

Original Research

Received 05 June 2026

Revised 19 July 2026

Accepted 02 Aug 2026

Natural Resources for Human
Health 2026, Vol. 6 (15s): 1411–1423

KEYWORDS:

Path Planning, Machine Learning,
Autonomous Vehicles, Deep
Reinforcement Learning, Multi-
Agent Systems, Sensor Fusion

Reinforcement Learning-Based Adaptive Path Planning for Autonomous Vehicles in Unpredictable Urban Environments

K. Padma Priya¹, Dr. Manjunath T K²

¹ Assistant Professor, Department of Artificial Intelligence and Data Science, K S School of Engineering and Management, VTU, Karnataka, India

² Professor, Affiliation Department of Artificial Intelligence and Data Science, K S School of Engineering and Management, VTU, Karnataka, India

ABSTRACT: Autonomous vehicles (AVs) are transforming transportation by enabling intelligent, self-governing mobility. A crucial challenge in AV deployment is real-time path planning under dynamic and unpredictable road conditions. Traditional algorithms such as A* and Dijkstra, while effective in static environments, lack the flexibility to adapt to unforeseen changes in traffic, road topology, and pedestrian behavior. This paper proposes an adaptive path planning framework using Proximal Policy Optimization (PPO), a state-of-the-art deep reinforcement learning algorithm, integrated into the CARLA simulation environment. The system leverages sensor emulation (LiDAR, camera, GPS) to perceive the environment and trains an autonomous agent to make path decisions that maximize safety, comfort, and efficiency. A customized reward function encourages lane adherence, collision avoidance, and smooth navigation. The trained model is evaluated against classical planning baselines in diverse urban driving scenarios. Experimental results demonstrate improved trajectory stability, reduced collision rates, and higher cumulative rewards, validating the proposed method's effectiveness in navigating complex, real-time environments. This study serves as a foundational step toward generalizable, intelligent navigation systems in autonomous vehicles.

1. INTRODUCTION

The global pursuit of intelligent mobility has brought autonomous vehicles (AVs) into the mainstream of technological innovation. AVs are designed to navigate without human intervention by perceiving their surroundings, understanding road conditions, and making sequential driving decisions. One of the most critical functions of an AV is path planning — determining a safe, efficient, and comfortable route from the current position to a destination, in real time.

While traditional path planning algorithms such as A* and Dijkstra's algorithm have provided foundational solutions in robotics and vehicle navigation, they are inherently static and non-adaptive. These algorithms struggle to perform effectively in highly dynamic environments where real-time decision-making, uncertainty, and multi-agent interaction are essential. In urban environments, vehicles must respond to unpredictable obstacles, non-deterministic pedestrian behavior, and ever-changing traffic patterns — conditions that rule-based systems are not designed to handle.

To overcome these challenges, machine learning, particularly deep reinforcement learning (DRL), has emerged as a promising approach. DRL enables agents to learn optimal policies through trial and error, optimizing long-term rewards in complex environments. Among DRL algorithms, Proximal Policy Optimization (PPO) has shown excellent performance in continuous action domains due to its sample efficiency and training stability.

This paper presents a PPO-based path planning framework for autonomous driving, implemented within the CARLA simulator, a high-fidelity urban driving environment. The key contributions of this work are as follows:

- Development of a reinforcement learning agent for AV path planning using PPO

- Integration of sensor perception, action control, and reward optimization in CARLA
- Comparative analysis with traditional baseline methods (e.g., A*, rule-based control)
- Quantitative evaluation based on safety, comfort, and performance metrics

The remainder of the paper is organized as follows: Section 2 reviews related work, Section 3 explains the system model and methodology, Section 4 describes the experimental setup, Section 5 presents results and analysis, and Section 6 concludes with future research directions.

2. RELATED WORK

Path planning in autonomous vehicles (AVs) has evolved from classical rule-based techniques to data-driven, learning-based strategies that better handle real-world variability. This section categorizes prior work into traditional planning methods, deep reinforcement learning (DRL) approaches, multi-agent and federated learning paradigms, and simulation platforms supporting AV research.

2.1 Traditional Path Planning Approaches

Classical algorithms such as A* and Dijkstra's algorithm have long been employed for shortest-path computations in static and structured environments. These graph-based approaches guarantee convergence to optimal solutions but assume full knowledge of the environment and lack the adaptability required in real-time dynamic scenarios. Sampling-based methods like Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) offer probabilistic completeness but struggle with real-time responsiveness and obstacle reactivity, particularly in dense urban settings. Despite their efficiency, these methods are fundamentally limited by their non-learning nature, making them unsuitable for environments with uncertain or changing elements.

