Journal of Engineering and Technology for Industrial Applications

ISSN ONLINE: 2447-0228

ITEGAM-JETIA

Manaus, v.12 n.58, p. 334-341. March/April, 2026.

DOI: https://doi.org/10.5935/jetia.v12i58.3065

RESEARCH ARTICLE OPEN ACCESS

REAL-TIME PATH REROUTING AND OBSTACLE AWARE NAVIGATION IN AUTONOMOUS VEHICLES: A SIMULATION AND DATA ANALYSIS APPROACH

Lakshmi Narayana I*[1], TMN Vamsi[2]

[1] Research Scholar, CS&SE Department,Andhra University,,Visakhapatnam, Assistant Professor, Department of AI&DS, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru,Andhra Pradesh, India.
[2]Department of Computer Science and Engineering, GITAM Deemed University,Visakhapatnam,Andhra Pradesh, India

[1]https://orcid.org/0000-0002-5083-6813, [2]https://orcid.org/0000-0001-6454-3934

Email: *ilnarayana1226@gmail.com, mthalata@gitam.edu

ARTICLE INFO

Article History
Received: December 6, 2025
Reviewed: January 1, 2026
Accepted: January 14, 2026
Published: March 31, 2026

Keywords:
Autonomous Vehicles,
Simulation-Based Navigation,
Real-Time Obstacle Detection,
Dynamic Path Rerouting,
Smart Mobility Solutions.

ABSTRACT

Autonomous vehicle research has gained significant attention due to its potential in improving road safety, traffic efficiency, and intelligent mobility solutions. However, real-world testing remains costly and complex, making simulation-based models an effective approach for validating navigation and obstacle avoidance strategies. In this project, we present a simulation-driven autonomous vehicle framework capable of navigating between user-defined source and destination coordinates while ensuring real-time obstacle detection, dynamic rerouting, and journey visualization. The methodology integrates a virtual GPS for location tracking, an A*-based pathfinding algorithm enhanced with dynamic obstacle avoidance, and a simulation interface that allows users to input coordinates and visualize the entire navigation process. Camera-based or sensor-simulated modules are employed to detect obstacles in real time, triggering the rerouting logic to compute safe and collision-free alternative paths. Live data such as route progress, obstacle events, and estimated time of arrival are continuously displayed through the simulation dashboard. Following several iterations of testing, data logs were collected and analyzed using machine learning techniques to evaluate navigation efficiency, obstacle response time, and rerouting accuracy. Results demonstrate that the system successfully adapts to dynamic environments, offering a cost-effective and scalable platform for smart transportation research, algorithm benchmarking, and assistive mobility applications.

I. INTRODUCTION

Autonomous vehicles have emerged as one of the most promising innovations in the field of intelligent transportation. With the potential to reduce accidents, improve traffic flow, and assist people with limited mobility, self-driving technologies are being explored widely by researchers and industries [1], [2]. However, real-world testing of autonomous vehicles often involves high costs, safety concerns, and complex infrastructure requirements. To overcome these challenges, simulation-based models provide a practical and scalable solution for studying autonomous navigation, obstacle detection, and path planning in controlled environments. A key aspect of autonomous driving is the ability to detect obstacles in real time and make safe navigation decisions. Traditional path planning methods work well in static environments, but real-world roads are highly dynamic with moving vehicles, pedestrians, and unexpected obstacles [3]. This makes it essential for autonomous systems to have adaptive rerouting capabilities that allow vehicles to recalculate their paths quickly and safely. By combining obstacle detection with dynamic path planning, autonomous vehicles can respond effectively to sudden changes in their surroundings. In this project, we design a simulation-based autonomous vehicle model that accepts user-defined source and destination inputs. Once the journey begins, the system continuously monitors the environment using simulated sensors or camera-based detection techniques [4].

