Proceedings of the 3rd International Conference on Functional Materials and Civil Engineering
DOI: 10.54254/2755-2721/2025.29347

Design and Implementation of a Multi-Tasking Vehicle Using Raspberry Pi and OpenCV

Yuang Wei

Department of Electrical and Electronic Engineering, Faculty of Science and Engineering, University of Nottingham Ningbo China, Ningbo, China
2744374466@qq.com

Abstract. This work investigates how a low-cost embedded platform can achieve reliable sensor fusion and multitask integration for autonomous navigation. A Raspberry Pi serves as the central controller, coordinating heterogeneous modules for line following, traffic-light recognition, ultrasonic-based stopping, audio feedback, and geometric symbol detection. Two key technical issues are examined: (i) the accuracy and reliability of ultrasonic ranging across distance, and (ii) the selection of image preprocessing parameters—particularly Gaussian kernel size and binarization threshold—to ensure robust vision under variable illumination and noise. Experimental results confirm that the bidirectional level shifter achieved stable 3.3–5 V translation, the ultrasonic sensor maintained reliable accuracy within 0–7 m with errors increasing at longer ranges, and optimized preprocessing improved trajectory extraction and template recognition. A discrete PID controller minimized path deviation, enabling smooth motion through curves and intersections. Overall, the findings demonstrate that compact, low-cost platforms can reliably integrate heterogeneous sensors for multitask robotic navigation, providing practical insights for robotics education and broader implications for robust sensor fusion in complex environments.

Keywords: Raspberry Pi, Sensor Fusion, Line Following, PID Control, Object Recognition

1. Introduction

With the rapid advancement of embedded computing and robotics, Raspberry Pi–based autonomous vehicles have become an effective platform for both education and research. Conventional line-following cars, however, are typically limited to single-task execution and lack environmental perception or multitasking capability. To address these limitations, this work designs and implements a multitasking line-following vehicle that integrates sensing, vision, and control.

The system employs a Raspberry Pi as the central controller, integrating an HC-SR04 ultrasonic sensor for distance measurement, an 8 MP camera for line tracking and object recognition, and peripheral circuits including a bidirectional level shifter and an audio amplifier. OpenCV is used for image preprocessing, morphological operations, and template matching, while a PID controller ensures robust trajectory tracking.

Recent reviews have underscored the Raspberry Pi's versatility for low-cost prototyping in automation, robotics, and intelligent monitoring, emphasizing its capability to unify sensing,

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.29347

decision-making, and actuation within compact embedded systems. Building upon this foundation, the present study demonstrates that sensor fusion and computer vision can significantly enhance autonomous navigation. The results not only validate the effectiveness of the proposed design but also provide practical insights for future robotics and intelligent-vehicle applications [1].

2. Background

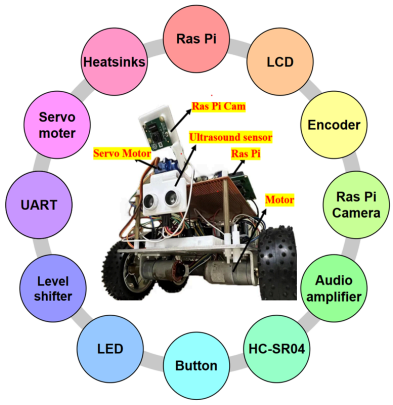

Figure 1. Prototype of the autonomous vehicle integrating key hardware components

As shown in Fig. 1, the proposed multitasking autonomous vehicle is built on a compact Raspberry Pi platform that integrates sensing, perception, and control into a unified perception–decision–actuation framework. The Raspberry Pi serves as the central controller, coordinating peripheral modules to achieve low-cost and reliable multitask operation. The Pi Camera performs lane tracking and symbol recognition through OpenCV-based image processing, including grayscale conversion, Gaussian smoothing, binarization, and morphological filtering for contour extraction and template matching. The HC-SR04 ultrasonic sensor measures distance via the sonar time-of-flight principle, initiating automatic stops when obstacles fall within a defined threshold. The SG90 servo motor converts control commands into steering angles through pulse-width modulation (PWM), while a discrete PID controller generates corrective steering signals based on the deviation between the detected line and the vehicle's central axis. In addition, an LM386 audio amplifier drives an 8 Ω loudspeaker to provide auditory feedback, enabling multimodal interaction that enhances system responsiveness. Collectively, these modules form an integrated platform demonstrating how compact embedded hardware can support robust sensor fusion and multitask coordination [1-5].

