International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 04 | Apr 2023 www.irjet.net p-ISSN: 2395-0072

Designing Autonomous Car using OpenCV and Machine Learning

R. Hemanth Kumar[1], T. Varun Sai Krishna[2], S. Durga Prasad[3], P. Varuna Sai[4]

[1,2,3,4] *Students, Dept. of Electronics & Communication Engineering, R.V.R. & J.C. College of Engineering, Guntur, Andhra Pradesh, India*

Abstract – *Self-driving cars, also known as autonomous vehicles, use sensors to perceive the environment and control the car's different actuators without human intervention. This project aims to design a self-driving car using two modules: lane detection and traffic signal detection. Lane detection is done using image processing with OpenCV, while traffic signal detection is done by using machine learning. The prototype of the self-driving car is built using raspberry pi as master device and Arduino as the slave device to capture and process the data and control the car's movements. The raspberry pi camera interface captures every frame of the environment for lane and traffic signal detection. The car makes the right decisions by referencing the lane, and the Arduino controls the orientation of the wheels and speed using motors. This project is inspired by recent surge in the automated car industry, and it shows the potential of self-driving cars in future.*

Key Words: **Autonomous car, Lane detection, traffic signal detection, Raspberry pi, Arduino, OpenCV, Machine learning.**

1. INTRODUCTION

Road accidents claim 1.25 million lives annually worldwide, causing injuries that lead to disabilities in 20-50 million people. India alone accounts for 10% of global road accidents, with the highest number of road fatalities. Human error is responsible for 78% of accidents, but technology, such as driverless cars, can help address this issue. While human error can never be completely eliminated, technology has made significant advancements in recent years, from radar-based collision detection to more advanced forms of technology. As a result, accidents can be prevented, and the use of autonomous vehicles may help reduce road fatalities in the future.

Self-driving cars have been a subject of growing interest in recent years, as technology continues to advance and people seek safer and more efficient means of transportation. These autonomous vehicles are designed to drive without human intervention, relying on sensors, software, and algorithms to interact with their surroundings.

We present our project, which aims to design and develop a self-driving car prototype using OpenCV and machine learning. Our objective is to create a car that can detect lanes and traffic signals in real-time, utilizing computer vision and machine learning techniques.

OpenCV is an open-source computer vision library that provides tools and algorithms for image processing, feature detection, and object recognition. We utilize the power of OpenCV to detect lanes and traffic signals from the frame feed captured by a Raspberry Pi camera. Additionally, machine learning algorithms are used for traffic signal and stop sign detection.

Our project involves two primary modules: lane detection and traffic signal detection. Lane detection is performed by OpenCV, which processes the frames from the camera and detects the road's edges. The detected lane markers are then used to control the car's steering and speed with the aid of motors connected to an Arduino board.

Traffic signal detection is performed using machine learning algorithms. The training dataset consists of traffic signal images captured from different angles and in various lighting conditions. The model is trained to recognize the various traffic signal types and their corresponding meanings.

Our self-driving car prototype is built using a Raspberry Pi as the master device and an Arduino board as the slave device. The Raspberry Pi captures frame frames from the camera and processes them using the OpenCV and machine learning modules. The Arduino board is responsible for controlling the car's wheel orientation and speed.

We evaluated and analyzed the performance of our self-driving car prototype by testing it under different lighting conditions to simulate real-world scenarios. The car accurately detected lanes and traffic signals and responded accordingly.

Our research paper presents a novel approach to designing a self-driving car prototype using OpenCV and machine learning. Our prototype demonstrates the potential of autonomous cars in the future of transportation. The combination of OpenCV and machine learning offers a powerful set of tools for image processing and AI algorithms that can be used to develop more advanced self-driving car models in the future.

2. HARDWARE AND SOFTWARE USED

For this project, various hardware components are used such as Raspberry Pi, Pi camera, Arduino, and L298N H-bridge, along with software tools like OpenCV, Arduino IDE,

and Cascade trainer GUI. Here's a brief overview of each hardware and software component used in the project.

