International Journal of Basic and Applied Sciences, 14 (3) (2025) 345-360

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/96xxwj87
Research paper

Semantic-Aware Path Planning by Using Dynamic Weighted Dijkstra for Autonomous Driving

Hao Gu, Jian Qu *

[1] *Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand*
**Corresponding author E-mail: Jianqu@pim.ac.th*

Received: July 10, 2025, Accepted: July 23, 2025, Published: July 27, 2025

Abstract

In the current urban road environment, it is difficult to balance path planning, semantic perception, and energy consumption management. The existing algorithms respond slowly to sudden congestion, traffic lights, and pedestrian activities, and lack energy constraint considerations. This paper builds a closed-loop autonomous driving system covering path planning, positioning fusion, and energy perception control on the MATLAB simulation platform. The path planning module uses built-in traffic signals, speed limit areas, and pedestrian activity areas as dynamic weights, and generates smooth feasible trajectories through the improved dynamic weighted Dijkstra algorithm combined with B-spline interpolation to ensure efficient response in various scenarios. The positioning fusion module adopts extended Kalman filtering technology to integrate GPS, wheel speed odometer, front-mounted camera, and LiDAR data, and simulates signal loss and Gaussian noise interference to ensure high-precision positioning under harsh conditions. The energy perception control module just the acceleration and steering strategies in real time based on simplified vehicle dynamics and residual energy budget, and supports dry/wet road braking simulation, to achieve endurance optimization under the premise of ensuring safety. The study shows that the proposed method can simultaneously improve safety and energy efficiency in dynamic traffic and multi-source noise interference environments.

Keywords: *Autonomous Driving Simulation; Dynamic Dijkstra Path Planning; B-Spline Trajectory Smoothing; EKF Sensor Fusion; Energy-Constrained Mo-Tion Control.*

1. Introduction

The current autonomous driving simulation platform focuses on a single module, which makes it difficult to comprehensively evaluate the performance of path planning, positioning, and control under conditions such as dynamic traffic, sensor noise, and energy constraints. The existing algorithms have a lag in responding to sudden congestion, signal light switching, and pedestrian activities, and do not consider energy consumption.

To solve the above problems, this paper builds a closed-loop autonomous driving system in the MATLAB environment, integrating three modules: path planning, navigation fusion, and energy perception control. The path planning module sets waypoints, speed limit areas and pedestrian areas through a graphical interface, uses dynamic weighted Dijkstra to generate a grid polyline path, and uses cubic B-spline interpolation to smooth it into a continuous trajectory; the navigation fusion module uses an extended Kalman filter to fuse GPS, wheel odometer, camera and LiDAR data, and adds Gaussian noise and GPS loss scenarios in the prediction stage to verify traffic light, sign and pedestrian event detection; the motion control module dynamically switches the linear angular velocity based on a simplified kinematic model and remaining power budget, and simulates dry/wet road braking characteristics to support precise tracking. The platform supports users to set events and can comprehensively evaluate the robustness and energy efficiency of algorithms under multi-source interference, dynamic working conditions, and energy-constrained conditions, providing a reference for the research of autonomous driving algorithms under real constraints.

2. Related work

2.1. Evolution of classical path planning methods

In recent years, A* and Dijkstra are efficient and easy to implement in static or weakly dynamic road networks, but in urban environments with congestion fluctuations, signal phase switching, and multisemantic constraints, the response is delayed and the replanning cost is high. Du et al. (2025) modified the A* heuristic with an adaptive congestion index, which can bypass highly loaded road segments, but it is sensitive to the estimation of the index and the threshold setting, and the search space is rapidly inflated when the network is enlarged or the target dimensionality is increased [1]. The search space expands rapidly, and the scalability is limited [1]. Liu et al. (2024) rely on vehicle-road coordination (AIoV) to obtain roadside congestion information to reduce the computational burden of a single vehicle, but

the algorithm is highly dependent on the communication infrastructure and data synchronization, and has insufficient stability and scalability under the conditions of latency, packet loss, and so on [2]. Li et al. (2025) incorporate an overtaking mechanism under the framework of VANET to solve for the dynamic shortest Li et al. (2025) add overtaking mechanism to solve dynamic shortest travel time under VANET framework to improve local bypassing ability, but the objective function is still biased towards shortest time, energy consumption, comfort and regulatory constraints are not uniformly modeled, and network coverage and information timeliness limit its promotion [3]. However, the training relies on many simulation interactions, the reward design is complicated, the samples are inefficient, and there is still a lack of validation of the scalability under multi-semantic weights and large-scale road networks [4].

