Journal of Research in Science and Engineering (JRSE)
ISSN: 1656-1996 Volume-8, Issue-3, March 2026
DOI: 10.53469/jrse.2026.08(03).04

Research on Autonomous Vehicle Motion Control Based on Deep Reinforcement Learning and Neural Networks

Gang Han[1], Ye Lin[1,2,*]

[1]College of Mechanical Engineering, Tianjin University of Science and Technology, Tianjin 300222, China
[1,2]Automotive Big Data and Intelligent Technology Laboratory, Tianjin University of Science and Technology, Tianjin 300222, China
**Correspondence Author, 422498097@qq.com*

Abstract: ***Traditional motion control for autonomous vehicles (AVs) predominantly relies on precise vehicle dynamics models. However, model mismatch often compromises control accuracy in highly dynamic and complex driving scenarios. This paper proposes an integrated control framework that synergizes data-driven methods with advanced motion planning. Specifically, for longitudinal control, the Deep Deterministic Policy Gradient (DDPG) algorithm is implemented to achieve adaptive and robust speed tracking. For lateral control, a Deep Neural Network (DNN) is trained using expert simulation data generated by a Stanley controller to handle non-linear path-following tasks. Furthermore, an enhanced Hybrid A* algorithm is introduced to perform both global and local path planning, ensuring kinematic feasibility. A comprehensive closed-loop control architecture is systematically established, bridging high-level decision-making with low-level dynamics constraints. Simulation results demonstrate the effectiveness of the proposed scheme in improving tracking precision and environmental adaptability.***

Keywords: Autonomous Driving; Deep Reinforcement Learning; Motion Control; Path Planning; Data-driven.

1. Introduction

With the rapid advancements in artificial intelligence, multi-sensor fusion, and high-performance computing, Autonomous Vehicles (AVs) have transitioned from theoretical laboratory research toward practical real-world deployment. Within the logical hierarchical architecture of AVs, the motion planning and control system serves as the core pillar ensuring safety, efficiency, and passenger comfort in complex dynamic environments [1-2]. Recently, the integration of deep learning technologies has provided unprecedented environmental adaptability for perception and decision-making modules, demonstrating significant potential in handling high-dimensional inputs and real-time control feedback [3-4]. Traditional motion control methods, such as Model Predictive Control (MPC) and the Stanley controller, primarily rely on precise physical dynamics models. These methods exhibit strong determinism and interpretability in constrained linear scenarios. However, they face significant challenges regarding model mismatch and insufficient robustness when encountering nonlinear dynamics at high speeds, sudden variations in road friction coefficients, or sensor signal latencies [5-6]. Furthermore, research indicates that optimization algorithms like MPC impose a heavy computational burden when dealing with high-dimensional constraints, making it difficult to consistently meet millisecond-level real-time response requirements [7]. To overcome the limitations of model-driven approaches, data-driven control strategies have become a focal point for both academia and industry. Deep Reinforcement Learning (DRL), particularly the Deep Deterministic Policy Gradient (DDPG) algorithm, enables a direct mapping from environmental observations to continuous action spaces through end-to-end learning from large-scale driving trajectories. DRL has demonstrated exceptional generalization capabilities in longitudinal speed control and collision avoidance tasks [8-9]. Regarding lateral steering control, modern Deep Neural Network (DNN) architectures can effectively compensate for the tracking residuals of traditional controllers on complex curved paths by learning from massive expert driving data [10-11]. Nevertheless, the ultimate effectiveness of motion control is highly dependent on the quality of the upstream path planning. For non-holonomic vehicles, path planning must strictly adhere to kinematic constraints, such as the minimum turning radius [12]. Traditional A* algorithms often generate piecewise-linear paths that are not directly executable by low-level controllers due to the neglect of heading angles and curvature continuity. Consequently, improved algorithms based on Hybrid A* incorporate Dubins curves and dynamic penalty functions to generate smooth trajectories that balance global optimality with local drivability [13-14]. This paper aims to propose a comprehensive data-driven vehicle motion control framework by integrating improved planning algorithms with deep learning techniques to enhance control precision and stability in complex scenarios.