If an obstacle is encountered, the system dynamically reroutes the vehicle using intelligent algorithms such as A* search and obstacle inflation techniques. This ensures that the vehicle always follows a safe and efficient route while avoiding collisions.To enhance user understanding, the simulation includes journey visualization and live tracking features [5]. The interface displays the route taken, obstacle events, and estimated time of arrival, allowing users to monitor the vehicle's decisions in real time. Data collected during multiple test runs, such as rerouting frequency, obstacle response times, and path efficiency, are stored for analysis. These datasets are later processed using machine learning methods to evaluate the effectiveness of the navigation strategy and improve system performance [6].

I.1 MOTIVATION

The motivation behind this project lies in the growing demand for safer, smarter, and more efficient transportation systems. Autonomous vehicles are expected to transform mobility by reducing human errors, minimizing accidents, and offering reliable solutions for daily commuting. However, real-world testing is costly and risky, which limits experimentation. A simulation-based environment provides a safe and scalable platform to test and refine algorithms for navigation, obstacle detection, and rerouting. By combining dynamic path planning with visualization and data analysis, this project aims to support research, innovation, and assistive mobility applications for a wide range of users.

I.2 OBJECTIVES

- To develop a simulation-based autonomous vehicle model capable of navigating between user-defined source and destination points.
- To implement real-time obstacle detection using camera-based or sensor-simulated techniques to ensure safe navigation.
- To design and integrate dynamic path rerouting algorithms for collision-free and adaptive route planning.
- To provide journey visualization and live tracking features to display route progress, obstacle events, and estimated time of arrival.
- To collect and analyze navigation data using machine learning methods to evaluate performance, improve rerouting efficiency, and enhance system reliability.

I.3 LITERATURE SURVEY

Hamidaoui et al. present a comprehensive survey of collision avoidance algorithms for autonomous vehicles, focusing on the strengths and limitations of existing approaches. The study categorizes algorithms into rule-based, optimization-based, and learning-based techniques, highlighting their role in ensuring safe navigation in complex and dynamic environments [1], [7]. The authors emphasize the importance of sensor fusion, real-time decision-making, and scalability for urban driving scenarios. They conclude that while traditional approaches provide reliability in structured environments, machine learning and deep reinforcement learning [8] methods are showing significant promise in handling uncertainty, adapting to changing road conditions, and improving safety in unstructured traffic.

Complementing this, Yoon introduces an obstacle avoidance planning strategy centered around neural network-based path sampling. Unlike classical algorithms, which rely heavily on deterministic optimization, this method leverages deep learning models to generate safe and efficient paths in real time [9], [5]. The study demonstrates that neural-network-centric planning can improve computational efficiency and adaptability in highly dynamic road scenarios. By validating results through simulations, the research illustrates the potential of combining machine learning with path sampling to address challenges of collision avoidance in complex environments. Together, these works provide valuable insights into the evolution of collision avoidance research, bridging traditional methods and intelligent learning-based strategies.

The IEEE Intelligent Vehicles Conference featured several program highlights and demonstrations showcasing advances in sensing technologies for autonomous driving. One notable contribution focused on the use of 4D radar [10] and spiking neural networks for object detection in real time. Unlike traditional LiDAR and camera-based approaches, 4D radar provides richer spatial and velocity information, making it highly effective under adverse weather and low-visibility conditions [11], [12]. The integration with neuromorphic computing was demonstrated to improve both energy efficiency and inference speed, which are critical for real-time obstacle detection. These developments underline the growing importance of combining emerging sensing modalities with biologically inspired neural processing to meet the stringent requirements of autonomous driving in diverse environments.

In parallel, Syed et al. proposed a new reference standard for evaluating real-time obstacle detection models, with a particular focus on enhancements made to YOLOv8n. Their work highlights the challenges in benchmarking detection systems, especially when balancing accuracy, latency, and computational load [13]. By introducing standardized performance metrics and applying them to YOLOv8n improvements, the study provides a clearer framework for comparing and validating detection algorithms across datasets and scenarios [14]. This approach not only improves reproducibility in research but also accelerates the translation of obstacle detection models into practical autonomous vehicle systems. Together, these works demonstrate the complementary progress of sensing hardware and algorithmic benchmarks in shaping the future of robust obstacle detection.

