Journal of IoT in Social, Mobile, Analytics, and Cloud (ISSN: 2582-1369)
www.irojournals.com/iroismac/

Self-Driving Car Using Image Processing

Abhishek Kumar Shukla[1], Dhananjay Kumar Yadav[2], Prachi Singh Rajput[3], Sittal Bhusal[4], Suraj Basant Tulachan[5]

[1-4]Students, [5]Assistant Professor, Department of Electronics and Computer Engineering, Pashchimanchal Campus, Pokhara, Nepal.

E-mail: [1]avshekshukla960@gmail.com, [2]yadavdhananjay143@gmail.com, [3]prachis9811@gmail.com, [4]sittalbhusal33@gmail.com, [5]sbtula@gmail.com

Abstract

Autonomous cars represent a new mode of transportation using hardware and software to facilitate autonomous driving without human intervention. This is a design document for the creation of a low-cost autonomous vehicle prototype based on the Raspberry Pi 3 as the core processor. The system combines lane detection, traffic light identification, and traffic sign recognition into a single multi-threaded pipeline that facilitates real-time multi-task perception. Experimental testing demonstrates that the prototype attains a mean accuracy of 87%, a processing rate of 13–14 frames per second, and a mean decision time of approximately 105 ms in outdoor and indoor environments. In contrast to earlier Raspberry Pi-based prototypes that executed tasks sequentially or merely followed lanes, the system exhibits the novelty of stable multi-task perception with limited resources. The findings indicate the potential and limitations of embedded systems in autonomous navigation, with possible future applications in transportation, industrial automation, and surveillance.

Keywords: Self-Driving Car, Raspberry Pi, Embedded Systems, Lane Detection, Traffic Sign and Light Recognition.

1. Introduction

1.1 Background

Autonomous driving is becoming one of the most important applications of artificial intelligence, computer vision, and robotics. Present-day autonomous cars built by organizations like Tesla, Waymo, and Baidu depend on high-performance GPUs, LiDAR, and

DOI: https://doi.org/10.36548/jismac.2025.3.004
Received: 15.08.2025, received in revised form: 13.09.2025, accepted: 28.09.2025, published: 07.10.2025
© 2025 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

radar to provide state-of-the-art performance. Although these types of systems are very accurate and reliable, their expense and computationally costly nature make them unsuitable for low-budget prototyping, research, and education.

Low-cost boards such as Arduino and Raspberry Pi have gained popularity in recent times for hobby and student project usage. These boards are occasionally used to test lane following, obstacle detection, or traffic sign recognition individually. The majority of existing prototypes are, however, task-specific and narrow in use, typically employing sequential processing pipelines that limit real-time capability and lane tracking [8] or obstacle avoidance [12]. The majority of the previous research also provides qualitative representation but doesn't quantitatively compare accuracy, speed, and system efficiency, all of which are critical for comprehensive research verification.

This work meets such challenges through the development of a multitask-perceiving Raspberry Pi-based autonomous prototype that integrates traffic sign and light perception, lane detection, and more into a unified multi-threaded execution pipeline. The work demonstrates how multitask-perceiving low-resource platforms can be used to drive further autonomous navigation research and provide cost-saving solutions for low-cost deployment based on affordability, parallel processing, and quantitative assessment.

1.2 Problem Statement

Despite how far autonomous driving research has come, most existing prototypes and commercial systems are based on high-end hardware, including GPUs, LiDAR, and radar, making them very costly and not feasible for small-scale research or educational purposes. While cheap options based on Raspberry Pi and Arduino have been promising in applications such as traffic sign recognition, obstacle detection, and lane recognition, these systems suffer from a number of major disadvantages:

- They are generally limited to isolated tasks, like lane following or obstacle detection, and not to integrated multi-task perception.
- All but the most advanced implementations use sequential processing, which limits responsiveness and frame rate and restricts their use in real time.

- Existing studies have often concentrated on qualitative demonstrations in the form of flow diagrams or screenshots but have provided little more than a limited quantitative evaluation of system accuracy, processing time, and decision delay.
- Little focus has been given to scalability and safety issues in scaling down low-cost prototypes into advanced autonomous platforms.

Consequently, a low-cost self-driving car prototype is required that offers quantitative performance evaluation in addition to integrating multiple perception tasks on embedded hardware. Solving this issue will aid in bridging the gap between realistic, affordable autonomous driving solutions and expensive, resource-intensive systems.