2. Methodology

2.1 Data-Driven Motion Planning Method

Figure 1 illustrates the overall architecture of the data-driven motion planning approach.

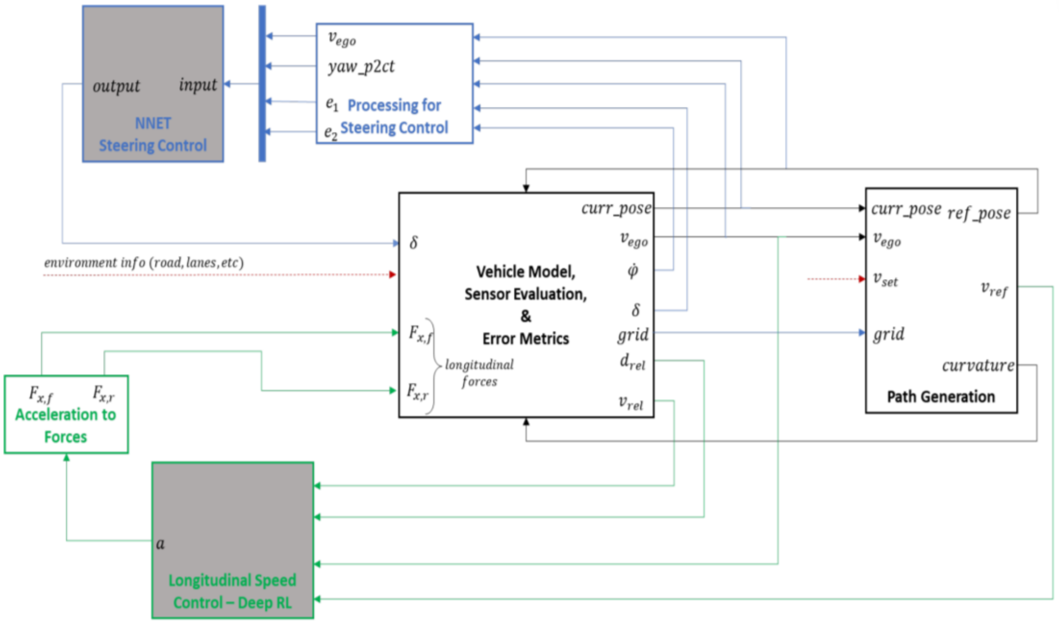

Figure 1: Overall Structure-Data-driven Approach

In this framework, the vehicle model receives longitudinal forces as inputs, which are derived by applying acceleration commands—generated by the Deep Reinforcement Learning (DRL) module—to the underlying dynamics model. Specifically, the DRL module takes environmental feedback and state information as inputs, including relative distance, relative velocity, reference speed, and path curvature. Based on these, the RL module calculates optimal acceleration or deceleration commands in real-time.

Another input to the vehicle model is the steering angle command output by the Neural Network (NNET) steering control module. The inputs for the NNET module are provided by a data processing module, which is responsible for the real-time calculation of lateral deviation, heading error, the difference between the path yaw angle and the lateral deviation direction, and the current ego-vehicle speed. Among these, the directional parameters are critical for determining the steering direction of the vehicle, the principle of which is shown in Figure 2.

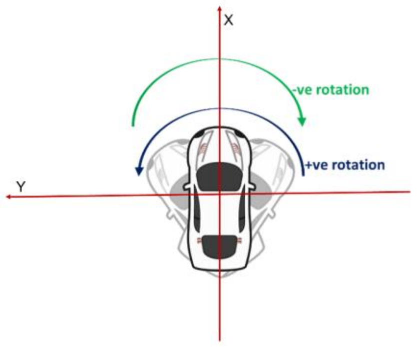

Figure 2: Direction of Rotation

The vehicle model unit also evaluates sensor measurements to calculate the relative velocity and distance between the ego vehicle and the lead vehicle (if present) in real-time. Figure 3 elaborates on the operational mechanism of the vehicle model unit.