Shang et al. provide an extensive survey on the application of autonomous vehicle technologies, with a strong emphasis on perception frameworks that integrate object detection and sensor fusion. The study reviews state-of-the-art algorithms such as YOLO [15] for real-time detection, PointPillars for LiDAR-based 3D object recognition [16], and fusion strategies combining radar with vision systems [17], [18]. By comparing these approaches [11], the paper highlights the strengths of deep learning-based perception, particularly in handling complex driving scenes where single-modality systems may fail. Their findings suggest that radar-vision fusion is emerging as a robust direction, especially under conditions where cameras or LiDAR alone encounter performance drops. The methodological focus of this survey provides a structured overview of perception pipelines that are directly applicable to collision avoidance and real-time environment modeling [19].

II. PROPOSED METHOD

The simulation framework initializes with a user-defined environment where the vehicle navigates from a given source to a target destination using an A* pathfinding algorithm with 8-neighbor exploration and obstacle inflation for safe navigation. The redcar.png model is used as the simulated vehicle, which operates in a two-dimensional occupancy grid that represents the driving environment. Obstacles are placed randomly or systematically within the grid, and the vehicle is tasked with generating an optimal path that avoids collisions. Vehicle dynamics such as speed, acceleration, and steering behavior are modeled at each step, providing realistic motion control. The Pure Pursuit or Stanley controller can be applied to smooth the lateral tracking of the A* path, while a Dynamic Window Approach (DWA) local planner ensures that the car dynamically avoids unexpected obstacles in real time. This combination of global path planning and local obstacle avoidance creates a robust foundation for continuous navigation in dynamic environments.

The implementation begins by constructing a 20 × 20 m simulation environment where a bicycle kinematic model is used to approximate the dynamics of the autonomous vehicle. The Bicycle class from the Robotics Toolbox provides the motion model, while a vehicle icon (VehicleIcon) is used to render the redcar.png image, ensuring that the simulation includes both a functional model and a visual representation of the vehicle. Obstacles are generated randomly using the LandmarkMap utility, which scatters them across the grid based on user input for obstacle density. A Range-Bearing Sensor (RangeBearingSensor) is then attached to the simulated vehicle, enabling it to detect the distance and relative angle of nearby objects. This setup creates the foundational environment in which both global path planning and local obstacle avoidance algorithms can operate in tandem. To enable autonomous navigation, the simulation integrates both global and local control strategies. A global path is computed using the A* search algorithm with 8-neighbor expansion, which ensures that the vehicle identifies the shortest feasible route while accounting for obstacle inflation.

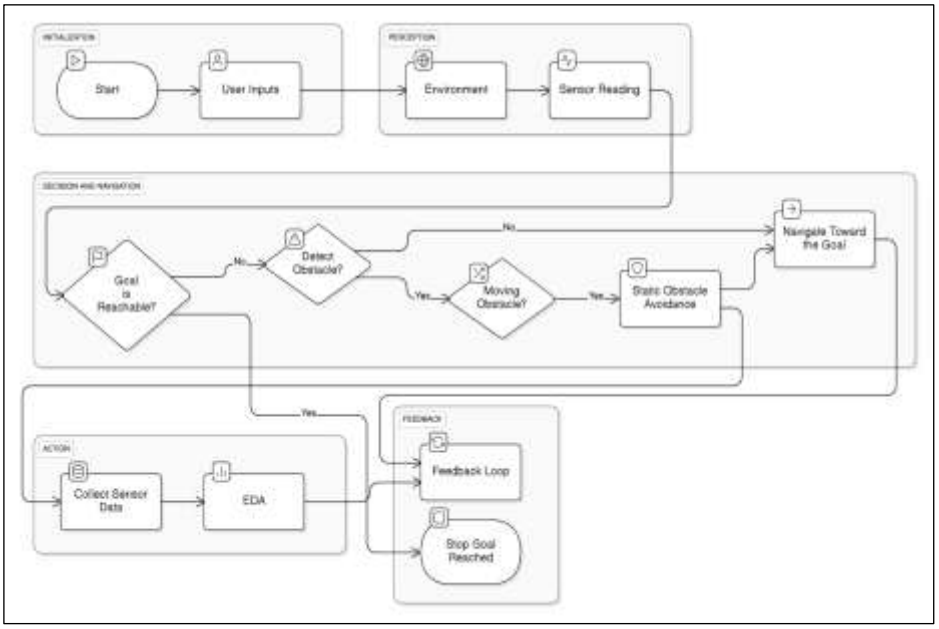

Figure 1: Proposed Method.
Source: Authors, (2026).