1.3 Objectives

The main objectives of the "Self-Driving Car Using Image Processing" paper are:

- To develop an integrated self-driving system using Raspberry Pi 3 and Arduino Nano for lane detection, traffic light recognition, and traffic sign recognition.
- To implement a multi-threaded execution pipeline that enables parallel perception tasks, improving system responsiveness compared to sequential execution.

2. Related Work

Studies on autonomous driving have concentrated more on integrating computer vision, robotics, and embedded systems to attain real-time decision-making ability. Previous studies, like Viswanathan and Hussein [1], analyzed core image-processing methods for traffic and lane feature detection through pre-processing, Canny edge detection, and Hough transforms to identify road edges. OpenCV has also been employed extensively for the estimation of distance and lane detection to provide safe navigation and avoid collision [3]. Likewise, through the exploration of ultrasonic-based obstacle avoidance methods, Borenstein and Koren [2] laid the groundwork for sensor-augmented driving systems. These studies laid down foundational methods for the detection of lanes and obstacles, even though they were narrow in scope and lacking in computation.

Subsequent work employed Raspberry Pi and Arduino microcontrollers to construct low-budget prototypes that would be able to conduct basic obstacle avoidance, traffic light

recognition, and lane detection [6], [7]. Shahane [7], for example, combined the Raspberry Pi and Arduino to achieve real-time lane following, while Pawar and Patil [6] designed a driver assistance system on the Raspberry Pi. Though these systems proved successful in validating proof-of-concept models, they were not scalable to more realistic traffic environments and their performance was primarily constrained to controlled conditions.

Recent developments have extended the limits of autonomous driving on low-end hardware. A Raspberry Pi-based lane-following vehicle was implemented by Isherwood and Secco [8], with a performance of around 20 Hz but only single-task perception. With inspiration drawn from NVIDIA's DAVE-2 model, DeepPicar [9] demonstrated that a Raspberry Pi 3 could support a CNN-based end-to-end controller operating at up to 100 Hz frequency, though again it did not multitask but only handled steering. While Kumar et al. [11] integrated Raspberry Pi with Arduino and CAN bus for distributed control of sensors, Ravindran et al. [10] designed Raspberry Pi-based delivery vehicles using computer vision for navigation. Likewise, Wagdy et al. [13] developed a parallel parking assistant based on a Raspberry Pi, and Golabhavi and Harish [12] proposed a system based on Raspberry Pi with sonar and GPS for the avoidance of accidents. These works point to increasing interest in low-cost embedded autonomy but are constrained to particular tasks or sensor fusion as opposed to integrated perception pipelines.

Meanwhile, panoptic driving perception was exemplified by top-performing models such as YOLOP [14], which detected lanes, drivable regions, and objects in real time simultaneously. While such systems generate state-of-the-art results, the need for GPU-accelerated hardware, such as the NVIDIA Jetson TX2, constrains the applicability of these systems to research and teaching environments with limited budgets.

By developing a prototype from a Raspberry Pi 3, the present work bridges this gap by supporting real-time multi-task perception integrating lane detection, traffic light recognition, and traffic sign recognition into a unified system. Our implementation brings a multi-threaded execution pipeline that allows parallel perception operations, enhancing responsiveness and maintaining an average 13–14 FPS frame rate in daylight and indoor conditions, compared to earlier Raspberry Pi–based solutions that ran the tasks sequentially or in mutual exclusion [8], [12]. This work provides an economical platform for both experimental work and applications

in reality by demonstrating that reliable autonomous navigation is feasible even on resource-constrained hardware.

3. Proposed Work

Lane detection, traffic signal detection, traffic sign detection, and navigation control are the four essential modules that have been used in the proposed prototype. For real-time computation, a multi-threaded processing pipeline was utilized for the execution of these modules on a Raspberry Pi 3 platform. Parallel execution is preferable in our approach, which improves responsiveness and frame rate, compared to previous Raspberry Pi–based prototypes, which processed perception tasks sequentially [6], [7].

3.1 Hardware and Software Selection

Raspberry Pi 3, chosen for its cost and processing power, formed the core of the system. An Arduino Nano took some of the low-level motor actuation load off the Pi, and additional parts included an L293D motor driver, ultrasonic sensors, and a Pi Camera Module. The Raspberry Pi was therefore free to concentrate on computationally demanding vision operations thanks to the division of labor.