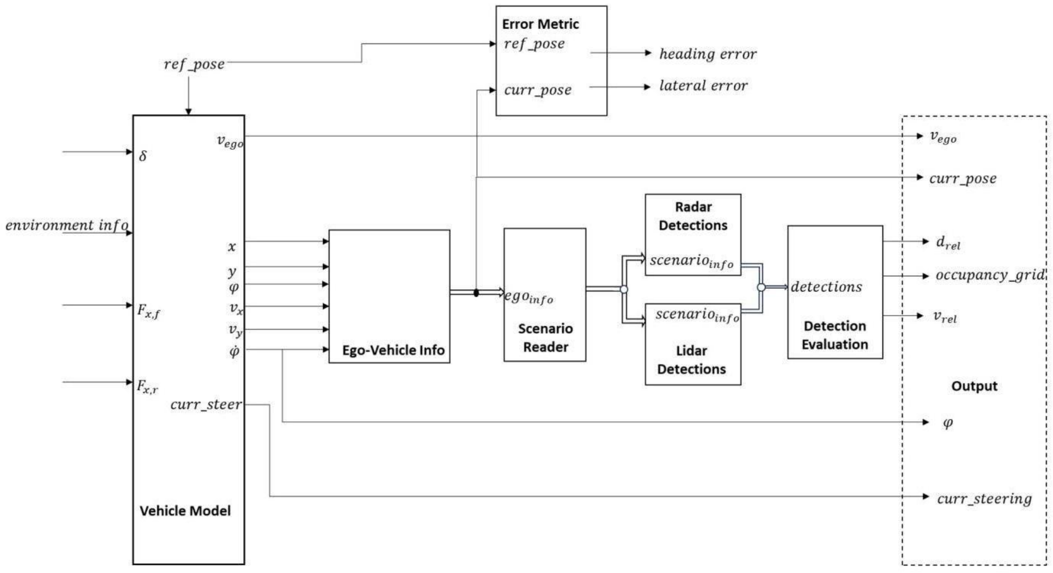

Figure 3: Vehicle Model Block

As shown in Figure 3, Within this module, the ego-information submodule encapsulates ego-centric state variables into structured bus signals for the scenario reading module. Furthermore, this submodule handles coordinate transformation, mapping the vehicle's coordinates from a local frame to a global coordinate system. The scenario reading module processes the predefined scenarios, roads, lanes, and other participants (non-ego vehicles), performing scenario dynamics simulations for each corresponding time step based on the ego-vehicle state. Subsequently, the current scenario state is transmitted to the radar detection module. This module uses information acquired from onboard sensors to generate object detection data, further calculating the relative distance and velocity to the lead vehicle. Additionally, initial vehicle states, such as initial position and yaw angle, are defined within these modules, typically using the starting point of the reference trajectory as the initial position.

The error measurement module retrieves the reference pose and the current pose from the vehicle model input and the ego-information submodule, respectively. By comparing their geometric relationship, the module calculates the heading error and lateral deviation. It should be noted that these error parameters are primarily used for offline validation and performance evaluation of the control effects. The final output parameters of the vehicle model module include the current pose, current speed, relative distance, relative velocity, current yaw rate, and current steering angle

The path generation module employs the Hybrid A* algorithm for both global and short-term (local) path planning. Figure 4 illustrates the mechanism of this module.