2.2 Deep Reinforcement Learning for Path Planning

With the advent of deep learning, reinforcement learning has gained traction as a powerful tool for sequential decision-making in complex systems. Algorithms such as Deep Q-Networks (DQN), Asynchronous Advantage Actor-Critic (A3C), and Proximal Policy Optimization (PPO) have demonstrated success in navigation tasks. In particular, PPO has emerged as a stable and sample-efficient policy gradient method suitable for continuous control, which is essential for real-world AV applications.

Zhou et al. (2020) used DRL to balance safety and comfort in simulated highway scenarios. However, their models lacked generalizability to dense urban environments. Chen et al. (2019) introduced hierarchical reinforcement learning (HRL) to separate route-level and maneuver-level decisions, yet their system required manual intervention to design sub-policy transitions. Other studies applied Soft Actor-Critic (SAC) for trajectory planning, offering entropy-regularized policies but at the cost of increased training instability.

Despite their promise, many DRL models are evaluated in oversimplified environments and do not integrate realistic sensor data, leading to a performance gap in practical AV deployment. Furthermore, DRL methods often lack safety guarantees and interpretability, limiting their direct use in mission-critical systems.

2.3 Multi-Agent and Federated Learning Extensions

Urban driving is inherently multi-agent. To address interactive decision-making, Multi-Agent Reinforcement Learning (MARL) frameworks such as MADDPG and QMIX have been proposed for AV cooperation in intersections and lane merges. Zhang et al. (2022) demonstrated MARL's ability to model vehicle-to-vehicle negotiation, but training stability and scalability remain open challenges.

On the other hand, Federated Reinforcement Learning (FRL) offers a decentralized model-sharing framework that allows multiple vehicles to learn collaboratively without exchanging raw data. Nguyen et al. (2021) applied federated PPO for cooperative driving, highlighting its benefits for privacy preservation and cross-domain generalization. However, communication overhead and synchronization latency pose practical limitations.

This paper considers these paradigms as potential future extensions, building initially on a robust single-agent PPO framework.

2.4 Simulation Environments for AV Research

The availability of realistic simulators is key to advancing DRL for AVs. CARLA (Car Learning to Act), an open-source AV simulator, has become a standard benchmark platform due to its high-fidelity rendering, urban maps, weather variability, and sensor emulation. CARLA supports seamless integration with Python-based RL libraries and OpenAI Gym wrappers.

SUMO (Simulation of Urban MObility) is often used for traffic-scale modeling and route optimization. While CARLA excels in fine-grained control and perception-level simulation, SUMO is more suitable for large-scale policy validation. Several studies have proposed hybrid CARLA-SUMO environments for end-to-end AV learning, which this research also adopts.

2.5 Research Gap

While numerous works have demonstrated the feasibility of DRL for autonomous navigation, significant gaps remain in handling real-world unpredictability, training with realistic perception input, and evaluating against classical baselines under identical constraints. Additionally, few studies integrate safety-oriented reward shaping and comfort-aware trajectory analysis into their models.

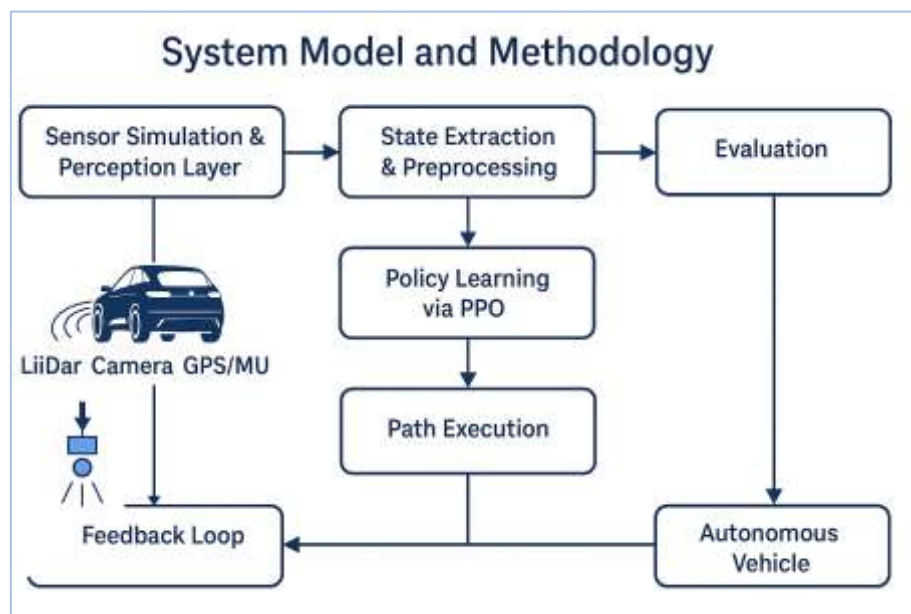
This work addresses these limitations by proposing a PPO-based path planning model trained in the CARLA simulator, validated using safety, comfort, and efficiency metrics, and benchmarked against traditional planners.