Software development was done using Python, and the base image processing library employed was OpenCV. Nano programming was done using the Arduino IDE for embedded programming and VNC Viewer for remote debugging. Object detection was performed using Haar Cascade classifiers because they are lightweight and can be applied in real time on Raspberry Pi hardware. This option is the result of a conscious trade-off: while more contemporary deep-learning based solutions like YOLOP [14] or CNN-guided approaches like DeepPicar [9] are more accurate, they use GPU computations or hardware that are outside of this prototype's cost concerns for cost-effectiveness. Several preprocessing steps, such as greyscale conversion, smoothing through Gaussian filtering, and Canny edge detection, were applied to facilitate simpler lane detection.

3.2 Lane Detection Pipeline

Hough transforms were used to calculate lane edges, and processing was limited to the road surface with the application of a region-of-interest mask. The offset from the lane center was calculated in terms of the vehicle's midpoint so that safe driving could be ensured. This

offset was then fed into a Proportional-Integral-Derivative (PID) controller, which continuously adjusted the steering in order to keep the vehicle on course. To find edges, we used Canny edge detection with lower and upper thresholds of 50 and 150, respectively, and an aperture size of 3. Lane candidates were found by using the Probabilistic Hough Transform with $\rho = 1$ pixel, $\theta = 1°$ ($\pi/180$ rad), minLineLength = 40 pixels, and maxLineGap = 20 pixels.

The camera had a resolution of 640 × 480 pixels and a frame rate of 30 fps. The vehicle was controlled using a PID controller to provide responsiveness and stability in lane tracking with gains Kp = 0.585, Ki = 0.01, and Kd = 0.2.