2.1 Raspberry pi

The Raspberry Pi is a small yet powerful single-board computer available at a low cost. It is equipped with a processor speed ranging from 700 MHz to 1.2 GHz for the Pi 3 model. The onboard memory capacity ranges from 256 MB to 1 GB RAM. The board has support for up to 4 USB ports and an HDMI port. Additionally, it has several GPIO (General Purpose Input Output) pins that support protocols like I^2C. The Raspberry Pi supports both Wi-Fi and Bluetooth connectivity, making it highly compatible with other devices. It is also versatile, supporting programming languages like Scratch and Python. Several operating systems are compatible with Raspberry Pi hardware, such as Ubuntu MATE, Snappy Ubuntu, and more. Raspbian is specifically designed to support Raspberry Pi's hardware and is a popular choice for projects involving the device[1][2].

Fig -1: Raspberry PI 3B+

2.2 Pi Camera

The Pi camera is a small and powerful device that can capture high-quality images and videos. It has a compact size of 25mm x 24mm x 9mm and connects to the Raspberry Pi through a flexible serial interface. With a 5-megapixel image sensor and focused lens, it can provide excellent video clarity, making it useful for time-lapse and slow-motion footage. Additionally, the camera is ideal for security purposes due to its wide image support and ability to take stills with a resolution of up to 2592x1944 and 1080p30 video on Camera module v1[3].

Fig -2: Raspberry PI Camera

2.3 Arduino

The microcontroller used in the project is based on ATmega329P and offers 14 digital input/output pins, of which 6 can be used as PWM outputs, and 6 analog inputs. It also features a 16 MHz quartz crystal, USB connection, power jack, ICSP header, and a reset button. The device comes with 32 kb of flash memory, and 2 kb of SRAM, and weighs approximately 25g. Additionally, the Arduino IDE has a user-friendly interface and utilizes the basic C programming language for coding[4].

Fig -3: Arduino UNO

2.4 L298N Motor Driver

The LN293D IC serves as a motor driver that connects the car's motors to the Arduino. By receiving signals from the Arduino, it can initiate or halt the motion of the motors in response to those signals.

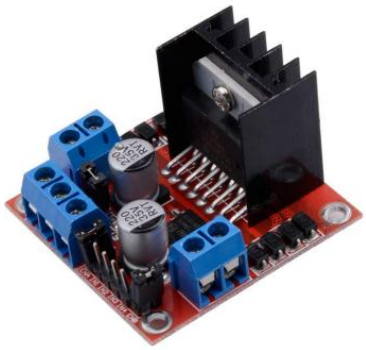

Fig -4: L298N H-Bridge Motor Driver

2.5 OpenCV

OpenCV (Open-Source Computer Vision) is an open-source library for computer vision that is widely used for real-time computer vision and machine learning applications. It offers an extensive range of powerful algorithms for computer vision tasks such as object detection, image processing, and face recognition. OpenCV has been significantly utilized in autonomous vehicles for

various purposes such as pedestrian detection, lane detection, and object detection. With machine learning algorithms, OpenCV can learn to recognize and react to different objects and circumstances on the road, making it an essential element in the autonomous vehicle development[5].

2.6 Arduino IDE

The Arduino IDE is a programming platform used to write code for the Arduino board. It includes a compile button that compiles the code and an upload tab that uploads the code onto the board. Programs written on the Arduino IDE are often called sketches and are saved with a .ino extension. The editor has various other features such as verify, save, and upload, include library, and serial monitor. The developers have created easy-to-use functions that make coding both easy and enjoyable. Additionally, there are numerous examples provided for each interface, allowing users to learn more about functions and hardware.

2.7 Cascade Trainer GUI

Cascade Trainer GUI is a user-friendly software with a graphical interface that is designed for training models for object detection. It is built on the OpenCV library and allows users to create cascade classifiers for various tasks, including face detection, license plate recognition, and more. The GUI makes it easy to select positive and negative images, customize training parameters, and evaluate classifier accuracy. Researchers and developers seeking custom object detection models for computer vision applications can benefit from Cascade Trainer GUI.

