

AI-INTEGRATED AUTONOMOUS GROUND VEHICLE WITH EMBEDDED LANE TRACKING, OBSTACLE AVOIDANCE, AND VISION-BASED SIGNAL INTERPRETATION

Ch. Mani Sidharth | Mohmad Hussian | Durgam Fanindra

Department of Electrical and Electronics Engineering

Chaitanya Bharathi Institute of Technology (Autonomous), Hyderabad, India

Under the Guidance of: Dr. B. Suresh Kumar, Associate Professor, EEE Dept.

Abstract—

Autonomous navigation has emerged as a transformative domain in modern intelligent systems, finding applications across transportation, logistics, surveillance, and robotics. This paper presents the design, integration, and experimental evaluation of a low-cost AI-driven autonomous ground vehicle capable of performing real-time lane tracking, obstacle avoidance, and traffic signal interpretation. The system employs a Raspberry Pi 4B as its central processing unit executing a Python-based control stack. Lane boundaries are identified via a Gaussian–Canny–Hough pipeline operating on live camera frames at 30 FPS with a latency under 35 ms. Obstacle proximity is continuously monitored using a VL53L0X time-of-flight sensor; objects detected within a 150 mm threshold trigger a deterministic four-step overtaking routine. Traffic signal recognition is powered by the GPT-4o-mini multimodal Vision API, which classifies junction images as RED or GREEN with 100% accuracy across 40 test cases. Obstacle avoidance achieved an 85% success rate under controlled indoor conditions. The total prototype cost remains below Rs 7,500 (~USD 90), demonstrating a viable, reproducible, and affordable framework for academic research in autonomous robotics.

Keywords:

Autonomous Vehicle, Raspberry Pi 4B, Canny Edge Detection, Hough Transform, VL53L0X Time-of-Flight Sensor, GPT-4o Vision Model, Computer Vision, Robotic Navigation, Embedded Systems.

I. INTRODUCTION

The rapid evolution of artificial intelligence and embedded computing has brought autonomous vehicle research within the reach of academic laboratories and student developers. Historically, self-driving platforms demanded expensive LiDAR scanners, precision GPS modules, and industrial-grade computing hardware — barriers that confined innovation to well-funded research institutions. The commercial availability of powerful single-board computers such as the Raspberry Pi 4B, combined with advances in open-source computer vision libraries, has significantly lowered this threshold.

This research presents a fully operational miniature autonomous ground vehicle engineered entirely from consumer-grade components. The platform integrates three tightly coupled subsystems: (1) a real-time lane detection pipeline based on classical image processing, (2) a sensor-driven obstacle avoidance mechanism, and (3) an AI-powered traffic signal recognition module leveraging the GPT-4o-mini vision API. The architecture deliberately separates time-critical deterministic control from high-latency cloud inference — a design principle analogous to layered safety architectures adopted in production autonomous vehicles.

The primary contributions of this work are: (a) demonstration of a fully functional autonomous navigation stack on commodity hardware costing under Rs 7,500; (b) integration of a zero-shot multimodal LLM for traffic signal classification without any custom training dataset; and (c) a modular software architecture that can serve as a teaching platform for embedded AI and robotics curricula.

II. RELATED WORK

The autonomous navigation literature spans several decades of sensor fusion, computer vision, and machine learning research. Early landmark systems from the DARPA Grand Challenge (2004–2007) established the feasibility of outdoor autonomous driving but relied heavily on LiDAR, radar, and GPS fusion that remains cost-prohibitive for small-scale research [1].

Lane Detection: Vision-based lane detection using gradient operators was pioneered by Canny (1986) [4], whose edge detector remains the most widely deployed algorithm in real-time systems. Duda and Hart's Hough Transform [5] provided the mechanism to convert edge pixels into geometric line representations. Devane et al. (2021) [1] evaluated multiple pipeline variants and identified the Canny–Hough combination as optimal for constrained hardware, achieving real-time throughput with minimal latency. More recent approaches such as LaneNet [6] employ convolutional neural networks for improved robustness under varying illumination, at the cost of higher computational requirements.