DOI: 10.54254/2755-2721/2025.29347

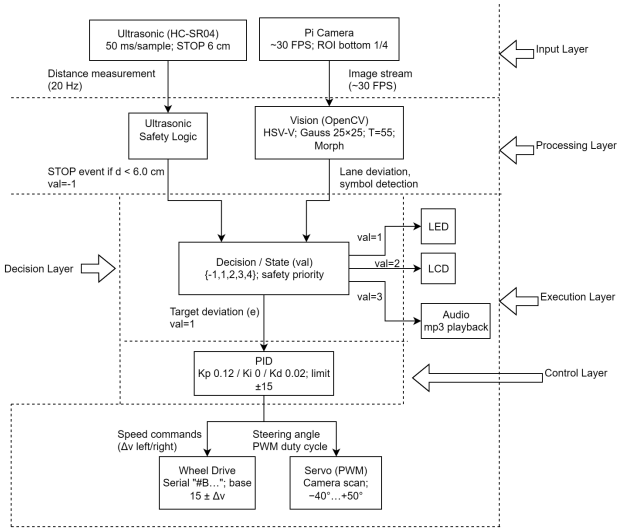

Figure 2. System architecture of the multitasking line-following vehicle

As illustrated in Fig. 2, the system adopts hierarchical architecture composed of three functional layers. The sensing layer captures environmental data through the Pi Camera and ultrasonic sensor, while the decision layer processes these inputs via OpenCV-based vision algorithms and a discrete PID controller to determine steering and velocity targets. The execution layer then actuates the wheel motors, display module, and audio components to realize control outputs. Continuous feedback among layers ensures real-time adaptability, enabling precise lane following, obstacle avoidance, and symbol-based task execution under dynamic lighting and environmental conditions. This modular and scalable design achieves multitask functionality on a single-board platform, offering a practical foundation for educational and research applications in embedded robotics and autonomous systems.

3. Methods

3.1. Ultrasonic distance measurement and vehicle stopping

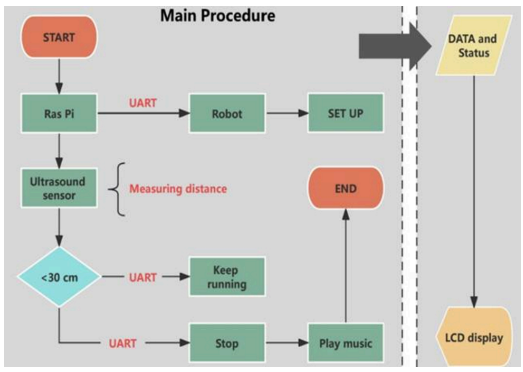

Figure 3. Control flow of the ultrasonic distance measurement and stopping mechanism

As illustrated in Fig. 3, the HC-SR04 ultrasonic sensor was mounted at the front of a four-wheel platform to evaluate its distance-based stopping performance. The module was periodically triggered by a 10 μs pulse, and the echo duration was measured via GPIO timing to compute distance according to (1). When the detected range exceeded the preset threshold, the vehicle maintained its

DOI: 10.54254/2755-2721/2025.29347

motion; once the distance fell below 30 cm, the Raspberry Pi issued a stop command through UART, activated audio feedback, and updated the LCD display. This closed-loop routine achieved consistent obstacle detection and reliable halting within 2–300 cm, with only slight accuracy degradation at longer ranges due to echo attenuation and multipath reflections.