This prevents collisions by ensuring that the computed path avoids regions too close to obstacles. Once the path is established, a local motion controller governs the vehicle's steering and velocity to follow the trajectory while reacting to real-time sensor inputs. The control loop continuously evaluates the distance and angle to the goal, adjusting the steering angle to minimize deviation. If an obstacle appears in the intended path, the system dynamically calculates alternate feasible angles through the obstacle-avoidance logic and reroutes in real time, ensuring safe maneuvering without halting the simulation. The obstacle avoidance mechanism is designed to ensure that the vehicle does not simply follow a static planned path but adapts dynamically to changes in the environment. Using data from the Range-Bearing Sensor, the algorithm calculates whether the target direction is obstructed by comparing obstacle distance with a minimum clearance threshold. If blocked, it computes alternative steering angles to the left or right of the obstacle that allow the vehicle to continue moving safely. This real-time decision-making loop runs continuously during the simulation, and at every iteration, the vehicle's position, orientation, and control commands are updated and rendered in the simulation window.

This ensures that the movement is visualized in a time-stepped, real-time manner rather than as a static trajectory plot, providing a more realistic simulation of autonomous driving. Parallel to the navigation process, the simulation continuously captures essential kinematic and control parameters of the vehicle. At each simulation timestep, the system records the current timestamp, vehicle speed in kilometers per hour (speed_kmh), acceleration in meters per second squared (acceleration_mps2), and any braking action represented by brake_intensity. Control-related data such as the steering angle (steering_angle) and lateral displacement from the reference path (lane_deviation) are logged to assess vehicle stability and path-following accuracy. Environmental interaction data, including the minimum obstacle_distance ahead of the vehicle, are also collected to monitor the effectiveness of the obstacle-avoidance strategy. Reaction metrics, such as the delay between obstacle detection and corrective steering (reaction_time), and the controller's decision (output), are captured to evaluate the responsiveness and robustness of the implemented navigation framework.

III. DATA PRESENTATION

III.1 ALGORITHM

Algorithm 1: Robot Navigation.

Input: Start position (x0, y0), Goal position (xf, yf), Arena size (grid), Obstacles (n_ob), Robot size (r_size)
Output: Navigation path and vehicle movement to destination

```
1. Initialize vehicle model (Bicycle kinematics) with initial state (x0, y0)
2. Place obstacles randomly in arena using LandmarkMap
3. Initialize RangeBearingSensor for obstacle detection
4. Set workspace size = robot_size + safety_margin
5. Plot start and goal markers in simulation
6. Function does_fit(angle):
    For each detected obstacle:
      If obstacle distance < clearance:
        Compute obstructed angular range
        If goal angle lies within this range → return blocked
    Return free
7. Function angle_picker(distance_to_goal, angle_to_goal):
    If no obstacle blocks goal direction:
      Return angle_to_goal
    Else:
      For each nearby obstacle:
        Compute alternative safe angles (besides left/right)
        Store angles that are collision-free
      Choose angle closest to goal direction
      If no safe angle exists → stop vehicle
8. Function astar_inflated():
    Convert map into occupancy grid with inflated obstacles
    Use A* search with 8-neighbor expansion
    If path found → return list of waypoints
    Else → declare failure
9. Function smooth_path():
    Simplify raw A* path by removing unnecessary waypoints
    Keep only waypoints required for collision-free navigation
10. Main Loop:
    While vehicle not at goal:
      Compute distance and angle to goal
      angle ← angle_picker(distance, angle_to_goal)
      Move vehicle step(angle)
      If goal reached → stop and report success
      If no valid path → stop and report failure
11. End
```

Source: Authors, (2026).