Cascade Trainer GUI uses Haar Cascade Classifier, an ML-based method, to identify objects in images or videos. Positive and negative images train the classifier to detect objects based on their features.

3. PROPOSED METHODOLOGY

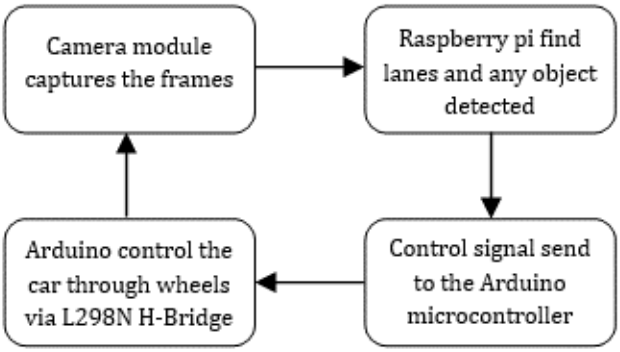

Fig -5: Proposed Self-driving model

The proposed model for autonomous cars relies on a Raspberry Pi connected to a Pi cam for capturing images. The captured image is sent to a laptop over the same network for processing. The processed image results in one of four outputs: left, right, forward, or stop. Upon announcement of the output, the corresponding signal is sent to the Arduino, enabling the car to move in a specific direction via motors. This model is an efficient and effective way to control the movement of the autonomous car using computer vision and machine learning techniques.

3.1 Camera Setup

The "raspicam_Cv.h" header file was utilized to access the Raspberry Pi camera and adjust frame parameters such as height and width. Image quality was enhanced by modifying the brightness, contrast, and saturation of the frame, and the required number of frames was set for the project's purposes. These actions allowed for obtaining real-time camera data in the project.

3.2 Region of Interest

To enable the vehicle to move between lanes on a road, we extracted the lanes by creating a region of interest based on converging lines in the camera view. To transform the lane lines into parallel lines, we used a bird's eye view approach by defining a trapezium shape over the lanes and transforming it into a rectangular view[6]. This technique improved lane detection and allowed for more precise vehicle movement on the road.

3.3 Lane center

The captured frame was divided into two equal parts to determine the left and right lanes. The bottom 1/3rd of the frame was grouped, and the maximum available pixel values were selected. The indexes of the maximum pixel values in a row were treated as the left and right lanes in their respective frames. We calculated the distance between the two obtained lanes and found the midpoint, which was