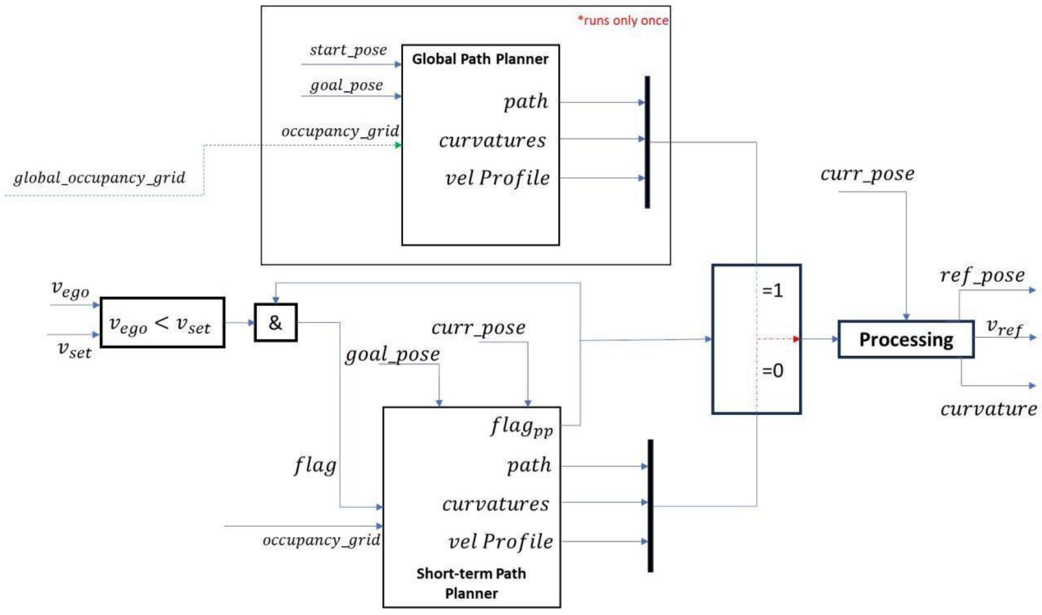

Figure 4: Path Generation Block

The global path planning module utilizes the Hybrid A* algorithm to plan the shortest path from the initial pose to the target pose and stores it in memory. In this global map, only road boundaries are treated as obstacles, allowing the planner to calculate complete path geometry, path curvature, and the velocity profile. The short-term path planning module is activated only when the ego-vehicle speed is lower than the set speed ($v_{ego} < v_{set}$), indicating the presence of a vehicle ahead. Its inputs consist of road boundary information and occupancy grid maps generated in real-time by onboard sensors. To ensure driving safety, the intermediate target pose for short-term planning is determined by the "three-second rule." Specifically, the target pose is set to the coordinates where the non-ego vehicle (lead vehicle) is expected to be three seconds after it reaches that location. This requires the system to dynamically extrapolate the intermediate target point based on the motion state of the lead vehicle. This target pose also belongs to the path generated by the global planner and is adjusted dynamically in real-time following the temporal updates of the occupancy map at each time step.

The path-following logic of the vehicle is managed by control flags. In the default state, the vehicle follows the global path, and the flag flag_{pp} is set to 0. When $v_{ego} < v_{\text{set}}$ and a specific trigger flag $flag\&flag_{pp}$ is 1, the vehicle switches to the short-term path execution mode. The flag $flag_{pp}$ is initially set to 1; if the vehicle has not yet reached the short-term target pose, $flag_{pp}$ is set to 0, and it reverts to 1 only upon reaching the target. When $flag_{pp}$ 为 1 and $v_{ego} \geq v_{set}$, the vehicle resumes following the global path. Local path planning continues to run until no obstacles remain in front of the vehicle (i.e., completion of the overtaking or obstacle avoidance maneuver).

Finally, the data processing module compares the current vehicle pose with the pose generated by the path planner to calculate an optimal target pose within the grid area ahead of the current position, which is then output as the reference pose for the underlying control logic.

2.2 Data-Driven Motion Planning Method

This section focuses on the development of an independent Reinforcement Learning (RL) based model, designed for verification and testing within a real-time simulation environment. The model is deployed in a straight-road scenario featuring dynamic non-ego vehicles (target vehicles).