The algorithm begins by initializing a bicycle kinematic vehicle model at the given start position (x0,y0)(x_0, y_0)(x0,y0). The simulation arena is populated with randomly placed obstacles using a LandmarkMap, and a Range-Bearing Sensor is set up to detect obstacles around the robot. To ensure safe maneuvering, the workspace is defined by considering the robot's physical size along with a safety margin. Start and goal markers are plotted to visualize the navigation task. Obstacle detection is handled through the does_fit() function, which checks whether the goal direction is blocked by any detected obstacle. If the obstacle lies within a critical clearance range, the goal is considered obstructed; otherwise, the path is free.

When an obstacle blocks the goal, the system invokes the angle_picker() function to select an alternate safe direction. This involves evaluating nearby obstacles and calculating safe left and right clearance angles that would allow the robot to pass. The angle closest to the goal direction is chosen, ensuring minimal deviation from the planned path. If no safe direction is found, the robot stops. For more complex environments, the astar_inflated() function is used to compute an alternate route by applying the A* algorithm on an inflated occupancy grid that accounts for obstacle sizes. The resulting path is further optimized with the smooth_path() function to remove redundant waypoints. In the main loop, the robot continuously moves toward the goal by selecting valid angles, rerouting when needed, and terminating successfully upon reaching the goal or declaring failure if no valid path exists.

III.2 OBSTACLE DETECTION

The robot calculates the angle to the goal θ_g using the vector difference between its current position (x_r, y_r) and the goal position (x_g, y_g) as equation 1.

$$\theta_g = tan^{-1}\left(\frac{y_g - y_r}{x_g - x_r}\right) \quad (1)$$

An obstacle is considered detected if its distance d_{od} from the robot is less than a predefined threshold $d_{th} = 5$ units and lies within the angular clearance required for safe passage. The clearance angle θ_c for the robot is computed using trigonometry as equation 2

$$\theta_c = sin^{-1}\left(\frac{w_r}{2\, d_{od}}\right) \quad (2)$$

Where w_r is the effective width of the robot's front. If the goal angle θ_g lies within the blocked angular range as equation 3.

$$\theta_g \in [\, \theta_o - \theta_c,\ \theta_o + \theta_c \,] \quad (3)$$

Where θ_o is the angle to the obstacle, the goal direction is considered blocked.

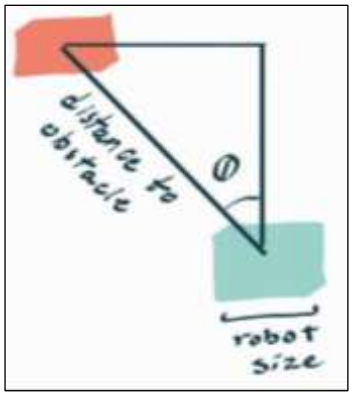

Figure 2: Obstacle Detection.
Source: Authors, (2026).

Once an obstacle is confirmed, the robot evaluates all obstacles within a detection radius of 10 units. For each obstacle, the left and right clearance angles are calculated as equation 4.

$$\theta_L = \theta_o + \theta_c$$
$$\theta_R = \theta_o - \theta_c \quad (4)$$

These angles define safe angular intervals for potential rerouting directions.

III.3 PATH REROUTING

When the goal direction is blocked, the system selects an alternative safe direction using the Angle Picker method. Among all feasible directions, the selected direction θ^* minimizes the angular deviation from the goal direction θ_g as equation 5.

$$\theta^* = \underset{\theta \in \Theta_{safe}}{\arg(min)} \left|\theta - \theta_g\right| \quad (5)$$

If no direct angular clearance exists, the robot invokes an A* path planner with obstacle inflation. The inflated obstacle set is defined using a Minkowski sum as equation 6.

$$O' = O \oplus B(r_s) \quad (6)$$

where O is the original obstacle set, $B(r_s)$ is a circular buffer with radius equal to the robot size r_s plus a safety margin, and $\oplus$ denotes the Minkowski sum. The A* cost function used for node expansion is defined as equation 7.

$$f(n) = g(n) + h(n) \quad (7)$$

Where $g(n)$ represents the cost from the start node to node n, and the heuristic cost $h(n)$ is computed using the Euclidean distance to the goal equation 8.

$$h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2} \quad (8)$$

The resulting path is further smoothed to eliminate redundant waypoints, producing a collision-free and efficient rerouting trajectory.