3.2. Vision-based line following with PID control

For stable trajectory tracking, the Pi Camera operated at a resolution of 1280 × 720 with manually tuned brightness and contrast. A 1280 × 100 region of interest (ROI) centered on the track was extracted and converted to grayscale using the weighted-luminance model [8]:

$$I(x,y) = 0.299R + 0.587G + 0.114B, \tag{1}$$

The frame was then binarized to isolate the lane markings [8]:

$$\mathrm{dst}(x,y) = \begin{cases} 255, & \text{if } \mathrm{img}(x,y) > T \\ 0, & otherwise \end{cases} \tag{2}$$

where T denotes the adaptive threshold separating the black line from the background.Noise was suppressed with Gaussian smoothing [8]:

$$I_g(x,y) = \sum_{x',y'} G(x',y') \cdot I(x-x', y-y') \tag{3}$$

and morphological opening–closing operations [6] removed small artifacts and reconnected broken edges.

The lane center x_c was determined from the mean horizontal coordinate of all black pixels,

$$x_c = \frac{1}{N_{pixel}} \sum_{i=1}^{N_{pixel}} x_i \tag{4}$$

and the lateral deviation e between x_c and the image midpoint represented the steering error.

A discrete PID controller generated the steering signal [9]:

$$The\ control\ law\ is:\ u[k] = K_p \cdot e[k] + K_i \cdot \Sigma e[i] + K_d \cdot (e[k] - e[k-1]) \tag{5}$$

with $K_p = 0.12,\ K_i = 0,\ K_d = 0.02$.

The integral term was disabled to prevent windup, and derivative filtering attenuated high-frequency jitter from visual noise [2]. Differential wheel speeds were adjusted around a nominal base velocity $V_{\text{base}} = 15$, limiting steering to ±15 units. When $e > 0$, the robot accelerated the left wheels and slowed the right ones; when $e < 0$, the opposite occurred, ensuring symmetric motion. Output saturation and anti-windup maintained stability.

This image-based feedback loop achieved smooth and reliable line tracking under varying illumination, confirming the robustness of the vision-PID scheme implemented on the Raspberry Pi platform.

DOI: 10.54254/2755-2721/2025.29347

3.3. Object recognition and task execution

The object recognition module was implemented in OpenCV C++ within the unified perception–decision–actuation pipeline. The algorithm integrates preprocessing, color segmentation, contour analysis, and template matching with decision fusion. Each frame undergoes Gaussian smoothing (5×5 kernel, σ = 1.5) to suppress high-frequency noise while preserving edges for stable thresholding and contour detection.

Color segmentation is performed in the HSV color space: the BGR image is converted to HSV and thresholded for red, blue, green, and yellow to obtain binary masks. For each color c and pixel x, the binary decision is defined:

$$M_c(x) = \begin{cases} 1, I_{hsv}(x) \in [\boldsymbol{L_C}, \boldsymbol{U_C}] \\ 0, \ otherwise \end{cases} \tag{6}$$

$$N_c = \sum_x M_c(x) \tag{7}$$

and the color c is detected when $N_c > T_c$ ($T_c = 1000$ in experiments).

Contour and shape recognition were then applied to the dominant color mask using polygonal approximation $(\varepsilon = 0.02\,\mathrm{arcLength})$. Classification followed vertex count and aspect ratio: 3→triangle, 4 with $0.8 \le AR \le 1.2$ →square, >6 →circle.

For viewpoint and scale robustness, template matching employed "largest-contour localization—homography-based rectification—normalized comparison." The warped 200×200 image was thresholded and compared to reference templates using pixel-wise similarity [10]:

$$S = 100 \cdot \tfrac{1}{N} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathrm{b1}![T_{ij} = I_{ij}] \tag{8}$$

and a match is accepted if $S > 65$.

For control feedback, the centroid and horizontal deviation are computed from image moments [8]:

$$x_c = \tfrac{m_{10}}{m_{00}}, \ y_c = \tfrac{m_{01}}{m_{00}}, \tag{9}$$

$$d = x_c - \tfrac{W}{2} \tag{10}$$

where d denotes the trajectory offset relative to the frame center.