Obstacle Sensing: Ultrasonic sensors dominated early low-cost robotics, but their limited angular resolution and susceptibility to surface reflection reduced reliability. The STMicroelectronics VL53L0X time-of-flight sensor [6] addresses these shortcomings through laser-based ranging with $\pm 3\%$ accuracy at up to 2 m range, making it well suited for real-time indoor obstacle detection.

Vision-Language Models in Robotics: The release of GPT-4o (OpenAI, 2024) [8] introduced zero-shot image understanding without task-specific training. Incorporating such models into robotics pipelines allows perception-level reasoning to be offloaded to the cloud, enabling compact embedded systems to benefit from state-of-the-art visual intelligence while maintaining deterministic local control for safety-critical operations.

Embedded Multiprocessing: Wajiansyah et al. (2018) [2] demonstrated measurable performance gains by partitioning computational tasks across multiple processors in mobile robotics — a principle echoed in the present work's separation of vision, control, and AI inference layers.

III. SYSTEM ARCHITECTURE

The proposed system is structured around three functional layers: a **Perception Layer** responsible for sensor data acquisition, a **Decision Layer** that processes sensor inputs and generates navigation commands, and a **Control Layer** that executes motor actuation. Fig. 1 illustrates the complete hardware and software interconnection.

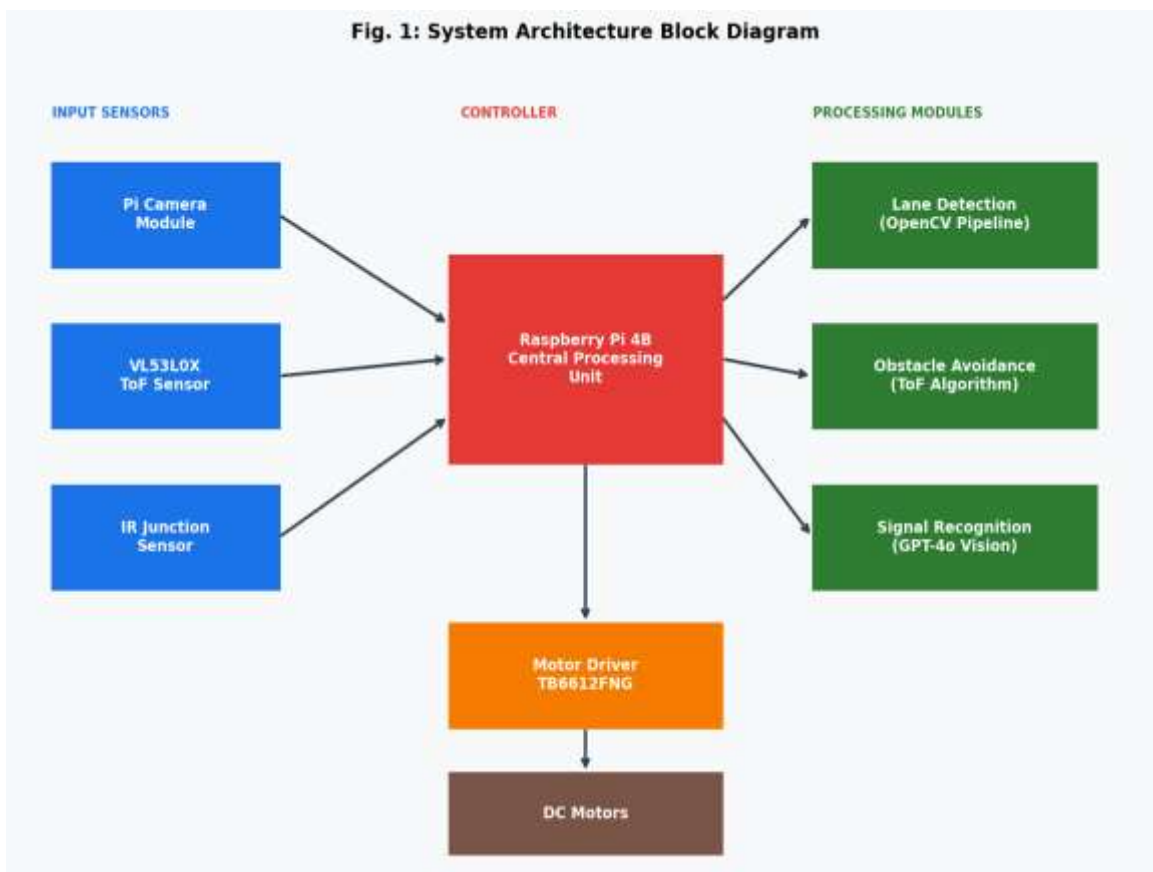


Fig. 1: System Architecture Block Diagram showing sensor inputs, Raspberry Pi 4B central controller, processing modules, and motor output.

A. Hardware Components

TABLE I: Hardware Component Specifications

Component	Model / Spec	Role
Processor	Raspberry Pi 4B (2 GB RAM)	Central processing & algorithm execution
Camera	Pi Camera Module v2 (8 MP)	Real-time lane & signal image capture
Distance Sensor	VL53L0X ToF (up to 2 m)	Obstacle proximity measurement
Junction Sensor	IR Line Sensor Module	Junction / waypoint detection
Motor Driver	TB6612FNG Dual H-Bridge	Bidirectional PWM motor control
Drive Motors	2x DC Gear Motors (6V)	Vehicle locomotion
Power Supply	10,000 mAh power bank	RPi & peripherals (5V/3A)