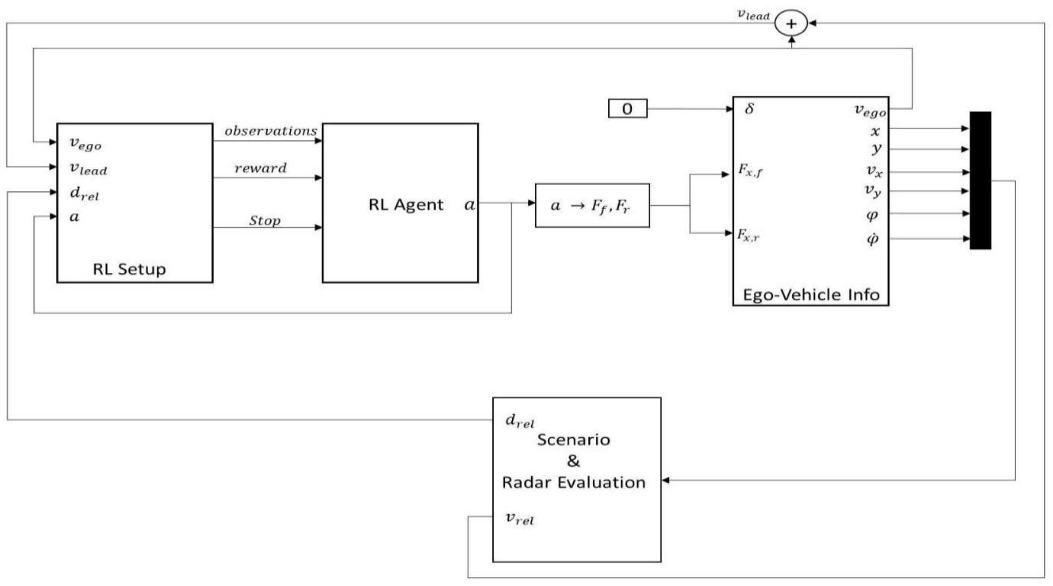

Figure 5: Reinforcement Learning Block – Model

Figure 5 illustrates the operational principle of the RL agent within the real-time simulation. The RL Setup Module is responsible for calculating the critical variables and constraints required by the agent. The observation space of this agent consists of the velocity error v_{err}, the integral of the velocity error, and the real-time velocity of the ego vehicle v_{ego}. Specifically, the velocity error is defined as the difference between the target velocity v_{set} and the current ego velocity, where the target velocity is determined as the minimum of the lead vehicle's speed and the reference speed v_{ref}.

The RL Setup Module also calculates the reward value based on the velocity deviation v_{err}, acceleration a, and distance error d_{err}. The distance error d_{err} is defined as the difference between the safety distance d_{safe} and the actual relative distance d_{rel}. The safety distance is computed using the current ego velocity, a fixed response time interval t_{gap}, and a default spacing d_{default}. Here, the time interval t_{gap} represents the duration required for the ego vehicle to react to the dynamic changes of the lead vehicle, while the default spacing d_{default} indicates the minimum longitudinal gap maintained when the vehicles are stationary. Equation (1) provides the specific calculation method for these parameters.

$$\begin{aligned} d_{\text{safe}} &= v_{\text{ego}} t_{\text{gap}} + d_{\text{default}} \\ d_{err} &= d_{\text{safe}} - d_{\text{rel}} \end{aligned} \tag{1}$$

Equations (2) and (3) demonstrate the calculation logic for v_{err}.

$$v_{\text{i}} = \begin{cases} v_{\text{lead}} & \text{if } d_{\text{err}} > 0 \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

To achieve a smooth transition between car-following and cruise control modes, an auxiliary velocity term v_{i} is introduced, as shown in Equation (2). When the actual distance is less than the safety distance ($d_{\text{err}} > 0$), the system adopts the lead vehicle's speed v_{lead} as a constraint. Conversely, if the distance remains within a safe range, this term is set to infinity to ensure it does not interfere with standard reference speed tracking.

$$\begin{aligned} v_{set} &= \min\left(v_{ref}, v_{\text{i}}\right) \\ v_{\text{err}} &= v_{set} - v_{ego} \end{aligned} \tag{3}$$

As described in Equation (3), the set velocity v_{set} is the minimum of the reference speed v_{ref} and the auxiliary speed v_{i}. The control deviation input required by the RL agent is thus the difference between this set velocity and the actual ego velocity.