A hierarchical decision strategy was implemented to integrate recognition results from multiple stages. Template matching was prioritized for its high reliability, followed by color detection, and shape recognition as fallback. If all methods failed, no class output was produced. Key hyperparameters included a Gaussian kernel of 5×5 (σ = 1.5), HSV ranges for four target colors (red, blue, green, yellow), a color pixel threshold of 1000, a polygonal approximation ratio of 0.02, a square aspect-ratio range of [0.8, 1.2], and a normalized template size of 200×200 pixels. Class-specific similarity thresholds were set at 65 for red, blue, and yellow, and 45 for green [10].

Each detected class $c \in \{\text{red}, \text{blue}, \text{green}, \text{yellow}\}$ was mapped deterministically to a predefined system action a: red triggered GPIO flashing, blue updated the LCD counter, green initiated audio playback, and yellow activated ultrasonic ranging with a stop condition (TRIG = 5, ECHO = 6). To improve reliability, activation occurred only when the class similarity exceeded its threshold in two consecutive frames and when the inter-trigger interval exceeded three seconds.

DOI: 10.54254/2755-2721/2025.29347

The overall execution pipeline, summarized in Fig. 4, combines line following, object recognition, and task-specific actuation within a unified perception–decision–control loop, achieving stable real-time interaction on the Raspberry Pi platform.

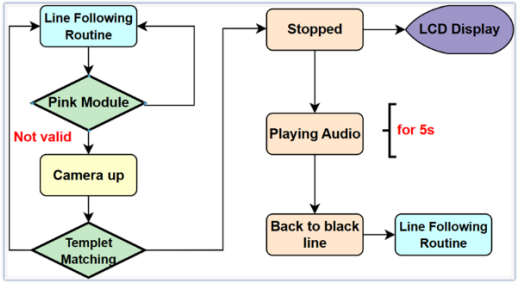

Figure 4. Flowchart of the task execution process including line following, object recognition

4. Results and discussion

4.1. Line following using a raspberry Pi Camera

The experimental track (Fig. 5) incorporated multiple functional nodes—traffic-light recognition, audio-visual alarms, music playback, ultrasonic distance-based stopping, shape counting, shortcut detection, and football kicking. During line following, the vehicle responded to these nodes in real time, providing an integrated evaluation of perception, decision-making, and control under multitasking conditions.

Results show stable trajectory tracking through curves and right-angle turns, verifying the robustness of the PID controller. Quantitatively, traffic-light recognition reached 96.7 % accuracy (0.41 ± 0.08 s response), shape counting 93.3 % (±0.3 object error), and ultrasonic stopping 100 % success within 0–7 m (±2.1 cm, σ = 0.9 cm). Shortcut detection achieved 95.0 % accuracy, while music playback and alarm flashing reached 100 % success. Even in dynamic tasks such as football kicking, success remained 90 % (±2.9 cm offset, σ = 1.3 cm). Task-specific results are summarized in Table 1.

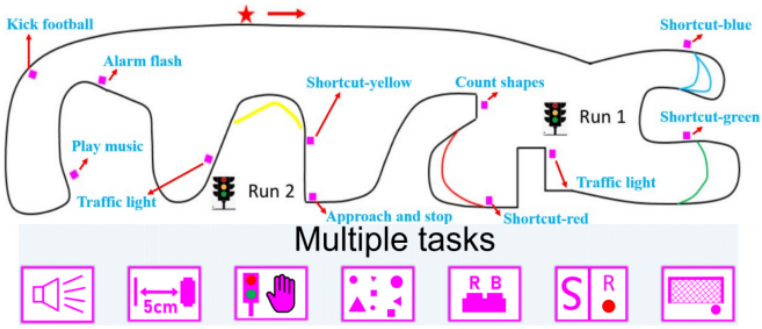

Figure 5. Track environment with integrated multi-task nodes for vehicle testing

DOI: 10.54254/2755-2721/2025.29347

Table 1. Methods and results of multiple-task experiments for the autonomous vehicle