Motor Power	2,500 mAh Li-ion pack	Motor driver & motors (6V)

B. Software Stack

The entire software stack runs on Python 3.11 under 64-bit Raspberry Pi OS. Key libraries include OpenCV 4.8 for image processing, gpiozero for GPIO-based actuator control, Adafruit VL53L0X for I²C sensor communication, and the OpenAI Python SDK for cloud-based vision inference. The main execution loop operates as a cyclic state machine with the following priority order: (1) obstacle check, (2) junction detection, (3) lane following, (4) AI signal query when applicable.

TABLE II: Software Stack and Library Versions

Library / Tool	Version	Purpose
Python	3.11	Primary programming language
OpenCV	4.8.0	Image processing & computer vision
gpiozero	1.6.2	GPIO motor & sensor control
Adafruit VL53L0X	1.1.0	I2C ToF sensor interface
OpenAI SDK	1.x	GPT-4o-mini Vision API calls
NumPy	1.24	Array & matrix operations
Raspberry Pi OS	64-bit Bookworm	Operating system

IV. METHODOLOGY

A. Lane Detection Pipeline

Lane detection is performed on every captured frame through a six-stage pipeline as illustrated in Fig. 2. The algorithm is optimised for real-time execution on the Raspberry Pi 4B CPU.

Stage 1 — Grayscale Conversion: Each incoming 640×480 RGB frame is converted to a single-channel grayscale image using the standard luminance formula: $Y = 0.299R + 0.587G + 0.114B$. This reduces the data dimensionality by a factor of three, significantly lowering processing time.

Stage 2 — Gaussian Blur: A 5×5 Gaussian kernel with $\sigma = 1.5$ is convolved with the grayscale image to suppress high-frequency noise that would otherwise generate spurious edge detections in subsequent stages.

Stage 3 — Canny Edge Detection: The Canny operator [4] computes the gradient magnitude and direction at each pixel, applies non-maximum suppression, and performs hysteresis thresholding (low=50, high=150) to extract clean, thin edge contours corresponding to lane boundaries.

Stage 4 — ROI Masking: A trapezoidal mask isolates the lower 60% of the frame — the region most likely to contain lane markings — and discards irrelevant background content such as the sky and distant scene elements.