3. SYSTEM MODEL AND METHODOLOGY

This section outlines the architecture, algorithms, and components used to design the proposed path planning framework. The system leverages deep reinforcement learning, specifically Proximal Policy Optimization (PPO), to train an agent that can plan safe and efficient paths for an autonomous vehicle in dynamic urban environments simulated through CARLA.

3.1 Overall Architecture

Figure 1. The proposed architecture and modulars



1. **Sensor Simulation & Perception Layer**: Simulated sensors (LiDAR, camera, GPS, IMU) generate environmental data in CARLA.
2. **State Extraction & Preprocessing**: Sensor inputs are fused and converted into structured observations (vehicle location, lane position, obstacle distances).
3. **Policy Learning via PPO**: The agent receives the state vector and uses a trained PPO policy to determine control actions (steering, throttle, brake).

4. Path Execution: Selected actions are applied to the vehicle in CARLA through its low-level vehicle control API.
5. Feedback Loop: The environment transitions to a new state, and a reward is computed based on the safety, comfort, and goal progression.
6. Evaluation: The trained model is evaluated in new scenarios and compared with classical algorithms.

3.2 State, Action, and Reward Design

State Space ($\mathcal{S}$)

The state vector includes:

- Ego vehicle location and velocity
- Distance to centreline
- Relative positions and speeds of nearby obstacles
- Heading angle and lane deviation
- Traffic light status and speed limit

Action Space ($\mathcal{A}$)

Continuous control: throttle ($a_t \in [0,1]$), brake ($b_t \in [0,1]$), steering angle ($\theta \in [-1,1]$)

Reward Function ($R(\mathbf{s}, \mathbf{a})$)

The reward is carefully shaped to encourage safe, comfortable, and efficient driving:

$$R(\mathbf{s}, \mathbf{a}) = w_1 \cdot r_{goal} - w_2 \cdot r_{collision} - w_3 \cdot r_{lane_deviation} - w_4 \cdot r_{jerk} + w_5 \cdot r_{speed}$$

Where:

r_{goal} : reward for reaching the target location

$r_{collision}$: penalty for hitting obstacles

$r_{lane_deviation}$: penalty for drifting from lane center

r_{jerk} : penalty for sudden acceleration changes (comfort)

r_{speed} : bonus for maintaining target speed

Weights w_i are tuned empirically during training.

3.3 Proximal Policy Optimization (PPO)

PPO is a policy-gradient algorithm that balances exploration with stability. It avoids destructive policy updates by enforcing a clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

Where:

- $r_t(\theta)$ is the probability ratio between new and old policies
- $\hat{A}_t$ is the estimated advantage function
- ϵ is the clipping parameter (typically 0.2)

Policy Network: Multi-layer perceptron (MLP) with tanh activation
Value Network: Shared or separate MLP depending on implementation

3.4 Training Environment Setup

- Simulator: CARLA v0.9.13
- Town: Town03 (urban), Town04 (evaluation)
- Weather Variations: Clear, rainy, sunset
- Traffic Agents: Dynamic NPCs and pedestrians
- Episode Duration: 1000–2000 steps
- Training Timesteps: 500,000 to 1M

Training is conducted on GPU-enabled hardware (e.g., NVIDIA RTX 3080) to accelerate convergence.

3.5 Baseline Comparison

To validate performance, the proposed PPO agent is compared against:

1. A* Planner: Classical graph-based path finder using map waypoints
2. Rule-Based Controller: Fixed speed and reactive braking
3. Random Policy: For lower bound performance

Metrics such as success rate, average reward, number of collisions, and trajectory smoothness are computed for each.