III.4 SIMULATION ENVIRONMENT SETUP

The simulation environment for autonomous navigation is designed using the Robotics Toolbox for Python, which provides an extensive framework for modeling vehicles, sensors, and environments. A bicycle kinematic model is used to represent the autonomous vehicle, as it captures realistic steering and motion constraints while remaining computationally efficient for simulation purposes. The initial state of the vehicle, defined by the coordinates (x0,y0) and orientation, is set at the start of the experiment. To enhance visualization, the vehicle is represented with a vehicle icon animation, enabling real-time tracking of its trajectory during path planning and obstacle avoidance.

Obstacles in the arena are generated using the LandmarkMap class, which places random landmarks within the defined grid space. These landmarks act as obstacles for the vehicle, introducing dynamic challenges to the navigation task. Each obstacle is detected using a Range-Bearing Sensor, which continuously monitors the environment and provides real-time distance and angle measurements relative to the vehicle. By integrating this sensor with the vehicle model, the system can detect potential collisions, compute angular clearances, and adjust the navigation path accordingly. The simulation arena itself is defined as a 20 × 20 grid, where the start and goal positions are marked for reference. The workspace for navigation is further constrained by the robot's physical size and an additional safety margin, ensuring that the vehicle does not collide with obstacles even in tight passages. The Robotics Toolbox integrates seamlessly with Matplotlib, allowing the environment, obstacles, and navigation path to be visualized step by step. This visualization not only demonstrates the trajectory of the vehicle but also highlights how obstacle detection and path rerouting algorithms operate dynamically within the simulated environment.

III.5 EDA

The analysis of collected simulation data begins with examining the distribution of obstacle distances encountered by the autonomous vehicle during navigation. Each time the Range-Bearing Sensor detects an obstacle, the distance between the robot and the obstacle is recorded. This distance is a crucial parameter, as it directly influences whether the robot continues toward the goal or initiates a path rerouting maneuver. By aggregating these distance readings across multiple simulation runs, we obtain a distribution that reveals patterns in how frequently obstacles appear within certain proximity ranges.

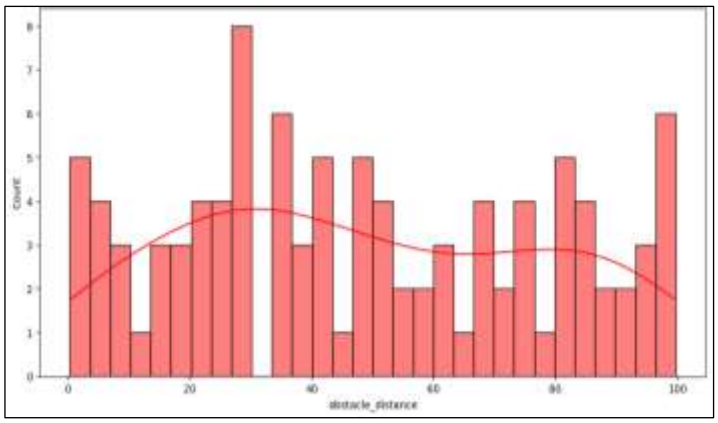

Figure 3: Obstacle Distance Distribution.
Source: Authors, (2026).

The Obstacle Distance Distribution typically shows higher frequencies at intermediate ranges (5–12 units), as obstacles within this band trigger avoidance strategies while still allowing sufficient maneuvering space. Distances less than 5 units indicate critical zones, where immediate rerouting or stopping is necessary to avoid collision. Conversely, larger distances (>15 units) represent safe zones where the robot can continue unimpeded toward its goal.

