, *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV*, 1st ed. 2018. Berkeley, CA: Apress : Imprint: Apress, 2018. doi: 10.1007/978-1-4842-3462-4.
[29] R. Szeliski, *Computer vision: algorithms and applications*. London ; New York: Springer, 2011.
[30] *Foundations of computer vision*. New York, NY: Springer Berlin Heidelberg, 2017.
[31] "Python | Thresholding techniques using OpenCV | Set-1 (Simple Thresholding) - GeeksforGeeks." https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/ (accessed Jun. 01, 2022).
[32] "Feature Detectors - Canny Edge Detector." https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm (accessed Jun. 01, 2022).
[33] H. Singh, *Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python*, 1st edition. New York: APRESS, 2019.
[34] "WiringPi." http://wiringpi.com/ (accessed Jun. 01, 2022).
[35] "Raspberry Pi 3 Pinout, Features, Specifications & Datasheet." https://components101.com/microcontrollers/raspberry-pi-3-pinout-features-datasheet (accessed Jun. 01, 2022).
[36] "Raspberry Pi 3 B+ Pinout with GPIO functions, Schematic and Specs in detail." https://www.etechnophiles.com/raspberry-pi-3-b-pinout-with-gpio-functions-schematic-and-specs-in-detail/ (accessed Jun. 01, 2022).
[37] "Arduino Uno Specification – TOMSON ELECTRONICS." https://www.tomsonelectronics.com/blogs/news/arduino-uno-specification (accessed Jun. 01, 2022).
[38] "Full-Bridge Motor Driver Dual - L298N." https://www.canakit.com/full-bridge-motor-driver-dual-l298n-com-09479.html (accessed Jun. 01, 2022).
[39] "Cascade Trainer GUI - Amin." https://amin-ahmadi.com/cascade-trainer-gui/ (accessed Jun. 01, 2022).
[40] "Jetson Nano vs Raspberry Pi 4: The Differences | All3DP." https://all3dp.com/2/raspberry-pi-vs-jetson-nano-differences/ (accessed Jun. 01, 2022).
[41] "Introduction to TensorFlow." https://www.tensorflow.org/learn (accessed Jun. 01, 2022).
[42] "NumPy." https://numpy.org/ (accessed Jun. 01, 2022).