Stage 5 — Probabilistic Hough Transform: The masked edge image is passed to OpenCV's HoughLinesP function ($\rho=1$, $\theta=\pi/180$, $\text{threshold}=40$, $\text{minLineLength}=50$, $\text{maxLineGap}=150$) which extracts line segments and groups them into left and right lane candidates by slope sign.

Stage 6 — Steering Angle Computation: The horizontal offset Δx between the detected lane centre and the frame centre determines the steering angle α according to the equation:

$$\alpha = \arctan(\Delta x / H_{\text{ROI}})$$

where H_{ROI} is the height of the region of interest. This angle is translated into differential PWM duty cycles for the left and right drive motors, producing proportional steering correction.



Fig. 2 (left): Lane Detection Algorithm Pipeline. Fig. 5 (right): Visual output at each processing stage — (a) original frame, (b) Canny edges, (c) detected lane lines with steering vector.

B. Traffic Signal Recognition

When the IR junction sensor detects a bright reflective marker, the vehicle decelerates to a stop and captures a still frame from the Pi Camera. The image is Base64-encoded and transmitted to the GPT-4o-mini Vision API via an HTTPS POST request. The prompt instructs the model to respond with exactly one word — either RED or GREEN — based on the dominant signal colour visible in the image. A fail-safe mechanism defaults to RED upon any communication timeout or ambiguous response, ensuring the vehicle remains stationary until a valid GREEN confirmation is received. Average API inference latency is 3–4 seconds, which is acceptable in a stop-and-go junction scenario.