Figure 1. Input Image with ROI Output

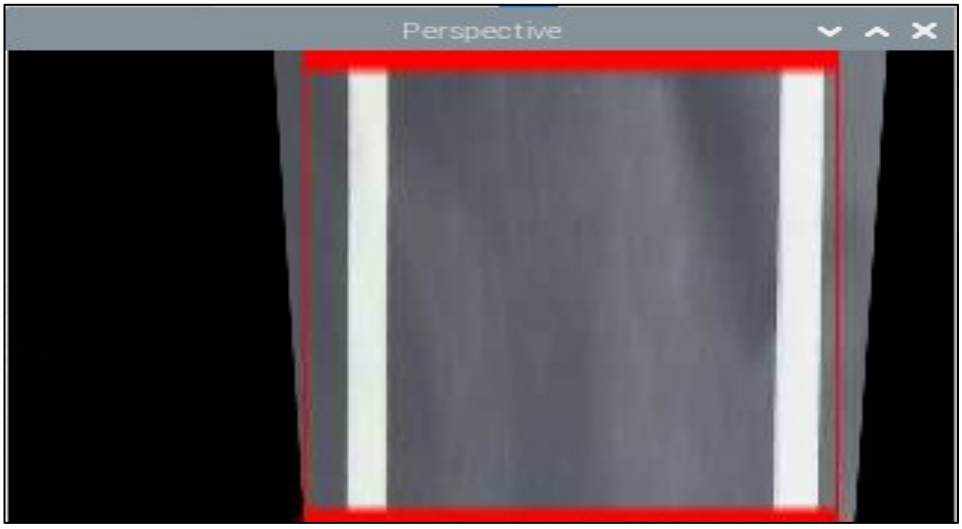

Figure 2. Perspective Transformation

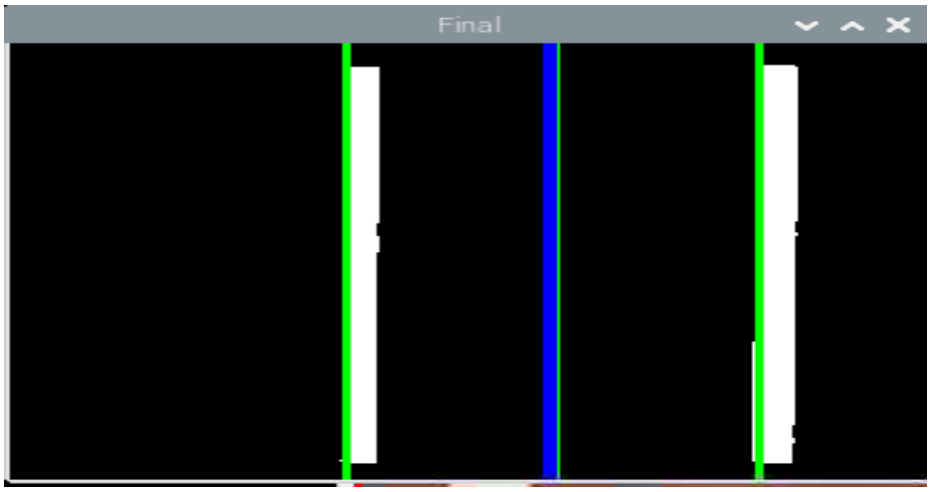

Figure 3. Final Output of Lane Detection

Figure 4: Output Showing the Offset When Car is Slightly Right

Figure 5. Output Showing the Offset When Car is Slightly Left

3.3 Traffic Light and Sign Detection

Road lights and traffic signs were identified using Haar Cascade classifiers trained on in-house datasets. N positive and M negative samples were acquired indoors and during the daytime. A standard cascade process of feature detection, boosting classifier, and sample preparation was applied during training.

During testing, the classifiers achieved the following detection rates:

Traffic sign recognition detection accuracy: 86% (indoors) and 89% (daylight).

- Accuracy of traffic light recognition: 83% (indoors) and 85% (daylight).
- Average per-frame detection latency: ~105 ms.

Due to their speed of computation and real-time inference feature on Raspberry Pi boards, Haar Cascades were specifically selected. While more accurate models like YOLOP [14] are available, their GPU requirements make them impractical to implement on low-resource embedded systems. Our findings suggest that, considering the Raspberry Pi's processing limitations, Haar Cascades are a suitable compromise between efficiency and accuracy to allow for accurate detection.

3.4 Parallel Processing

The system used a multi-threaded model of execution to operate in real time. While the main thread was responsible for image capture and communication with the Arduino, separate threads were assigned to lane detection, traffic light detection, and traffic sign detection. Compared to sequential execution, this design enhanced throughput by about 30%, while performance in tested environments increased from around 9 frames per second to around 13–14 frames per second. Prior Raspberry Pi-based systems executed operations one after another, decreasing the system's responsiveness [8], [12]. Parallelization in this case is a game-changer.

3.5 Navigation System

More abstract motions, e.g., forward, left, right, or stop, were determined by the navigation module depending on the perception output combination. The Arduino Nano received these commands through serial communication and used the L293D driver to provide current to the motors. PID-based control reduced oscillations and enhanced trajectory stability

by providing smooth transitions of steering for lane center offsets. This perception/control modularity offers a proof of concept and a scalable platform for future low-cost autonomous driving systems.