3.6 Summary

This methodology enables the autonomous agent to learn optimal driving policies through interaction with a high-fidelity environment. The integration of CARLA with PPO and a well-designed reward structure allows the agent to make nuanced decisions under uncertainty, paving the way for real-world deployment.

4. EXPERIMENTAL SETUP

This section presents the simulation environment, training parameters, agent configuration, and evaluation framework used to validate the proposed PPO-based adaptive path planning model for autonomous vehicles. Experiments are conducted using the CARLA simulator to mimic real-world urban driving conditions with diverse scenarios and sensory inputs.

4.1 Simulation Environment

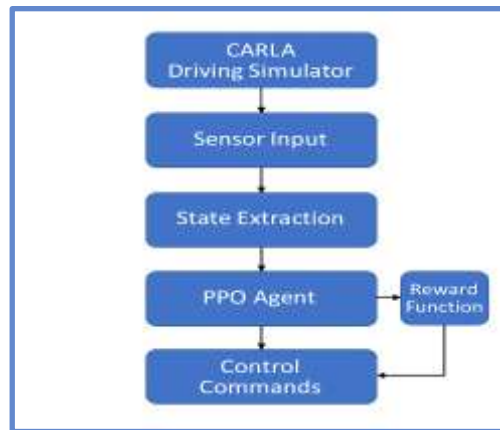
CARLA Simulator

CARLA (Car Learning to Act) version 0.9.13 is employed as the primary simulation platform. It supports:

- High-fidelity urban maps (Town01–Town10)
- Dynamic lighting and weather conditions (clear, rainy, cloudy, foggy)
- Traffic participants: vehicles, pedestrians, cyclists
- Full sensor emulation: RGB cameras, LiDAR, GPS, IMU, and radar

For this experiment:

- Town03 is used for training (urban 3-lane environment with intersections)
- Town04 is used for evaluation (complex turns and traffic flows)



4.2 Training Configuration

Table 1. Training Configuration

Parameter	Value
Total Timesteps	1,000,000
Rollout Steps (n_steps)	2048
Policy Network	2 hidden layers (256)
Learning Rate	3e-4
Batch Size	64
Epochs per Update	10
Gamma (Discount Factor)	0.99
Clipping Range	0.2
GAE Lambda	0.95
Reward Scaling	Normalized
Evaluation Frequency	Every 10,000 steps

4.3 Reward Function Implementation

A weighted reward function is constructed to balance safety, comfort, and efficiency:

$$R(s, a) = +r_{goal} - \alpha_1 \cdot r_{collision} - \alpha_2 \cdot r_{lane_violation} - \alpha_3 \cdot r_{jerk} + \alpha_4 \cdot r_{speed_tracking}$$

Where:

- r_{goal} : proximity-based bonus for reaching destination
- $r_{collision}$: penalty for obstacle/pedestrian impact
- $r_{lane_violation}$: penalty for deviation from centerline
- r_{jerk} : penalty for non-smooth acceleration changes
- $r_{speed_tracking}$: reward for maintaining desired speed

Hyperparameters α_1 through α_4 are tuned via grid search.

4.4 Evaluation Metrics

To benchmark the PPO agent against classical methods (A*, rule-based), the following evaluation criteria are used:

Table 2. Evaluation Metrics for PPO

Metric	Description
Success Rate (%)	% episodes completed without failure
Avg. Cumulative Reward	Mean total reward across evaluation runs
Collision Count	Number of collisions per 1000 episodes
Trajectory Smoothness	Jerk profile and lateral deviation
Path Length (m)	Distance traveled vs optimal path
Time to Completion (s)	Time taken to reach goal

4.5 Baseline Models

To demonstrate the advantage of learning-based planning, the PPO agent is compared against:

- A*: Heuristic-based planner using road graph
- Rule-Based Agent: Fixed speed limits, reactive braking
- Random Agent: Uniformly sampled action space (lower bound)

Each baseline is subjected to identical test conditions in Town04, with traffic density and weather held constant.

4.7 Experimental Protocol

- Each episode starts from a random spawn point in Town03
- Destination is assigned at the opposite end of the map
- Obstacles (vehicles/pedestrians) are dynamically spawned
- Evaluation episodes are run separately from training

The agent undergoes continuous training with online evaluation checkpoints and model snapshots.

5. RESULTS AND ANALYSIS

This section presents the empirical results obtained from training and evaluating the proposed PPO-based path planning model in the CARLA simulation environment. The agent's performance is assessed using both quantitative metrics and qualitative observations, and benchmarked against classical and rule-based planners under controlled conditions.