Once all state variables are initialized, the reward function $r(t)$ is calculated according to Equation (4).

$$r(t) = -10 v_{\text{err}}(t)^2 - 0.2 a(t)^2 + C(t) + 0.1 D(t) \tag{4}$$

In this equation, the first term constrains the ego vehicle to adhere to the set velocity, ensuring safe operation. The second term penalizes abrupt changes in acceleration to prevent passenger discomfort. Since the primary objective of the controller is to maintain the set velocity, higher weight coefficients are assigned to v_{err}. The components C and D represent positive rewards derived from the velocity and distance parameters, as detailed in Equations (5) and (6).

$$D = \begin{cases} 1 & \text{if } d_{\text{rel}} \text{ is finite \& } d_{\text{err}} > 0 \\ 0.1 & \text{if } d_{\text{rel}} \text{ is not finite} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$C = \begin{cases} 1 & \text{if } v_{err} < 0.25 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Furthermore, the termination criteria for the simulation depend on the relative distance and the ego velocity. If either d_{rel} or v_{ego} becomes negative—signaling a collision or an abnormal reversing maneuver—the simulation terminates immediately.

Through these computations, the RL agent outputs continuous acceleration commands. These commands are subsequently converted into longitudinal forces and fed into the vehicle state-space model. The vehicle model then transmits the ego information to the scenario and radar evaluation modules, which compute new values for d_{rel} and v_{rel} as feedback for the RL Setup Module, thereby completing the closed-loop control system.

2.3 Neural Network-Based Steering Control

This section details the development process of an independent neural network (NN) model and investigates its validation and testing methodologies within a real-time environment. The primary objective is to enable the vehicle to maintain stability while traversing curved paths. In this control logic configuration, the ego vehicle receives only a constant velocity command, and it is assumed that no dynamic obstacles exist along the planned path. These constraints are intentionally imposed to isolate and rigorously validate the performance of the trained network specifically in path-following tasks.

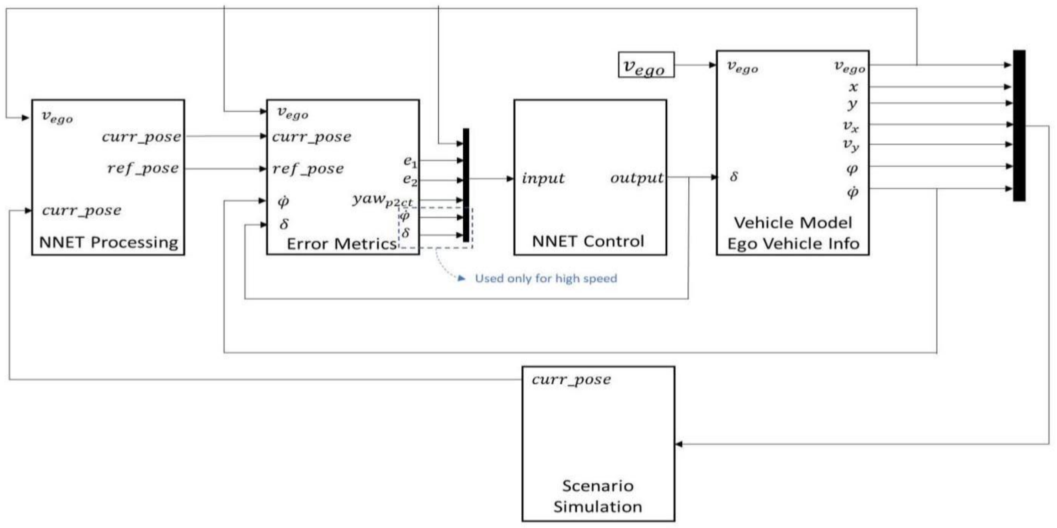

Figure 6: Neural Network Block - Model

Figure 6 illustrates the interaction mechanism between the neural network-based steering control model and the real-time simulation environment. Within this framework, the NNET processing module calculates the reference pose by retrieving the waypoint nearest to the center of the vehicle's front axle. Since the simulation provides only the coordinates of the rear axle, the front axle position must be derived geometrically using the vehicle's yaw angle φ and wheelbase w, a critical prerequisite for achieving precise steering control.

Considering the inherent one-step time delay of feedback signals in the simulation environment, the vehicle's speed significantly affects its actual position at the current time step. To address this, the pre-processing module performs extrapolatory compensation for the real-time pose of the front axle by integrating the ego velocity and the sampling interval. Under the assumption that the steering angle remains constant during this interval, the physical model for this motion compensation is illustrated in Figure 8.