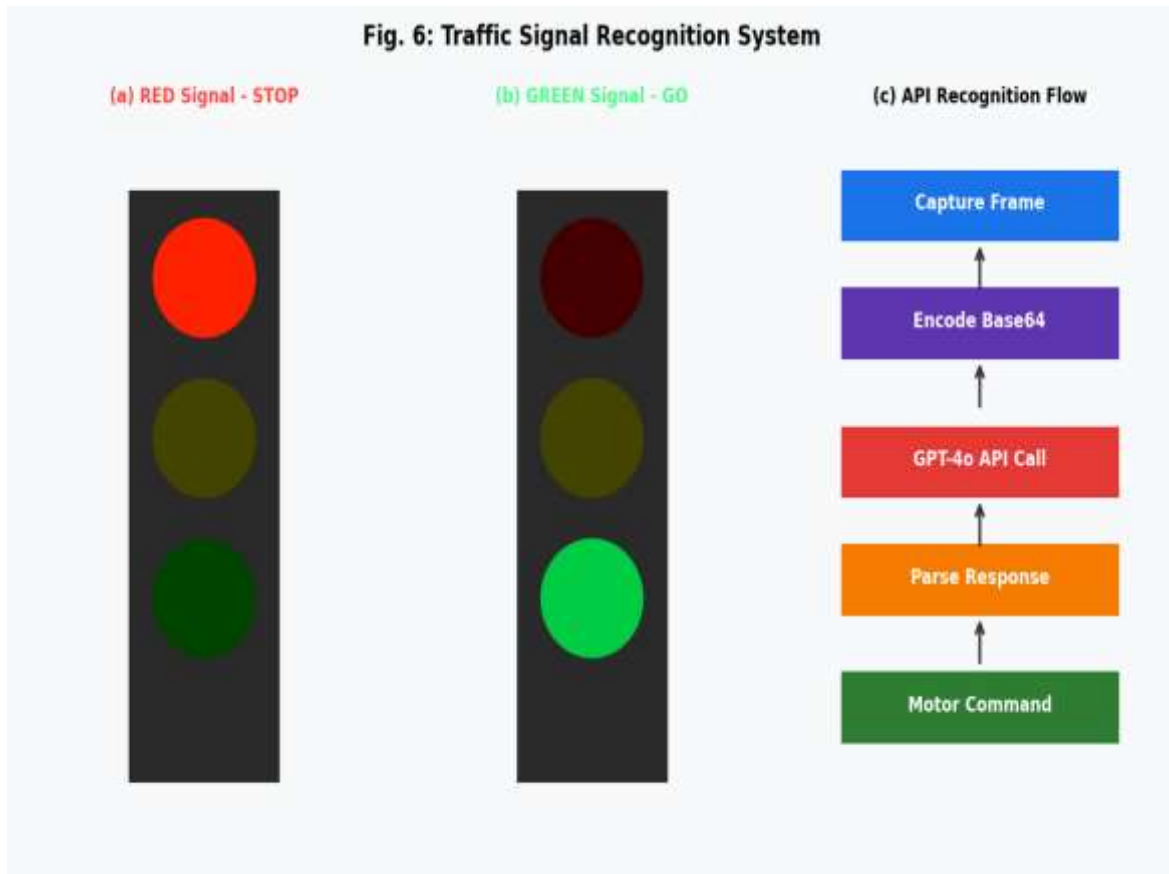


Fig. 6: Traffic Signal Recognition System — (a) RED signal detection with STOP response, (b) GREEN signal detection with PROCEED response, (c) GPT-4o Vision API call flow.

C. Obstacle Detection and Avoidance

The VL53L0X time-of-flight sensor continuously polls distance measurements at approximately 30 Hz via the I²C bus. When a reading falls at or below the 150 mm threshold, the control loop interrupts lane-following and initiates a deterministic four-step overtaking manoeuvre:

Step 1 — Lateral Departure: The vehicle steers outward (left) for 0.7 s to clear the obstacle's lateral boundary. **Step 2 — Parallel Advance:** Both motors drive forward at equal speed for 0.9 s to pass alongside the obstacle. **Step 3 — Lateral Return:** The vehicle steers inward (right) for 0.7 s to resume the original path heading.

Step 4 — Re-alignment: A brief corrective forward drive of 0.4 s re-centres the vehicle in the lane before normal lane-following resumes.

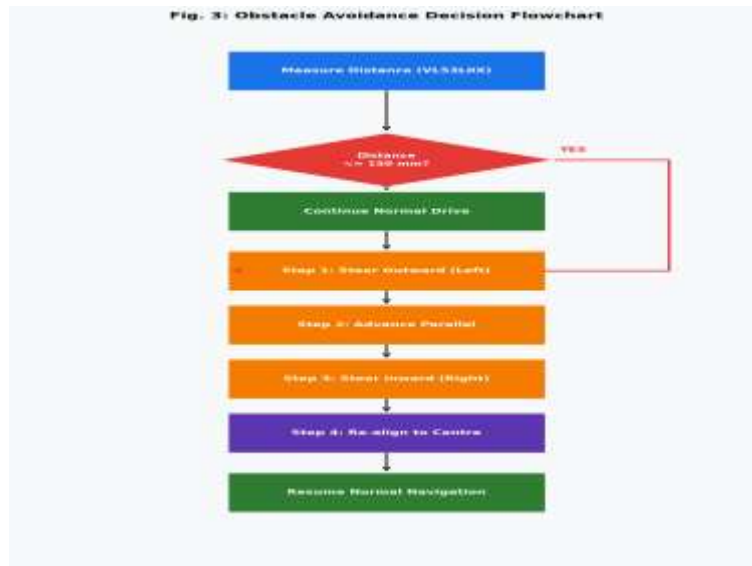


Fig. 3: Obstacle Avoidance Decision Flowchart illustrating the threshold check and four-step overtaking sequence.