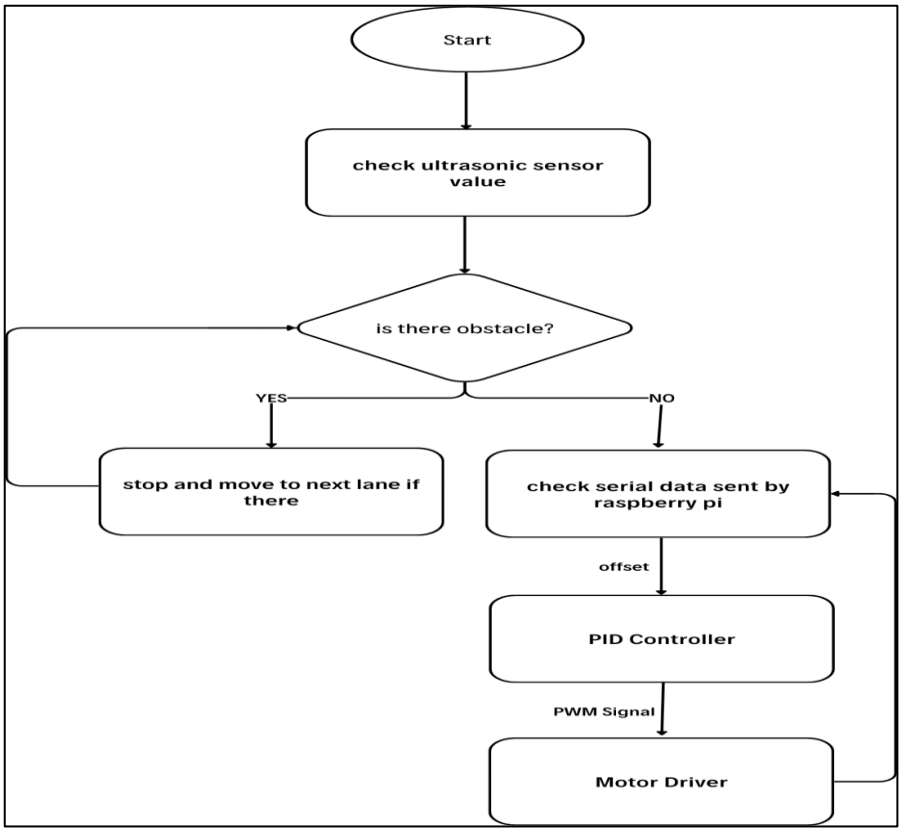

Figure 6. Block Diagram of Navigation System

Figure 7. Self Driving Car

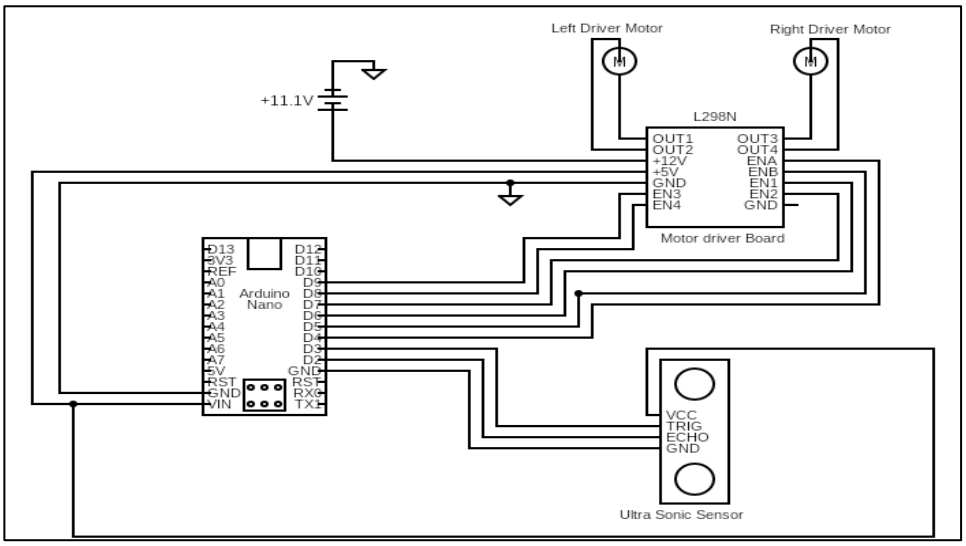

Figure 8. Schematic Diagram of Self Driving Car

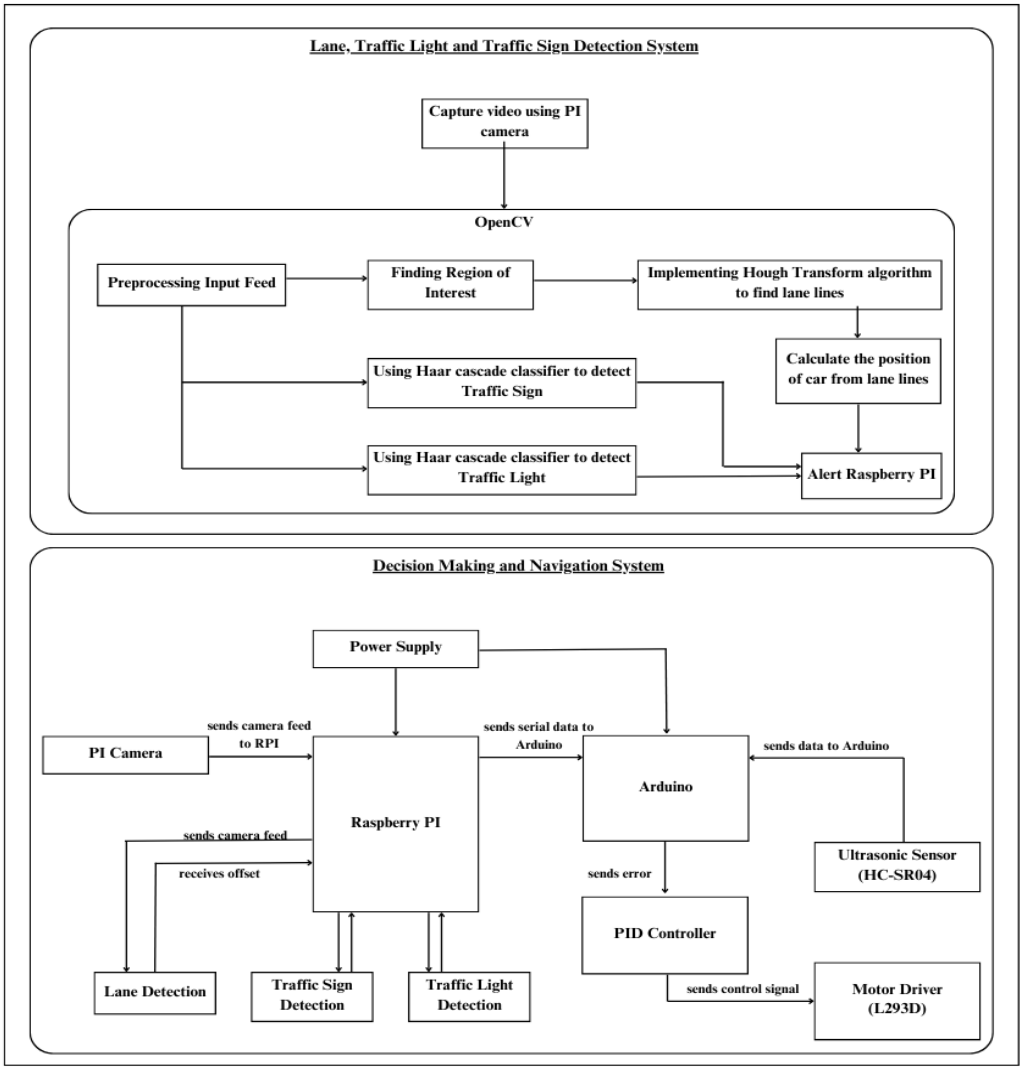

Figure 9. Overall System Architecture

4. Results

Experiments were conducted on a 5m * 3m test track with lanes, curves, obstacles, traffic light and traffic signs under controlled indoor conditions and daylight conditions.

Table 1. Experimental Results

Condition	Lane Detection Accuracy	Traffic Sign Detection Accuracy	Traffic Light Detection Accuracy	Avg. FPS
Daylight (outdoor road)	91%	86%	89%	13–14
Indoor (controlled track with artificial lighting)	88%	83%	85%	14

Overall, the system's average navigation success rate was 87% across all trials. Real-time response on Raspberry Pi hardware was achieved with an average latency of around 105 ms for each decision. Multi-threaded execution over sequential pipelines consistently increased frame rates by about 30%, confirming the benefits of the proposed design.

Performance under less-than-ideal lighting or bad weather was weak, though the prototype performed well in good lighting conditions. It was not possible to conduct experiments in rain, fog, or nighttime conditions due to hardware and dataset constraints. These are common issues among comparable Raspberry Pi-based systems [8], [12]. The experiments show that multitask perception and control can be achieved on a low-cost platform with the same accuracy and responsiveness as commercial systems.

While the prototype performed best under ideal daylight and artificially simulated indoor environments, it was not tested in stressful conditions such as nighttime driving, rain, fog, or heavy traffic. All these areas are high-priority for future research and are consistent with the weaknesses of other Raspberry Pi–based autonomous prototypes [8], [12].

5. Discussion

The outcome of this endeavor delivers low-cost embedded platforms to offer real-time autonomous driving features in the appropriate form through parallel processing and optimization of lightweight algorithms. Different from past rapid prototype uses of Raspberry

Pi, which solely focused on executing a specific program such as lane following [8] or extensively relied on other sensors [12], this work merges the rapid prototype uses of three perception modules—lane detection, traffic sign detection, and traffic light detection—into a single framework.

Contributions to innovation are:

- Multithreading and concurrent perception task design that significantly enhances FPS and responsiveness on Raspberry Pi platforms.
- Qualitative detection performance, latency, and frame rate readings show lightweight solutions with strong performance on low-cost real-world computers.
- It can be executed in real-time over Haar Cascades and standard image processing without spending money on high-end GPU-based machines; thus, it serves as a cheap research and learning platform worldwide.