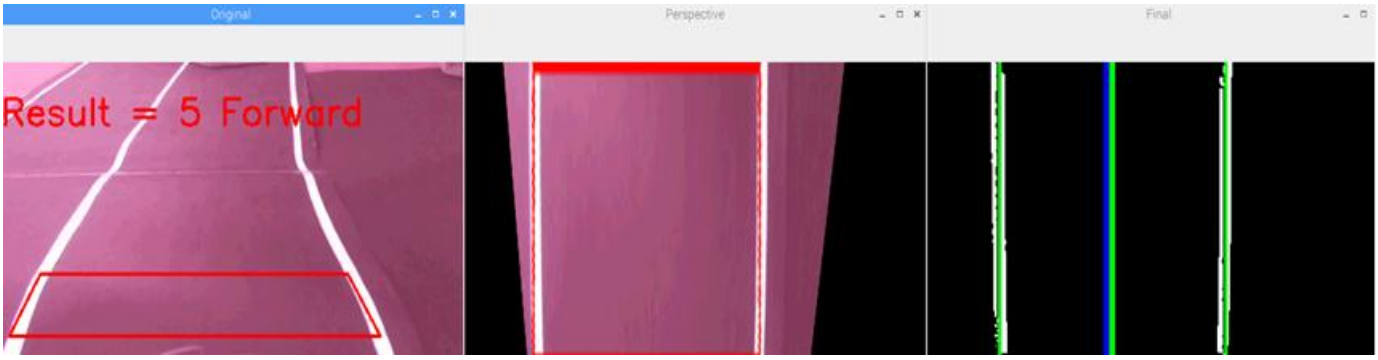

Fig -6: Image processed frames

considered the lane center. The car reference point was set as the frame center, and it was used to navigate the car in the correct direction between the lanes on the road. The midpoint was used to decide whether to steer the car to the left or right to maintain its position within the lane.

3.4 Master and slave device communication

Commands are sent from the Raspberry Pi to the Arduino based on the distance between the lane center and frame center, triggering actions such as turning left, right, or moving forward. The Arduino IDE was used to design the left, right, and forward functions, while the speed of the motors was controlled using PWM signals. However, the output from the Arduino was not powerful enough to drive the motors, so we used an L298N motor driver, which was connected to an external power supply. The driver received the control signals from the Arduino, allowing the car to move forward or turn left or right as needed.

3.5 Object detection

To detect stop signs and traffic signals, machine learning techniques were applied using the Cascade Trainer GUI tool. This tool enabled the creation of custom object detection classifiers without the need for coding. The tool provided adjustable parameters like the number of stages, minimum hit rate, and maximum false alarm rate.

3.6 HAAR cascade

HAAR cascades detect patterns of pixel intensity in an image via HAAR features - simple rectangular features that capture variations in image intensity. For a HAAR cascade classifier to be trained, positive images of the object and negative images of the background without the object were used. During training, many HAAR features were generated and the best ones were selected to differentiate between positive and negative examples. Upon completion, the HAAR cascade trainer produced a .xml file that could be integrated with OpenCV using predefined functions to enable object detection.

3.7 Environment setup

The car was designed on a small scale, and for that purpose, the roads, stop signs, and traffic signals were custom-made. The road was designed in black with white lanes to achieve the maximum variation in pixel values when converted to grayscale. Moreover, the stop signs and traffic signals were made small in size and were designed to fit the camera angle and height of the car to ensure accurate detection. These customized designs were critical in achieving high precision in object detection and providing the car with an optimal environment for testing and development.

4. RESULTS

The lane detection accurately detected the lanes on the road in various lighting conditions. This process was able to detect straight and curved lanes and provide real-time feedback on the car's position relative to the lane.

The fig 6 illustrates the lane detection process with three output frames. The leftmost frame displays the captured image from the Raspberry Pi camera with a region of interest outlined. The center frame shows a perspective view of the road, resembling a bird's eye view. The rightmost frame displays the detected lanes, with the lane center and frame center correctly marked. The frames demonstrate the ability to accurately identify the lanes and provide real-time feedback to control the car's movements.

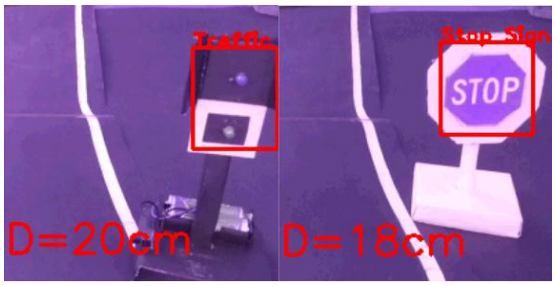

Fig -7: Traffic signal and Stop sign detection

The fig 7 demonstrates the impressive object detection capabilities of the autonomous car system, including the identification of stop signs and traffic signals. In the leftmost frame, a bold red rectangle signals the detection of a traffic signal in the captured image, while the rightmost frame displays a striking red rectangle when a stop sign is detected. Both frames also display the distance between the car and the detected object in centimeters, allowing for effective navigation. The stop sign detection is particularly impressive, with a highly accurate distance estimate. Additionally, the traffic signal detection algorithm is incredibly reliable, only activating when the red light is illuminated, ensuring that the car stops only when necessary for safety.