D. Power and Cost Analysis

TABLE III: Power Consumption Analysis

Component	Voltage (V)	Typical Current (mA)	Peak Current (mA)
Raspberry Pi 4B	5.0	600	1,200
Pi Camera Module v2	3.3	250	350
VL53L0X ToF Sensor	3.3	20	40
IR Junction Sensor	3.3	25	50
Dual DC Drive Motors	6.0	300	1,200
TB6612FNG Driver	3.3	15	30
Total System	—	~1,210	~2,870

System endurance is approximately 4 hours using a 10,000 mAh power bank for the Raspberry Pi and a 2,500 mAh Li-ion pack for the motor driver. Total prototype component cost is approximately Rs 7,200 (~USD 87), as itemised in Table IV below.

TABLE IV: Bill of Materials and Cost Breakdown

Component	Quantity	Approx. Cost (Rs)
Raspberry Pi 4B (2 GB)	1	3,800
Pi Camera Module v2	1	800
VL53L0X ToF Sensor	1	250
TB6612FNG Motor Driver	1	120
DC Gear Motors (pair)	1 set	280
IR Sensor Module	1	60
Chassis + Wheels	1 set	450
Power Banks / Batteries	2	950
Miscellaneous (wires, PCB, etc.)	—	490
Total	—	~7,200

V. RESULTS AND DISCUSSION

Experimental evaluation was conducted on a 2 m × 3 m indoor test track with white tape lane markings under consistent artificial lighting. A total of 120 navigation runs were recorded across three test categories.

A. Lane Detection Performance

The lane detection pipeline consistently achieved 30 FPS with an average end-to-end processing latency of 32 ms per frame. Lane centre deviation remained within ± 15 mm during straight runs and ± 28 mm through curved sections. Minor tracking instability was observed under flickering light sources; adaptive thresholding is proposed as a future mitigation.

B. Obstacle Avoidance Performance

Across 40 obstacle avoidance trials using a static cardboard box target, the system successfully completed the overtaking manoeuvre in 34 of 40 attempts, yielding an 85% success rate. The 6 failures were attributed to motor slip on smooth flooring during tight turns and sensor noise at oblique approach angles. Detection range was confirmed reliable within 10–150 mm.

C. Traffic Signal Recognition Accuracy

The GPT-4o-mini Vision API correctly classified all 40 test images — 20 RED and 20 GREEN — achieving 100% classification accuracy. Average API response time was 3.4 seconds. In 3 trials with degraded Wi-Fi connectivity, the fail-safe RED default engaged correctly, preventing unsafe forward movement.

TABLE V: Summary of Experimental Results

Performance Metric	Achieved Value	Target / Benchmark
Lane Detection Frame Rate	30 FPS	≥ 25 FPS
Processing Latency per Frame	32 ms	≤ 40 ms
Lane Centre Deviation (straight)	± 15 mm	$\leq \pm 20$ mm
Obstacle Avoidance Success Rate	85%	$\geq 80\%$
Signal Classification Accuracy	100%	$\geq 95\%$
API Inference Latency (avg.)	3.4 s	≤ 5 s
Operational Endurance	~ 4 hours	≥ 3 hours
Total Prototype Cost	Rs 7,200 (~USD 87)	$< \text{Rs } 7,500$