Greater accuracy and varied robustness are expensive with hardware platforms like YOLOP [14] and DeepPicar [9], but they are out of reach due to hardware restrictions. Cost efficiency in autonomous vehicle studies is guaranteed through this paper with varied demonstration performance at low-cost embedded deployment expenses.

As compared to DeepPicar [9], which maintained steering control at 100 Hz on Raspberry Pi but did not achieve lane following over distance, our system demonstrates the advantage of being capable of carrying out multiple perception tasks at rates similar to real time. Indeed, for YOLOP [14], it sees better objects and lanes but at the cost of GPU-based Jetson hardware, which is considerably more expensive and less portable. This platform established here addresses this gap by offering cheap, multi-tasking, real-time independence on Raspberry Pi hardware best suited for learning and research.

6. Future Scope

The new autonomous vehicle model holds enormous promise but comes with a number of restraints that must be addressed in its daily application. The performance of the computer vision algorithm and sensor fusion significantly contributes to the system's reliability and

efficiency, both of which are susceptible to peripheral conditions such as light, weather, and surface. In addition, the Raspberry Pi's computing capacity constraints limit the sophistication of algorithms and real-time processing, reducing system responsiveness and flexibility. The prototype cannot handle sophisticated traffic flows or communicate with other vehicles and pedestrians in an appropriate way and therefore cannot realistically be used in the real world. Even though the prototype proved robust for daytime and indoor operation, sensitivity to visible light hinders its deployment in low or adverse weather conditions. Such limitations were also imposed by earlier Raspberry Pi-based implementations [8], [12]. These results indicate that deep learning-based perception models [9], [14] and sensor fusion technology (e.g., infrared camera, LiDAR, or radar) should be applied in order to provide robustness in real-world deployment.

To transition the prototype to production vehicles, additional safety features like fail-safe emergency braking, redundant control and steering systems, and V2X communication for cooperative driving will have to be incorporated. Safety regulations for automobiles, such as ISO 26262, will have to be adhered to. Bigger power supply units and heavy-duty actuation hardware will need to be incorporated to sustain larger vehicle physical forces.

7. Conclusion

This paper announces a cost-effective autonomous car prototype using Raspberry Pi and Arduino that is able to recognize real-time lanes, obstacles, traffic signs, and traffic lights. Our system realizes an average of 87% accuracy at 13 FPS in lab-controlled indoor settings, which reveals that lightweight image processing can offer trustworthy navigation assistance. Our system achieves outstanding cost-effectiveness, efficiency, and responsiveness compared with other prototypes. Scaling to larger, more realistic scenarios with improved sensors and safety systems must be conducted in the future.

References

[1] Viswanathan, Vidya, and Rania Hussein. "Applications of image processing and real-time embedded systems in autonomous cars: a short review." International Journal of Image Processing (IJIP) 11, no. 2 (2017): 35.

[2] Borenstein, Johann, and Yoram Koren. "Obstacle avoidance with ultrasonic sensors." IEEE Journal on Robotics and Automation 4, no. 2 (2002): 213-218.

[3] Blyth, Pascale-L., Milos N. Mladenovic, Bonnie A. Nardi, Norman M. Su, and Hamid R. Ekbia. "Driving the self-driving vehicle: Expanding the technological design Horizon." In 2015 IEEE International Symposium on Technology and Society (ISTAS), IEEE, 2015, 1-6.

[4] McCall, Joel C., and Mohan M. Trivedi. "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation." IEEE transactions on intelligent transportation systems 7, no. 1 (2006): 20-37.

[5] Fagnant, Daniel J., and Kara Kockelman. "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations." Transportation Research Part A: Policy and Practice 77 (2015): 167-181.

[6] Pawar, Narayan Pandharinath, and Minakshee M. Patil. "Driver assistance system based on Raspberry Pi." International Journal of Computer Applications 95, no. 16 (2014).

[7] Shahane, Vikrant, Hrushikesh Jadhav, Mihir Sansare, and Prathmesh Gunjgur. "A self-driving car platform using raspberry Pi and Arduino." In 2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA, IEEE, 2022, 1-6.