Task	Method	Result
Shortcut	Camera + template matching	Recognition accuracy 95.0% (19/20), avg. response time 0.6 s
Count shapes	HSV + contour approximation	Correctness 93.3% (28/30), avg. error ±1 object
Traffic light	Color thresholding	Recognition accuracy 96.7% (29/30), avg. response time 0.4 s
Approach/stop	Ultrasonic sensor (30 cm)	Stopping success rate 100% (30/30), avg. distance error ±2.1 cm
Play music	Audio amplifier + command	Playback success rate 100%, avg. delay 0.3 s
Alarm flash	GPIO control	Trigger success rate 100%, latency < 0.1 s
Kick football	PID motion + collision	Success rate 90% (9/10), avg. offset error ±3 cm

Table 2. System-level performance metrics of the vehicle

Metric	Value (Quantitative)
Avg. frame processing time	24.7 ms ± 3.1 ms (~40 fps)
Lap completion rate	90% (9/10)
Avg. lateral deviation	2.2 cm ± 0.8 cm
End-to-end stop latency	0.83 s ± 0.12 s (detection → halt)
Operating range (ultrasonic)	0–7 m reliable; >7 m errors increase sharply

Beyond individual task performance, the system exhibited strong real-time responsiveness and navigation stability. As summarized in Table 2, the average frame processing time was 24.7 ± 3.1 ms (~40 fps), the lap completion rate reached 90 % (9/10 trials), and the mean lateral deviation from the track center was 2.2 ± 0.8 cm. The end-to-end stop latency from obstacle detection to full halt averaged 0.83 ± 0.12 s. These results verify that the proposed vehicle maintains reliable multitask operation and consistent system-level performance in complex scenarios.

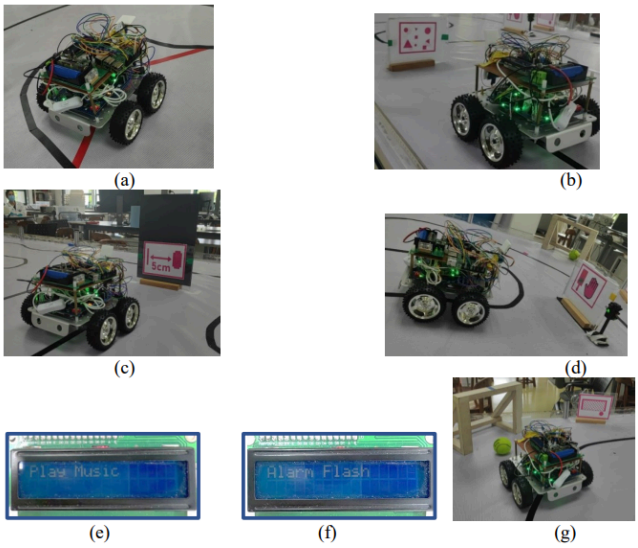

Figure 6. Results of Multiple task. (a): Shortcut; (b): Count shapes; (c): Approach and stop; (d) Traffic light; (e): Play music; (f)Alarm flash; (g): Kick football

DOI: 10.54254/2755-2721/2025.29347

These metrics are summarized in Table 2, which presents the methods and results for each functional node. Representative outcomes are illustrated in Figure 6, demonstrating that the system executed complex sequences—such as ball kicking, light flashing, and distance-based parking—continuously and without noticeable delay. Collectively, these findings validate the high accuracy, responsiveness, and coherence of the proposed system in multitask scenarios, confirming its effectiveness for autonomous navigation in complex environments

4.2. Sensor error and image processing sensitivity analysis

Reliable obstacle avoidance depends on the accuracy of the HC-SR04 ultrasonic sensor, whose range-dependent errors were experimentally characterized over 0–9 m (Figs. 7). Measurements show small deviations (< 5 cm) below 1 m, but increasing errors beyond 5 m, reaching 10–20 cm at 400–600 cm and > 60 cm near 800 cm. The error growth follows

$$d = \frac{v \cdot t}{2}, \quad \Delta d \approx \frac{v}{2}\Delta t \tag{11}$$

with uncertainty modeled by

$$\sigma_d^2 \approx \left(\frac{v}{2}\right)^2 \sigma_t^2 + \left(\frac{t}{2}\right)^2 \sigma_v^2 \tag{12}$$