The autonomous car designed using OpenCV and machine learning successfully navigated a pre-determined path with high accuracy and safety. It could detect and respond to changes in the environment like lane markings, traffic signals, and stop signs. The results showcase the potential of using these technologies for designing autonomous cars.

5. CONCLUSION

In this paper, a method to make a model of a self-driving car is presented. The different hardware components along with software and neural network configuration are clearly described. With the help of Image Processing and Machine Learning, a successful model was developed which worked as per expectation.

Aside from transportation, self-driving cars can also have significant benefits in the automation industry. They could be used for patrolling and capturing images, reducing accidents caused by the carelessness of goods carrier vehicles, and ensuring better logistic flow. Buses for public transport would be more regulated due to minimal errors. Hence, due to its greater autonomous nature and efficiency, an autonomous car of this nature can be practical and is highly beneficial for better regulation in the goods and people mover's section.

Further research in this area could focus on enhancing the car's capabilities, such as adding additional modules for obstacle detection, pedestrian detection, and GPS mapping. Investigating the car's performance in different weather and lighting conditions could also be explored. Ultimately, the potential impact of this technology is significant, and it is expected to transform the transportation industry in the coming years.

Overall, the use of machine learning and image processing can help self-driving cars detect and respond to different objects in the environment, which is critical for ensuring safe navigation. With the help of these technologies, successful models have been developed, implemented, and tested. Further research and development in this area hold immense potential for revolutionizing the transportation and automation industries.

REFERENCES

[1] Matt Richardson, Shawn Wallace, Getting Started with Raspberry Pi, 2nd Edition, Published by Maker Media, Inc., USA, 2014. Book ISBN: 978-1-457-18612-7.

[2] https://www.hongkiat.com/blog/pi-operating-systems/

[3] https://www.raspberrypi.org/documentation/hardware/camera/

[4] https://www.arduino.cc/en/Reference/Board?from=Guide.Board

[5] Gary Bradski, Adrian Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, "O'Reilly Media, Inc.". Copyright, September 2008, 1st Edition, Book ISBN: 978-0-596-51613-0.

[6] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. G. Sukhatme, S. Schaal, W. Burgard, and D. Fox, editors& Proceedings of the Robotics Science and Systems Conference, Philadelphia, PA, 2006.

[7] S Swetha, Dr. P Sivakumar, "SSLA Based Traffic Sign and Lane Detection for Autonomous cars", 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) ©2021 IEEE, April 2021.

[8] G. S. Pannu, M. D. Ansari, and P. Gupta, "Article: Design and implementation of autonomous car using raspberry pi", International Journal of Computer Applications, vol. 113, no. 9, pp. 22–29, March 2021.

[9] Yue Wanga, Eam Khwang Teoha & Dinggang Shenb, Lane detection and tracking using B-Snake, Image and Vision Computing 22 (2004) , available at:www.elseviercomputerscience.com, pp. 269–280.

[10] Joel C. McCall &Mohan M. Trivedi, Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation, IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 1, March 2006, pp. 20-37.

[11] Xiaodong Miao, Shunming Li & Huan Shen, On-Board lane detection system for intelligent vehicle based on monocular vision, International Journal on Smart Sensing and Intelligent Systems, vol. 5, no. 4, December 2012, pp. 957-972.

[12] International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)

Vol. 5, Issue 4, April 2018 Prototype of Autonomous Car Using Raspberry Pi.

[13] Design and Implementation of Autonomous Car using Raspberry Pi International Journal of Computer Applications (0975 – 8887) Volume 113 – No. 9, March 2015.

[14] Pratibha I Golabhavi , B. P. Harish, 2020, Self-Driving Car Model using Raspberry Pi, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 02 (February 2020).

[15] Self-Driving Car using Raspberry-Pi and Machine Learning International Research Journal of Engineering and Technology (IRJET) Volume 06, Issue 03, March 2019.