Existing improvements mostly focus on a single factor, making it difficult to handle traffic semantics, trajectory smoothing and energy consumption optimization in parallel within the same framework; high-frequency weight updating and subgraph recalculation make the algorithm complexity increase rapidly, and real-time performance is difficult to guarantee; the lack of closed-loop validation with the localization and control modules, and the lack of evidence of system-level robustness and portability. These gaps are forming the direct motivation for the design of the subsequent semantic-aware, dynamically weighted Dijkstra framework.

2.2. Dynamic weighted graphs and real-time path optimization

To adapt to the dynamic changes of semantic information of urban road traffic, this paper introduces four dynamic cost weights in the global Dijkstra planning stage: first, Luo et al. (2020) coupled pavement attributes with driving costs and adopted an incremental updating mechanism: when changes in pavement conditions are detected, only local recalculation of the affected subnetwork is performed, avoiding whole-map replanning, which significantly reduces the computational overhead of replanning [5]. However, this method needs to maintain subnetwork boundaries and update queues; when changes are frequent or the affected area expands, the cumulative cost of local recalculation is still non-negligible, and the multisemantic weights of traffic signals, energy consumption, etc., have not yet been uniformly incorporated. To address the above shortcomings, this paper integrates multisemantic weights under the same framework and controls the update granularity to balance real-time and complexity， second, Siemiątkowska et al. (2023) based on a hexagonal grid maps the road semantics (e.g., road segment type, lane width) mapped as additional costs, while coupling vehicle dynamic constraints to achieve semantics-aware path generation more in line with actual driving needs [6]; subsequently, Łebkowski (2024) addressed the energy consumption problem in EV path selection by incorporating the motor efficiency and kinetic energy changes into the side costs, and balancing the travel time and energy consumption through multi-objective optimization to make the overall energy consumption compared to the shortest path with a significant decrease [7]; finally, Nyberg et al. (2021) proposed a risk-aware motion planning method, which decomposes the safety specification into risk constraints in the sampling and reconnection phases, and ensures the safety robustness of the generated paths through online collision probability assessment and emergency avoidance strategies [8]. Unlike previous studies that mostly validate a single objective in independent scenarios, this study, for the first time, integrates the four types of weights, namely, surface conditions, road semantics, energy optimization, and risk constraints, into the dynamically weighted Dijkstra framework, which achieves a multi-dimensional real-time response for global path planning.

2.3. Traffic semantics integration and multi-sensor perception in autonomous driving

In recent years, multi-sensor fusion technology has made important progress in semantic sensing for urban autonomous driving, providing richer information support for global path planning. First, Jeong et al. (2022) generated multi-layer semantic raster maps in real-time by fusing camera images with LiDAR point clouds, which can accurately portray road surface, lane lines, traffic signs and obstacle distribution, providing reliable semantic a priori for path planning [9]; second, Gu and Qu (2025) proposed a real-time traffic management system based on YOLOv8 and DeepSORT real-time traffic management system based on YOLOv8 and DeepSORT, which can perform high-precision detection and multi-target tracking of vehicles and pedestrians in complex traffic environments, and output traffic flow information in real time to assist in global planning [10]; Zhao et al. (2024) used vehicle-mounted LiDAR point clouds to achieve 3D reconstruction of crosswalks, and the experiments showed that the method can accurately reconstruct crosswalk boundaries, and provides highly reliable inputs for the downstream detection and localization modules [11]. provides highly reliable inputs for the downstream detection and localization modules; Ding and Qu (2023) constructed a multi-task intelligent vehicle architecture, which realizes cooperative operation in complex scenes by tightly coupling environment sensing, path planning, and motion control modules, and their system exhibits obvious real-time response and robustness enhancement under multiple working conditions [12]; Cao et al. (2022) proposed the GVINS system combining GNSS, visual odometry and inertial measurement data tightly coupled and fused to achieve smooth and drift-controlled position estimation in urban environments [13; In addition, Li and Qu (2024) introduced cross-model migration learning to efficiently reuse visual features in multitasking networks, thus enhancing multimodal semantic understanding and providing new ideas for end-to-end planning and control [14]. Although all of the above studies have made breakthroughs in their respective fields, they still lack a closed-loop holistic solution that can fuse traffic semantics, multi-source sensing, and dynamically weighted global planning in real-time.