[8] Isherwood, Zach, and Emanuele Lindo Secco. "A Raspberry Pi computer vision system for self-driving cars." In Science and Information Conference, Cham: Springer International Publishing, 2022, 910-924.

[9] Bechtel, Michael G., Elise McEllhiney, Minje Kim, and Heechul Yun. "Deeppicar: A low-cost deep neural network-based autonomous car." In 2018 IEEE 24th international conference on embedded and real-time computing systems and applications (RTCSA), IEEE, 2018, 11-21.

[10] Ravindran, Vijay, S. Chandrika, Ram Prakash Ponraj, C. Krishnakumar, S. Devadharshini, and S. Lakshmi. "Autonomous Delivery Vehicle Using Raspberry Pi and Computer Vision." In International Conference on Paradigms of Communication, Computing and Data Analytics, Singapore: Springer Nature Singapore, 2023, 481-493.

[11] Kumar, HR Sridhar, and Sudarshan Gurupad Hegde. "Autonomous Car Using Raspberry Pi and Can Bus Protocol." International Journal of Innovative Science and Research Technology 8, no. 5 (2023).

[12] Golabhavi, P., and B. Harish. "Self-driving car model using raspberry pi." International Journal of Engineering Research & Technology 9, no. 2 (2020): 176-180.

[13] Wagdy, Mahmoud, Ahmed Roushdy, Ahmed Sabry, Abdallah Mostafa, Mahmoud Magdy, and Amal S. Mehanna. "Autonomous Parallel Parking Using Raspberry-PI." In International Conference on Advanced Technologies for Humanity, Cham: Springer International Publishing, 2021, 19-28.

[14] Wu, Dong, Man-Wen Liao, Wei-Tian Zhang, Xing-Gang Wang, Xiang Bai, Wen-Qing Cheng, and Wen-Yu Liu. "Yolop: You only look once for panoptic driving perception." Machine Intelligence Research 19, no. 6 (2022): 550-562.

[15] Kozłowski, Michał, Szymon Racewicz, and Sławomir Wierzbicki. "Image analysis in autonomous vehicles: A review of the latest AI solutions and their comparison." Applied Sciences 14, no. 18 (2024): 8150.

[16] Gajjar, Henil, Stavan Sanyal, and Manan Shah. "A comprehensive study on lane detecting autonomous car using computer vision." Expert Systems with Applications 233 (2023): 120929.

[17] Canny, John. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (2009): 679-698.

[18] Mo, Tiande, Siqian Zheng, Wai-Yat Chan, and Renhua Yang. "Review of AI image enhancement techniques for in-vehicle vision systems under adverse weather conditions." World Electric Vehicle Journal 16, no. 2 (2025): 72. Zhang, Yuxiao, Alexander Carballo, Hanting Yang, and Kazuya Takeda. "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey." ISPRS Journal of Photogrammetry and Remote Sensing 196 (2023): 146-177.

[19] Cheng, Hong. Autonomous intelligent vehicles: theory, algorithms, and implementation. Springer Science & Business Media, 2011.

[20] Fathy, Mahmoud, Nada Ashraf, Omar Ismail, Sarah Fouad, Lobna Shaheen, and Alaa Hamdy. "Design and implementation of self-driving car." Procedia Computer Science 175 (2020): 165-172.

[21] Woo, Seokkyun, Jan Youtie, Ingrid Ott, and Fenja Scheu. "Understanding the long-term emergence of autonomous vehicles technologies." Technological Forecasting and Social Change 170 (2021): 120852.

[22] Zhang, Jingyu, Jin Cao, Jinghao Chang, Xinjin Li, Houze Liu, and Zhenglin Li. "Research on the application of computer vision based on deep learning in autonomous driving technology." In INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND APPLICATIONS, Singapore: Springer Nature Singapore, 2023, 82-91.

[23] Miller, Tymoteusz, Irmina Durlik, Ewelina Kostecka, Piotr Borkowski, and Adrianna Łobodzińska. "A critical AI view on autonomous vehicle navigation: The growing danger." Electronics 13, no. 18 (2024): 3660.

[24] Du, Mingfang. Autonomous Vehicle Technology. Springer, 2023.

[25] Janai, Joel, Fatma Güney, Aseem Behl, and Andreas Geiger. "Computer vision for autonomous vehicles: Problems, datasets and state of the art." Foundations and trends® in computer graphics and vision 12, no. 1–3 (2020): 1-308.