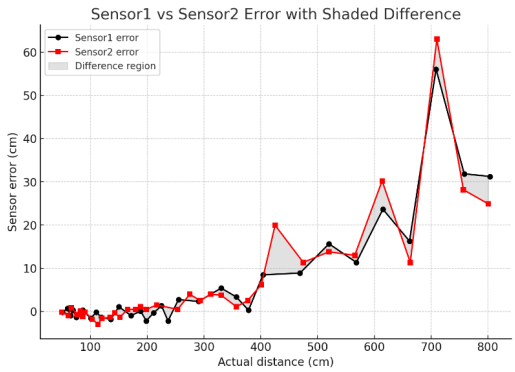

Figure 7. Error curves of two ultrasound sensors across the measurement range

Low SNR and multipath reflections at long range further amplify deviations. The shaded region in Fig.8 highlights inter-sensor variation, minimal below 2 m but expanding at longer distances. Overall, the sensor performs reliably within 0–7 m, while bias and fluctuation grow rapidly beyond 6 m. These results indicate the practical range is shorter than the nominal specification and underline the need for calibration, averaging, and complementary sensing for robust obstacle detection [7].

4.3. Morphological processing for robust line following

A key challenge of the system lies in the sensitivity of vision-based preprocessing to parameter selection. The Gaussian kernel size and V-channel threshold critically affect contour stability for line following and symbol recognition. When set too low, residual noise causes fragmented trajectories (under-segmentation); when too high, the foreground expands excessively, leading to background misclassification (over-segmentation). To quantify this, a parameter sweep was performed, and the results are shown in Fig. 8.

DOI: 10.54254/2755-2721/2025.29347

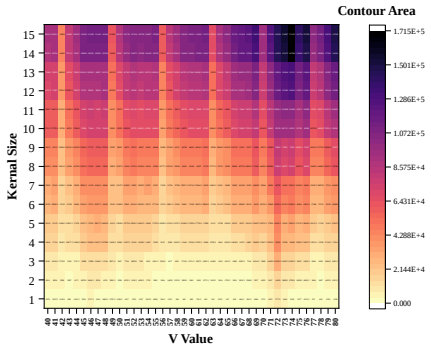

Figure 8. Heatmap of system error under varying preprocessing parameters

The heatmap, varying the V-threshold (40–80) and Gaussian kernel size (1–15), reveals two consistent trends. Larger kernels suppress high-frequency noise and reconnect broken edges, whereas higher thresholds under THRESH_BINARY_INV enlarge the extracted contour. Extreme parameter combinations cause over-segmentation, degrading corner detection and tracking accuracy. Prior studies highlight this trade-off between denoising and structural preservation, showing that excessive smoothing or aggressive thresholds distort geometric integrity [11-14].

As illustrated in Fig. 9, morphological operations play a crucial role in noise removal and shape enhancement during binary image preprocessing. Consistent with multi-scale morphological theory, optimal stability occurs with medium-to-large kernels and moderate thresholds, balancing noise suppression and edge fidelity. The adopted configuration (threshold = 55, kernel = 25 × 25, σ = 5) lies within this stable region, enabling robust contour extraction, right-angle recognition, and accurate trajectory detection [15].