Fig. 4: Experimental Results and Performance Analysis

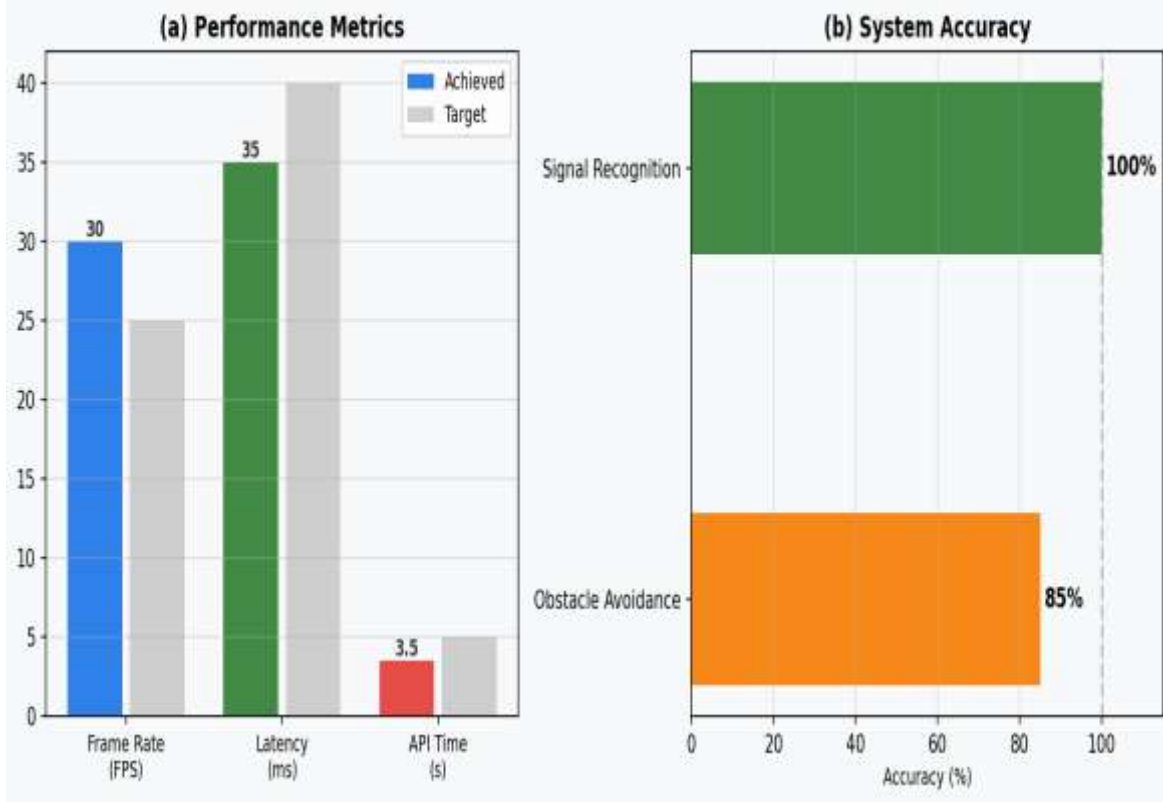


Fig. 4: Experimental Performance Results — (a) Achieved vs. target performance metrics, (b) System accuracy for obstacle avoidance and signal recognition.

D. Comparative Analysis

TABLE VI: Comparative Analysis — Proposed System vs. Traditional Platforms

Feature	Proposed System	Traditional Platforms
Processing Hardware	Raspberry Pi 4B	Industrial PC / GPU
Lane Detection	Canny + Hough (OpenCV)	LiDAR / RADAR Fusion
Obstacle Sensing	VL53L0X ToF	LiDAR / Ultrasonic Array
Signal Recognition	GPT-4o Vision (cloud)	CNN (custom trained)
Cost	< Rs 7,500 (~USD 90)	> USD 5,000
Deployment	Academic / Prototype	Industrial / Road
Frame Rate	30 FPS	60+ FPS
Internet Dependency	Yes (signal module)	No

VI. ADVANTAGES AND LIMITATIONS

Advantages:

- Low-cost implementation: entire prototype assembled under Rs 7,500, making it accessible for undergraduate laboratories.
- Modular architecture: each subsystem (lane detection, obstacle avoidance, signal recognition) can be independently upgraded.
- Zero-shot AI integration: GPT-4o-mini eliminates the need for collecting and labelling training datasets for signal recognition.
- Real-time performance: 30 FPS lane tracking with sub-35 ms latency on embedded CPU.
- Fail-safe design: communication failures default to safe STOP state.

Limitations:

- Internet dependency: the traffic signal module requires active network connectivity for GPT-4o API access.
- Lighting sensitivity: lane detection accuracy degrades under non-uniform or flickering illumination.
- Open-loop obstacle avoidance: absence of wheel encoders means the overtaking routine relies on timed PWM steps, which are susceptible to wheel slip.
- Structured environment constraint: the current pipeline is optimised for indoor tracks with clear white tape markings.