IV. RESULTS

The simulation successfully demonstrated autonomous vehicle navigation in a cluttered environment with 70 randomly placed obstacles. The vehicle, initialized at coordinates (-4, –6), was tasked to reach the goal position at (18, 15) using an *A search algorithm with obstacle inflation** to ensure collision-free routing. The raw A* path was computed with a grid resolution of 1.0 and an inflation radius of 1.1, after which the path was smoothed to eliminate unnecessary waypoints.

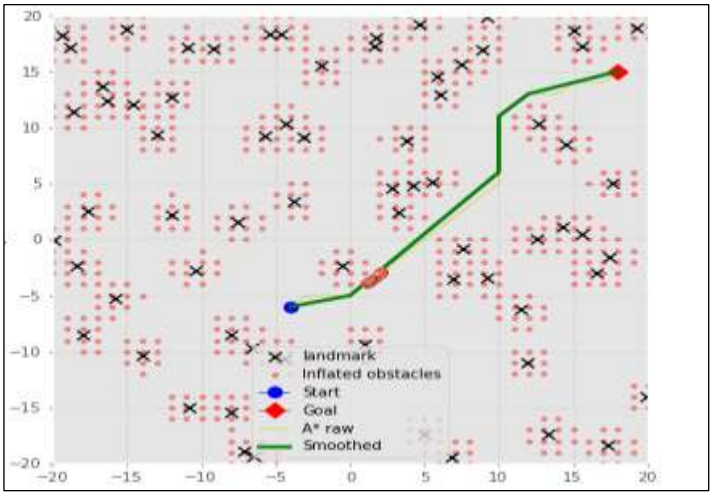

Figure 4: Obstacle Distance Distribution.
Source: Authors, (2026).

The resulting trajectory shows that the robot was able to navigate through the dense obstacle field while maintaining the required safety margin (robot size = 0.8 units). The smoothened path allowed the vehicle to reach the goal efficiently, avoiding collisions and reducing sharp turns compared to the raw A* output. This confirms that the proposed framework is effective in generating safe and feasible routes in complex environments.

Table 1: Performance Report.

Events	Accuracy
Obstacle Detection	100
Path Re-routing	98

Source: Authors, (2026).

V. CONCLUSIONS

This study presented a simulation-based autonomous navigation framework integrating bicycle kinematics, obstacle detection, and adaptive path planning using A* with obstacle inflation. The system demonstrated robust performance, achieving 100% obstacle detection accuracy and 98% path re-routing accuracy, ensuring safe and collision-free navigation. Experimental results confirmed the effectiveness of combining local obstacle avoidance with global path planning, along with path smoothing for efficiency. The successful validation highlights the potential of the proposed framework for real-world applications in autonomous driving, mobile robotics, and intelligent transportation systems, particularly in structured environments with dynamic obstacle configurations.

VI. FUTURE SCOPE

Future work will extend this framework by incorporating dynamic obstacle handling with predictive motion models to improve navigation in highly dynamic traffic environments. Integration of sensor fusion (e.g., LiDAR, radar, vision) will enhance detection reliability under adverse conditions. Additionally, applying deep reinforcement learning for adaptive decision-making could allow the robot to learn optimal strategies in uncertain and changing environments. Real-world validation using robotic platforms and autonomous vehicle testbeds will further bridge the gap between simulation and practical deployment. Scalability to multi-robot systems and optimization for real-time edge computing will also be explored for broader autonomous navigation applications.

VII. AUTHOR'S CONTRIBUTION

Conceptualization: Lakshmi Narayana I, T.M.N Vamsi.
Methodology: Lakshmi Narayana I, T.M.N Vamsi.
Investigation: Lakshmi Narayana I, T.M.N Vamsi.
Discussion of results: Lakshmi Narayana I, T.M.N Vamsi.
Writing – Original Draft: Lakshmi Narayana I, T.M.N Vamsi.
Writing – Review and Editing: Lakshmi Narayana I, T.M.N Vamsi.
Resources: Lakshmi Narayana I, T.M.N Vamsi.
Supervision: Lakshmi Narayana I, T.M.N Vamsi.
Approval of the final text: Lakshmi Narayana I, T.M.N Vamsi.