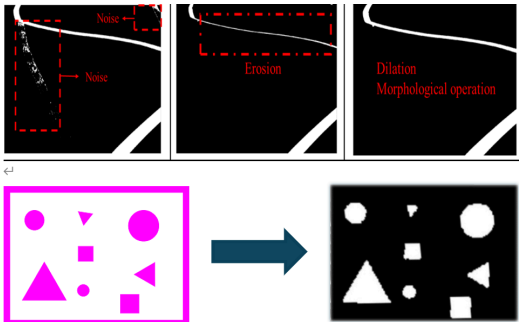

Figure 9. Morphological operations for noise removal and shape enhancement in binary images

The visualizations illustrate trajectory extraction across preprocessing stages. The raw binary image contains scattered noise from reflections and sensor fluctuations. Applying erosion removes isolated artifacts but slightly thins trajectory edges, whereas dilation restores continuity, producing coherent contours. This morphological sequence suppresses background noise while preserving geometric structure, outperforming direct global thresholding, which often causes centroid localization errors. Incorporating erosion and dilation thus stabilizes contour extraction and ensures consistent input for downstream modules, aligning with recent studies highlighting adaptive morphology and illumination-aware thresholding for robust segmentation [11,12,15,16].

Morphological preprocessing is therefore crucial for reliable right-angle detection and perspective-normalized template matching in embedded line-following systems. Its effectiveness is

DOI: 10.54254/2755-2721/2025.29347

further supported by comparisons with prior work. Chen et al. [17] and others [18,19] reviewed autonomous and swarm robotics, noting limitations in multitask execution. Typical line-following robots achieve accurate tracking only under controlled conditions and rarely integrate multitask perception and actuation. For instance, Brigido et al. [19] reported accuracies below 95 % under variable lighting, whereas our system consistently maintains over 96 % across diverse tasks. By combining validated hardware, optimized vision preprocessing, and real-time PID control on a compact Raspberry Pi platform, the proposed design achieves higher versatility and robustness than previous approaches.

5. Conclusion

This study implemented a multitasking autonomous vehicle on a Raspberry Pi platform, integrating ultrasonic ranging, camera-based vision, audio feedback, and bidirectional level shifting. Both component-level validation and system-level experiments confirmed reliable operation. The vehicle achieved high task performance—96.7 % traffic-light recognition, 93.3 % shape counting, 100 % ultrasonic stopping, and 90 % football-kicking success—with overall indicators of 24.7 ms average frame time, 90 % lap completion rate, and 2.2 cm mean lateral deviation.

Despite stable performance, the current perception strategy relies on discrete event detection (e.g., right-angle recognition), limiting smooth trajectory tracking and continuous steering adjustment. Recent studies suggest that lane-slope–based steering estimation can generate continuous control inputs for improved lane keeping and curve following [13].

Future work will integrate event-triggered and continuous control by combining right-angle detection with lane-based steering estimation, incorporating robustness mechanisms such as slope tolerance to handle illumination variation and noise. Learning-based methods will also be explored for navigating dynamic, multi-obstacle environments.

Overall, the results demonstrate that compact, low-cost embedded platforms can reliably integrate heterogeneous modules for multitask navigation, offering practical value for robotics education and research while identifying future directions—continuous steering estimation and sensor fusion—to enhance robustness under real-world conditions.

References

[1] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, “A comprehensive review on applications of Raspberry Pi, ” Computer Science Review, vol. 52, p. 100636, 2024, doi: 10.1016/j.cosrev.2024.100636.

[2] J. Li, "OpenCV-based PID control line following vehicle with object recognition and reaction, " in AIP Conf. Proc., vol. 3144, no. 1, p. 050009, 2024, doi: 10.1063/5.0214211.

[3] R. Linelson, F. R. Saputri, S. D. Wijaya, V. R. Lee, and M. D. Al-Haidar, “A security radar system on a semi-autonomous car based on the ultrasonic sensor, servo motor and Arduino Uno, ” in Proc. 2023 Int. Conf. Smart Cities, Autom. & Intell. Comput. Syst. (ICON-SONICS), Bali, Indonesia, Dec. 2023, pp. 1–6, doi: 10.1109/ICON-SONICS59898.2023.10435222.

[4] J. Sigut, M. Castro, R. Arnay, and M. Sigut, “OpenCV basics: A mobile application to support the teaching of computer vision concepts, ” IEEE Transactions on Education, vol. 63, no. 4, pp. 328–335, Nov. 2020. doi: 10.1109/TE.2020.2993013.