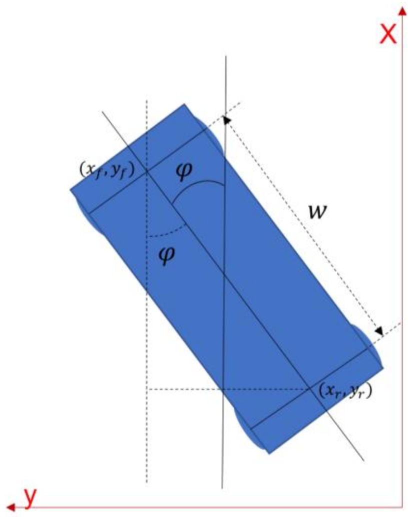

Figure 7: Rear to Front Coordinates

Figure 7 depicts the vehicle's position within the coordinate system and defines the geometric relationship between the front and rear wheels. The transformation formula for deriving the front axle coordinates from the rear axle coordinates is given in Equation (7).

$$\begin{aligned} x_f &= x_r + w\cos(\varphi) \\ y_f &= y_r + w\sin(\varphi) \end{aligned} \tag{7}$$

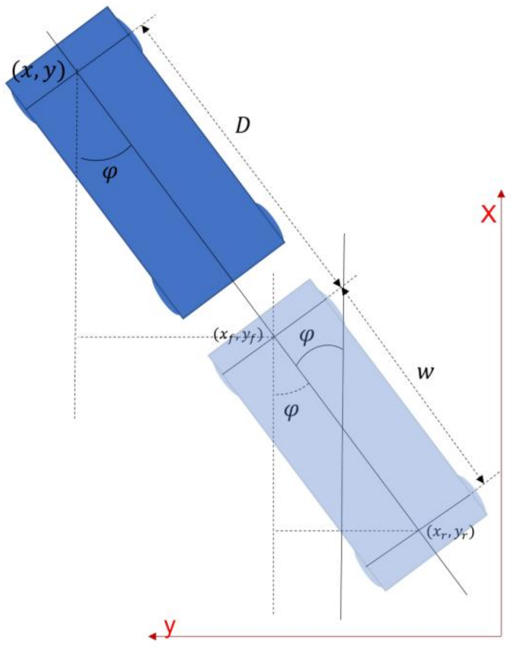

Figure 8: Rear to Front Coordinates with effect from vego

Equation (8) provides the formula for calculating the current position of the ego vehicle. Once the current position is determined, the reference pose is identified by searching for the nearest waypoint to that point.

$$\begin{gathered} D = v_{ego} Ts \\ x = x_f + D\cos(\varphi) \\ y = y_f + D\sin(\varphi) \\ \mathrm{curr}_{\mathrm{pose}} = \{x, y, \varphi\} \end{gathered} \tag{8}$$

Subsequently, the error measurement module calculates the lateral error e_1 by determining the Euclidean distance between the reference pose and the current pose, while the heading error e_2 is derived from their angular difference. To determine the specific orientation of the ego vehicle relative to the reference trajectory (i.e., whether it is to the left or right of the path), the term yaw_{p2ct} is introduced. This requires the prior calculation of the azimuth angle between the current and reference poses, as shown in Equation (9).

$$\begin{gathered} \mathrm{curr}_{\mathrm{pose}} = \{x, y, \varphi\} \\ ref_{\mathrm{pose}} = \{x_p, y_p, \varphi_p\} \\ CTE_\theta = \mathrm{atan2}(y_p - y, x_p - x) \end{gathered} \tag{9}$$

The parameter yaw_{p2ct} is defined as the difference between the reference path heading angle φ_p and the azimuth angle CTE_θ, normalized to a $[-\pi, \pi]$ range. If yaw_{p2ct} is positive, the vehicle is situated to the right of the reference trajectory; if negative, it is to the left.

$$\begin{gathered} yaw_{p2ct} = \varphi_p - CTE_\theta \\ yaw_{p2ct} = \begin{cases} yaw_{p2ct} - 2 * \pi & \text{if} \, yaw_{p2ct} > \pi \\ yaw_{p2ct} + 2 * \pi & \text{if} \, yaw_{p2ct} < -\pi \end{cases} \end{gathered} \tag{10}$$

Finally, e_1、e_2、yaw_{paw2ct} and v_{ego} are utilized as feature vectors for the neural network module. The NNET module computes the optimal steering angle, which is transmitted to the vehicle model along with the constant velocity command. The vehicle model then calculates real-time state information and relays it back to the scenario simulation module, thereby providing closed-loop pose feedback to the NNET processing module.