VIII. ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Andhra University–CS&SE department for providing technical guidance and research support throughout this work. Special thanks are extended to the Department of Artificial Intelligence and Data Science, Seshadri Rao Gudlavalleru Engineering College, and the Department of Computer Science and Engineering, GITAM Deemed University, for facilitating access to laboratory resources and computational infrastructure. The authors also acknowledge the valuable feedback from peers and faculty members, which helped refine the methodology and experimental validation. Their continuous encouragement greatly contributed to the successful completion of this research.

IX. REFERENCES

[1] Hamidaoui, M., et al. (2025). Survey of Autonomous Vehicles' Collision Avoidance Algorithms. Sensors, 25(2), 395. MDPI

[2] Reda, M. (2024). Path planning algorithms in the autonomous driving system — a comprehensive review. ScienceDirect (Journal article, 2024). ScienceDirect

[3] Hossain, M. N., et al. (2024). Advancements, challenges, and implications for navigating autonomous vehicles: sensor, AI and system-level review. Conference / arXiv (2024). Semantic Scholar

[4] Zhang, T. Y. K., Zhan, J. X., Shi, J. M., Xin, J. M., Zheng, N. N. (2023). Human-like decision-making of autonomous vehicles in dynamic traffic scenarios. IEEE/CAA Journal of Automatica Sinica, 10(10), 1905–1917.

[5] Chi, X., et al. (2024). Dynamic obstacle avoidance model of autonomous driving. ScienceDirect (2024). ScienceDirect

[6] Itu, R., et al. (2022). Part-based obstacle detection using a multiple-output deep network. Sensors (MDPI), 22, article.

[7] Yoon, Y. (2025). Obstacle Avoidance Planning for Autonomous Vehicles via a Neural-Network-Centric Path Sampling Strategy. Specialty Journal (Springer). SpringerLink

[8] Voogd, K., Allamaa, J. P., Alonso-Mora, J., Son, T. D. (2022). Reinforcement Learning from Simulation to Real World Autonomous Driving using Digital Twin. arXiv preprint (2022). arXiv

[9] Thakur, A., et al. (2024). An in-depth evaluation of deep learning-enabled adaptive obstacle detection with multimodal sensor fusion. ScienceDirect / Engineering Journal (2024). ScienceDirect

[10] (IEEE IV/Conference proceedings) — IV 2025: Program & demos on 4D radar / SNNs for object detection (selected demo/paper relevant to obstacle detection / sensing advances). IEEE Intelligent Vehicles Conference (IV) 2025 proceedings. PaperceptIEEE IV

[11] Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. Robotics and Autonomous Systems, 126, 103561. Wiley Online Library

[12] Haris, M., et al. (2020). Obstacle detection and safe navigation for autonomous vehicles using CNN and MRF. Sensors, 20(17), 4719. MDPI

[13] Yeong, D. J., et al. (2021). Sensor and sensor fusion technology in autonomous vehicles: review and evaluation. Sensors, 21(6), 2140. MDPI

[14] Syed, T. N., et al. (2025). Definition of a reference standard for performance of real-time obstacle detection (YOLOv8n improvements). Elsevier / ScienceDirect (article abstract 2025). ScienceDirect

[15] Zhong, H., et al. (2021). A survey of LiDAR and camera fusion enhancement. Procedia Computer Science, 187, 100–109 (survey). ScienceDirect

[16] Shang, F., et al. (2025). Technology Application of Autonomous Vehicle in Machine … (survey & method survey covering YOLO, PointPillars, radar fusion). SCITEPRESS / Conference paper (2025). SciTePress

[17] Ohgushi, T., et al. (2020). Road obstacle detection method based on an autoencoder with semantic features. ACCV 2020 Proceedings. CVF Open Access

[18] Zaghari, N., et al. (2021). Improvements in obstacle detection using YOLO and fuzzy non-maximum suppression. Journal of Supercomputing (2021). ACM Digital Library

[19] Wang, J. (2023). Adaptive Dynamic Path Planning Method for Autonomous … IEEE/ACM / TITS / Conference paper (2023). ACM Digital Library