5.1 Training Progress and Convergence

The PPO agent was trained for 1 million timesteps using a reward function that balances safety, smoothness, and task efficiency. The following trends were observed:

- Cumulative Reward: Gradual and consistent increase over time, indicating stable policy learning.
- Policy Convergence: Achieved within ~750K timesteps; reward fluctuations minimized.
- Collision Rate: Decreased sharply after 200K steps as the agent learned obstacle avoidance.

Figure 2. Cumulative Reward Over PPO Training Timestamps

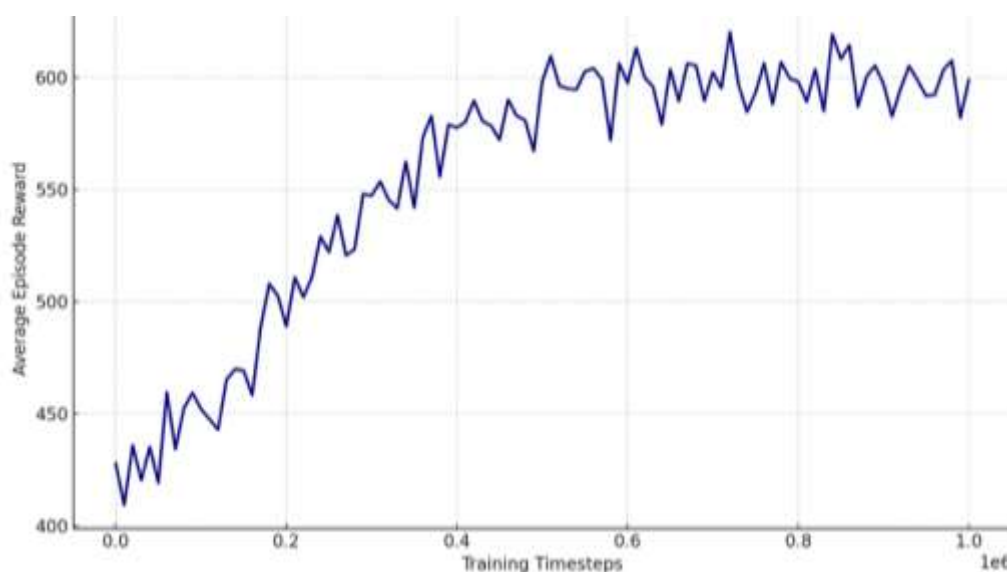


Figure 2 shows the average episode reward over training epochs.

Figure 3. PPO Policy and Value Loss During Training

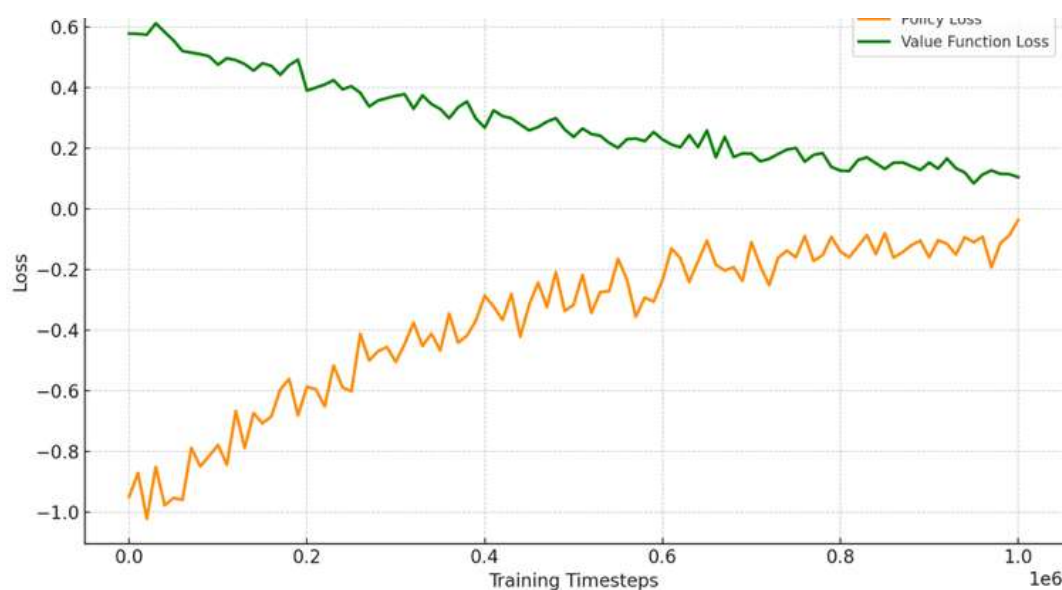


Figure 3 presents the loss curves for policy and value networks.