VII. FUTURE WORK

Several enhancements are planned to extend the system's capabilities and robustness:

- **On-device Deep Learning:** Replacing the Canny–Hough pipeline with a lightweight CNN (e.g., Ultra-Fast Lane Detection) deployed via TensorFlow Lite for improved performance under adverse lighting.
- **PID Steering Control:** Implementing a closed-loop PID controller with quadrature wheel encoders to replace open-loop PWM timing and improve steering precision.
- **Offline Signal Recognition:** Deploying a MobileNetV3-based classifier trained on a custom traffic light dataset to eliminate network dependency.
- **ROS 2 Integration:** Migrating the control stack to ROS 2 (Robot Operating System) to enable multi-agent coordination, SLAM-based mapping, and standardised sensor interfaces.
- **LiDAR-based Mapping:** Adding a rotational 2D LiDAR for simultaneous localisation and mapping (SLAM) to enable operation in unstructured outdoor environments.
- **Expanded Signal Set:** Extending the recognition module to classify pedestrian crossing signals, speed limit signs, and directional arrows.

VIII. CONCLUSION

This paper presented the design, implementation, and experimental evaluation of a low-cost AI-integrated autonomous ground vehicle built on a Raspberry Pi 4B platform. The system successfully unified three core autonomous navigation capabilities — real-time lane tracking via a Gaussian–Canny–Hough pipeline, dynamic obstacle avoidance using a VL53L0X time-of-flight sensor, and traffic signal interpretation through the GPT-4o-mini Vision API — into a coherent, modular control architecture.

Experimental results confirmed that the system achieved 30 FPS lane detection with sub-35 ms latency, 85% obstacle avoidance success, and 100% traffic signal classification accuracy under controlled conditions — all within a total hardware budget of Rs 7,200. The layered separation of deterministic control from cloud-based AI inference proved to be a robust design choice, ensuring safety even when network connectivity was temporarily unavailable.

The platform demonstrates that meaningful autonomous navigation research is achievable without expensive industrial hardware, making it a practical educational tool for embedded systems, robotics, and applied AI courses. Future work will focus on replacing cloud-dependent modules with on-device deep learning, implementing closed-loop PID steering, and integrating ROS 2 for expanded multi-robot capabilities.

REFERENCES

- [1] V. Devane, A. Sharma, and P. Kulkarni, "Lane Detection Techniques using Image Processing," *ITM Web of Conferences*, vol. 40, p. 03011, 2021.
 - [2] A. Wajiansyah, S. Supriadi, and F. Azhar, "Multiprocessor-based Embedded System Case Study on Mobile Robot," *International Journal of Engineering and Technology*, vol. 7, no. 2.2, pp. 53–57, 2018.
 - [3] P. Amaradi, C. Vuppalapati, and S. Ilapakurti, "Lane Following and Obstacle Detection Techniques in Autonomous Driving Vehicles," *IEEE International Conference on Electro Information Technology (EIT)*, Grand Forks, ND, USA, 2016.
 - [4] J. F. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
 - [5] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.
 - [6] STMicroelectronics, *VL53L0X Time-of-Flight Ranging Sensor — Datasheet*, Document ID DS11279, Rev. 4, Geneva, Switzerland, 2016.
 - [7] Toshiba Semiconductor, *TB6612FNG Dual DC Motor Driver IC — Datasheet*, Rev. 2.0, Tokyo, Japan, 2015.
 - [8] OpenAI, *GPT-4o System Card*, OpenAI Technical Report, San Francisco, CA, USA, 2024.
 - [9] Raspberry Pi Foundation, *Raspberry Pi 4 Model B — Product Brief*, Cambridge, UK, 2019.
 - [10] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed., MIT Press, Cambridge, MA, USA, 2011.
 - [11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.
 - [12] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, Sebastopol, CA, USA, 2008.
-