3. Conclusion

This paper presents a comprehensive motion control framework for autonomous vehicles, effectively synergizing data-driven learning with advanced heuristic planning. By systematically integrating longitudinal and lateral control modules with a kinematically feasible path planner, the proposed system addresses the limitations of traditional model-driven approaches in complex scenarios. The study yields the following key findings:

1) Longitudinal Control Efficiency: The implementation of the DDPG algorithm enables robust and adaptive speed tracking. By utilizing a multi-objective reward function that balances velocity error, distance constraints, and acceleration smoothness, the controller ensures both operational safety and passenger comfort.

2) Lateral Control Precision: The DNN-based steering control, trained on expert data and enhanced by extrapolatory motion compensation, successfully mitigates the impact of signal latencies in simulation. The integration of lateral and heading errors allows the vehicle to maintain high tracking precision even on complex curved paths.

3) Planning Feasibility: The enhanced Hybrid A* algorithm, incorporating the "three-second rule" for dynamic target selection, provides smooth and executable trajectories that strictly adhere to non-holonomic constraints. The dual-mode switching logic between global and local planning ensures stability during both cruising and obstacle avoidance maneuvers.

Simulation results confirm that this closed-loop architecture significantly improves environmental adaptability and tracking accuracy compared to traditional models. Future research will focus on the collaborative motion control of multi-vehicle systems and the optimization of the algorithm's real-time execution on low-power embedded platforms to bridge the gap between simulation and real-world deployment.

References

[1] Li, X., et al. Real-time trajectory planning for autonomous urban driving. IEEE/ASME Transactions on Mechatronics. 2016; 21(2):740-753.
[2] Chen, Y., Li, L. Advances in Intelligent Vehicles, London: Academic Press; 2013.
[3] Zhao, L., Li, L. A survey on recent advancements in autonomous driving using deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems. 2024; 25(1):112-135.
[4] Suraci, S.V., et al. Autonomous Systems: Deep learning for planning and control in autonomous vehicles, London: IntechOpen; 2025.
[5] Paden, B., et al. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on Intelligent Vehicles. 2016; 1(1):33-55.
[6] Falcone, P., et al. Predictive active steering control for autonomous vehicles. IEEE Transactions on Control Systems Technology. 2008; 16(3):534-543.
[7] Wang, S., et al. Reinforcement learning for autonomous driving: A survey. IEEE/CAA Journal of Automatica Sinica. 2019; 6(4):813-826.
[8] Yue, Y., et al. An integrated deep reinforcement learning-linear control strategy for longitudinal control of connected and automated vehicles. Transportation Research Part C: Emerging Technologies. 2024; 158:104-121.
[9] Lu, C., et al. Deep reinforcement learning for advanced longitudinal control and collision avoidance. Available from: https://arxiv.org/abs/2404.19087 [Accessed 10th March 2026]
[10] Alabd, A. Deep neural network-based linear quadratic programming for vehicle path tracking. Available from: https://ieeexplore.ieee.org/document/10211027 [Accessed 12th March 2026]
[11] Mu, M., et al. Model-structured neural networks to control the steering dynamics of autonomous race cars. Available from: https://arxiv.org/abs/2507.20427 [Accessed 14th March 2026]
[12] Dubins, L.E. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics. 1957; 79(3):497-516.
[13] Huang, S., et al. Improved hybrid A* algorithm based on lemming optimization for path planning of autonomous

vehicles. Applied Sciences-MDPI. 2025; 15(14): 7734-7750.

[14] Bonetti, A., et al. Improved path planning algorithms for non-holonomic autonomous vehicles: Roadmap hybrid A*. Available from: https://arxiv.org/abs/2304.14043 [Accessed 15th March 2026]