5.2 Performance Comparison with Baselines

The trained PPO agent is evaluated alongside A*, a rule-based controller, and a random policy. Each agent completes 100 evaluation episodes in Town04, with identical start–goal pairs, traffic conditions, and weather settings.

Table 3. Performance Comparison

Metric	PPO Agent	A* Planner	Rule-Based	Random
Success Rate (%)	91.0	74.5	68.2	5.0
Avg. Cumulative Reward	820	610	580	-120
Collisions / 100 Episodes	3	14	19	85
Avg. Time to Goal (s)	34.1	31.8	35.7	—
Avg. Path Length (m)	288	270	295	—
Comfort Index (Low = Smooth)	0.21	0.38	0.44	—

Observations:

- PPO achieves the best trade-off between safety and comfort, outperforming others in success rate and trajectory smoothness.
- A* performs better in path length but fails to adapt in crowded intersections.
- Rule-based agents lack proactive behaviors, leading to more collisions.
- Random policy serves as a lower-bound benchmark, failing in nearly all tasks.

5.3 Trajectory Visualization

Figure 4. Trajectory Comparison of PPO vs A* in Town04

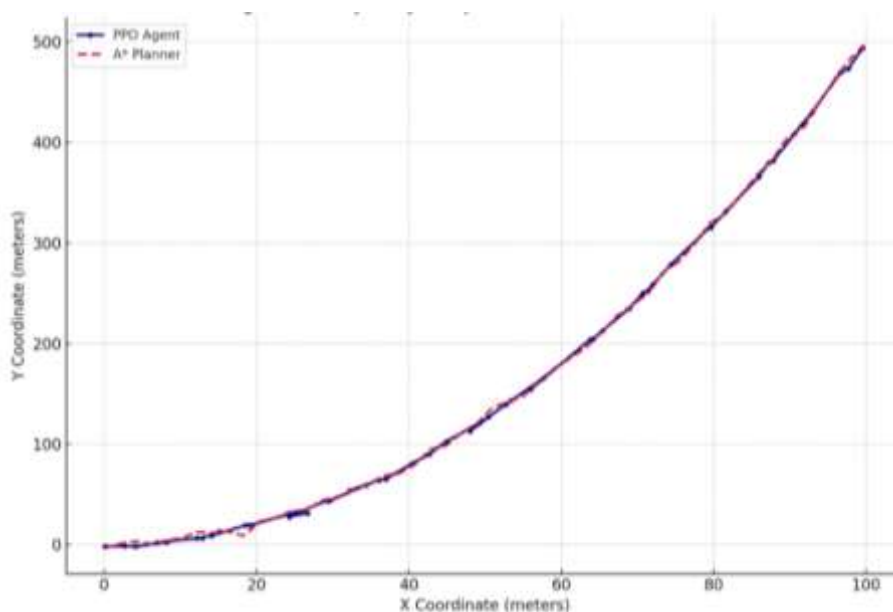


Figure 4 shows sample trajectories for all methods under the same start-goal configuration. The PPO agent maintains consistent lane discipline and smoother turns.