[5] X. Hu and J. Jin, “Embedded digital tuning radio based on stm32, ” Journal of Physics: Conference Series, vol. 2417, no. 1, p. 012002, 2022, doi: 10.1088/1742-6596/2417/1/012002

[6] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. London, U.K.: Pearson, 2018.

[7] N. I. Abdulkhaleq, I. J. Hasan, and N. A. J. Salih, “Investigating the resolution ability of the HC-SR04 ultrasonic sensor, ” IOP Conf. Ser.: Mater. Sci. Eng., vol. 745, no. 1, p. 012043, 2020, doi: 10.1088/1757-899X/745/1/012043

DOI: 10.54254/2755-2721/2025.29347

[8] A. Elen and E. Dönmez, “Histogram-based global thresholding method for image binarization, ” Optik, vol. 274, p. 171814, 2024, doi: 10.1016/j.ijleo.2024.171814.
[9] S. Dormido and A. Visioli, “PID control, ” in Encyclopedia of Systems and Control, 2nd ed., J. Baillieul and T. Samad, Eds. Cham, Switzerland: Springer, 2021, pp. 1–10. doi: 10.1007/978-3-030-44184-5_245.
[10] S. Cao, J. Zhu, Y. Li, and R. Ji, "Recurrent homography estimation using homography-guided image warping and focus transformer, " in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Vancouver, BC, Canada, Jun. 2023, pp. 12790–12799, doi: 10.1109/CVPR52729.2023.01229.
[11] Q. Chen, W. Wang, Z. Wang, H. Jia, and M. Zhao, "Blurred lesion image segmentation via an adaptive scale thresholding network, " Appl. Sci., vol. 15, no. 17, p. 9259, 2025, doi: 10.3390/app15179259.
[12] Z. Yang, S. Zuo, Y. Zhou, J. He, and J. Shi, "A review of document binarization: Main techniques, new challenges, and trends, " Electronics, vol. 13, no. 7, p. 1394, 2024, doi: 10.3390/electronics13071394.
[13] M. T. Gunawan, H. Alwi, and A. R. Al Tahtawi, “The Implementation of Steering Angle Estimation on Miniature Raspberry Pi-based Autonomous Car, ” 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA), Singapore, 2022, pp. 1037–1042, doi: 10.1109/ICIEA54703.2022.10005980.
[14] M. Zhang, J. Wang, X. Cao, X. Xu, J. Zhou, and H. Chen, "An integrated global and local thresholding method for segmenting blood vessels in angiography, " Heliyon, vol. 10, no. 22, p. e38579, 2024, doi: 10.1016/j.heliyon.2024.e38579.
[15] L. Liang, Y. Zhang, C. Liu, K. Wang, X. Zhang, and W. Liu, "Noise-robust image edge detection based on multi-scale automatic anisotropic morphological Gaussian kernels, " PLoS One, vol. 20, no. 5, p. e0319852, May 2025, doi: 10.1371/journal.pone.0319852.
[16] G. Sharma, R. K. Agrawal, and D. S. Chauhan, "An improved morphological approach for noise-robust image segmentation in real-time vision systems, " IEEE Access, vol. 12, pp. 55732–55745, 2024, doi: 10.1109/ACCESS.2024.3389123.
[17] W. Chen, W. Chi, S. Ji, H. Ye, J. Liu, Y. Jia, J. Yu, and J. Cheng, "A survey of autonomous robots and multi-robot navigation: perception, planning and collaboration, " Robot. Auton. Syst., 2024, doi: 10.1016/j.robot.2024.104567.
[18] E. Debie, K. Kasmarik, and M. Garratt, "Swarm robotics: A survey from a multi-tasking perspective, " ACM Comput. Surv., vol. 56, no. 4, Article 49, 2024, doi: 10.1145/3611652.
[19] W. J. H. Brigido and J. M. Parente de Oliveira, "The line follower robot: a meta-analytic approach, " PeerJ Comput. Sci., vol. 11, p. e2744, Mar. 2025, doi: 10.7717/peerj-cs.2744.