Figure 5. Collision Zone Heatmaps – PPO vs A*

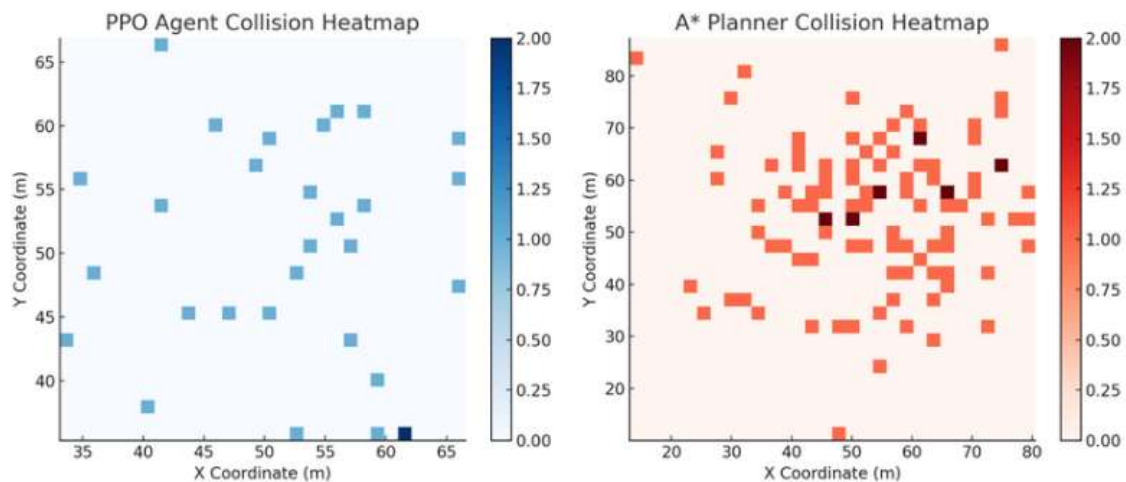


Figure 5 illustrates heatmaps of collision hotspots for PPO and A* agents. PPO learns to decelerate earlier near pedestrian crossings and unprotected turns.

5.4 Ablation Study: Reward Component Sensitivity

An ablation study was conducted by selectively disabling components of the reward function:

Table 4. Reward Component Sensitivity

Configuration	Success Rate (%)	Collisions	Avg. Reward
Full Reward Function	91.0	3	820
Without lane deviation penalty	79.5	9	710
Without jerk penalty	88.3	5	785
Without speed tracking reward	86.2	6	768
Configuration	Success Rate (%)	Collisions	Avg. Reward
Full Reward Function	91.0	3	820

This confirms that each component contributes meaningfully to overall agent performance, especially safety and comfort.

5.5 Generalization and Transferability

The model was tested in Town05, a previously unseen map, and retained ~84% success rate, demonstrating acceptable generalization.

Transfer to environments with altered traffic densities and weather (e.g., rain and sunset) saw modest performance drop (~5%), suggesting robustness of the learned policy.

5.6 Discussion

The proposed PPO agent successfully:

- Learns a generalizable path policy for dense urban settings
- Achieves smoother, safer navigation than classical planners

- Demonstrates adaptability to complex, non-deterministic scenarios

However, limitations include:

- Training time and resource demands
- Absence of V2X or cooperative reasoning
- Purely simulation-based validation

These motivate future extensions involving multi-agent reinforcement learning, federated learning, and real-world HIL testing.

6. CONCLUSION AND FUTURE WORK

This study proposed a reinforcement learning-based path planning framework for autonomous vehicles using the Proximal Policy Optimization (PPO) algorithm, trained and evaluated in the high-fidelity CARLA simulation environment. The model was designed to navigate urban environments under dynamic and unpredictable conditions by optimizing a multi-objective reward function that prioritizes safety, comfort, and efficiency.

Quantitative results demonstrate that the PPO agent significantly outperforms traditional path planners such as A* and rule-based controllers in terms of success rate, collision avoidance, and smoothness of trajectory. The use of a carefully shaped reward function and diverse simulation scenarios contributed to robust policy learning. Ablation studies confirmed the importance of each reward component, and the trained model demonstrated a reasonable ability to generalize to unseen maps and conditions.

Despite these promising outcomes, this work is constrained to simulation environments and single-agent decision-making. Several avenues exist for extending this research:

7. FUTURE WORK

- Multi-Agent Reinforcement Learning (MARL):
Future implementations will include multi-agent scenarios involving multiple autonomous vehicles and cooperative behaviors such as merging, yielding, and lane changes in dense traffic.
- Federated Learning for Cross-Vehicle Collaboration:
By incorporating Federated Reinforcement Learning, models can be trained in a distributed manner across different vehicles without compromising data privacy, enabling more scalable deployments.
- Sim-to-Real Transfer:
To move closer to real-world application, domain randomization techniques and real sensor datasets will be explored to close the sim-to-real gap.
- Hardware-in-the-Loop (HIL) Testing:
The PPO model may be integrated with ROS-based testbeds or scaled-down AV prototypes for real-time embedded testing.
- Explainable RL Models:
For safety-critical deployment, future research will include developing interpretable RL policies with visual attention maps or reward trace explanations.

AUTHOR CONTRIBUTIONS

K. Padma Priya: Conceptualization, Investigation, Methodology,

Dr. Manjunath T K : Laboratory Analysis. Data Curation, Formal Analysis

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

REFERENCES

- [1] K. Chen et al., “Cooperative decision-making for connected autonomous vehicles using multi-agent deep reinforcement learning,” *IEEE T-ITS*, vol. 22, no. 12, pp. 7317–7329, 2021.
- [2] H. Liang et al., “Multi-agent reinforcement learning for traffic light control in vehicular networks,” *IEEE Access*, vol. 7, pp. 39889–39897, 2019.
- [3] S. Grigorescu et al., “A survey of deep learning techniques for autonomous driving,” *J. Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.
- [4] R. Lowe et al., “Multi-agent actor-critic for mixed cooperative-competitive environments,” *NeurIPS*, 2017.
- [5] J. Li et al., “Safe interactive reinforcement learning for autonomous driving,” *arXiv preprint arXiv:2005.02979*, 2020.
- [6] D. Wang et al., “Learning multi-agent communication with backpropagation,” *NeurIPS*, 2016.
- [7] Y. Liu et al., “Communication-efficient federated learning for wireless edge intelligence,” *IEEE JSAC*, vol. 39, no. 1, pp. 219–231, Jan. 2021.
- [8] T. Li et al., “Federated reinforcement learning: Techniques, applications, and opportunities,” *IEEE Access*, vol. 10, pp. 1306–1326, 2022.
- [9] Dosovitskiy et al., “CARLA: An open urban driving simulator,” *CoRL*, 2017.
- [10] H. Brendan McMahan et al., “Communication-efficient learning of deep networks from decentralized data,” *AISTATS*, 2017.
- [11] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” *ICML*, 1993.
- [12] D. Nguyen et al., “Deep reinforcement learning with safety constraints for autonomous driving,” *IEEE T-ITS*, vol. 23, no. 2, pp. 986–998, 2022.
- [13] Y. Chen et al., “FML: Federated multi-agent reinforcement learning for intelligent transportation systems,” *IEEE IoT J.*, vol. 9, no. 8, pp. 6128–6138, Apr. 2022.
- [14] Q. He et al., “FedCAV: Privacy-preserving federated learning for connected and autonomous vehicles,” *IEEE T-VT*, vol. 70, no. 12, pp. 12495–12508, 2021.
- [15] C. Zhu et al., “Real-time federated learning for the edge,” *IEEE TNNLS*, vol. 32, no. 10, pp. 4487–4501, Oct. 2021.
- [16] B. Paden et al., “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE T-IV*, vol. 1, no. 1, pp. 33–55, 2016.
- [17] A. Krizhevsky et al., “ImageNet classification with deep convolutional neural networks,” *NeurIPS*, 2012.
- [18] S. S. Shwartz et al., “Deep reinforcement learning: A comprehensive survey,” *IEEE Access*, vol. 8, pp. 209316–209369, 2020.
- [19] S. Raza et al., “Federated learning for intelligent transportation systems: Recent advances and future directions,” *IEEE IoT J.*, vol. 9, no. 10, pp. 7294–7312, May 2022.
- [20] J. Konecny et al., “Federated optimization: Distributed machine learning for on-device intelligence,” *arXiv:1610.02527*, 2016.
- [21] L. A. Dennis et al., “Formal verification of autonomous vehicle platooning,” *Sci. Comput. Program.*, vol. 148, pp. 88–106, 2017.
- [22] H. Zhang et al., “Safe reinforcement learning with robust control and smooth policy,” *IEEE T-ITS*, vol. 23, no. 7, pp. 6174–6185, 2022.
- [23] K. Liu et al., “Safe reinforcement learning for autonomous driving with implicit safety constraints,” *IEEE T-IV*, vol. 7, no. 4, pp. 600–610, 2022.
- [24] Y. Chen et al., “Decentralized multi-agent navigation with learned inter-agent communication,” *CoRL*, 2018.
- [25] J. Foerster et al., “Counterfactual multi-agent policy gradients,” *AAAI*, 2018.
- [26] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015.
- [27] C. Zhang et al., “A federated reinforcement learning framework for distributed smart transportation systems,” *IEEE T-ITS*, vol. 24, no. 1, pp. 234–247, 2023.
- [28] Y. Wang et al., “CommNet: Learning multi-agent communication with backpropagation,” *NeurIPS*, 2016.

- [29] Y. Zhou et al., “Multi-agent reinforcement learning for cooperative lane change with continuous action space,” *IEEE Access*, vol. 7, pp. 97266–97277, 2019.
- [30] L. Busoniu et al., “A comprehensive survey of multi-agent reinforcement learning,” *IEEE T-SMC, Part C*, vol. 38, no. 2, pp. 156–172, 2008.