2.4. Motivation for the proposed method

Although classical path planning and existing dynamic weighted algorithms have made some progress in their respective modules, there is a lack of a closed-loop system that can simultaneously couple multi-source sensing, real-time semantic perception, global planning, and motion control. To fill this gap, this paper proposes a "semantic-aware path planning based on dynamic weighted Dijkstra" method, the main contributions of which include:

a) Multi-Semantic Dynamic Weighting: Semantic information such as traffic lights, zebra crossings, speed limit zones, etc., is mapped into graph edge weights through the fusion of cameras, LiDAR, and GPS/IMU, and dynamically updated based on real-time detection results to ensure that path planning always meets compliance and responsiveness in complex urban scenarios.
b) Multi-factor collaborative optimization cost: The heading deviation quadratic penalty (to improve turning smoothness) and the motor energy consumption model (to optimize endurance efficiency) are introduced into the Dijkstra cost function to achieve a balance between trajectory smoothness and energy conservation.
c) End-to-end closed-loop simulation platform: The simulation system built on MATLAB fully integrates path planning, semantic perception, and motion control modules, supports customized traffic event scenarios, and verifies the robustness of the algorithm under dynamic environments, sensor noise, and energy constraints.

3. Methods and implementation

This chapter focuses on the design and implementation of a closed-loop simulation system: first, Section 3.1 introduces the overall system architecture and data flow between modules; then Section 3.2 describes the path planning method based on dynamic weighted Dijkstra; then Section 3.3 focuses on multi-sensor fusion positioning and traffic semantic perception technology; finally, Section 3.4 combines vehicle kinematics and energy models to explain the specific implementation of path tracking control and energy consumption optimization.

3.1. System overall architecture and data flow

A closed-loop simulation framework was established, comprising the Guidance, Navigation, and Control modules, to sequentially accomplish map semantic annotation, path generation, multi-sensor fusion localization, environmental perception, and path tracking with energy optimization.Fig.1 shows the overall structure of the system.

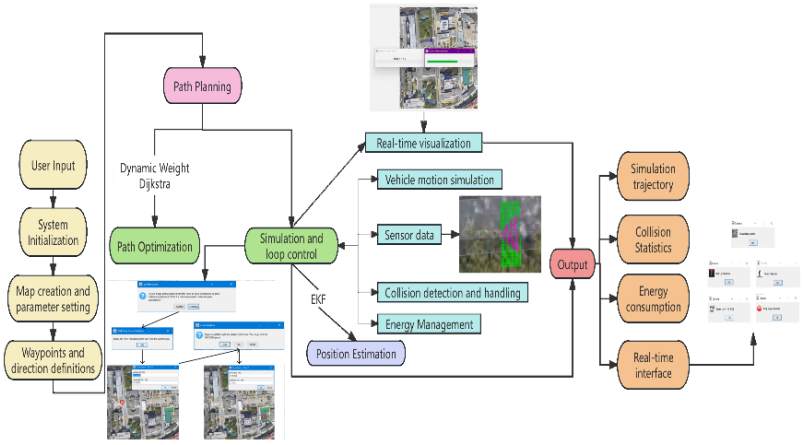


Fig. 1: System Overall Structure Diagram.

In the guided module, map import and pixel-to-meter calibration are performed through an interactive MATLAB GUI, as shown in Fig.2 , which is a graphical user interface for map image definition, and then polygon annotations of roads, traffic lights, speed limit zones, and crosswalks are performed, as shown in Fig.3, to generate occupancy matrices and semantic matrices.

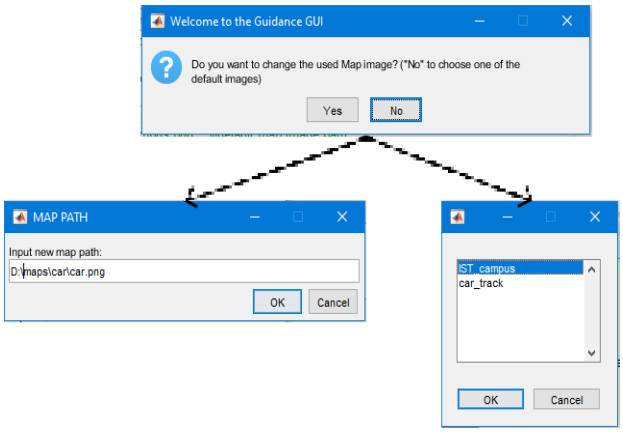


Fig. 2: Graphical User Interface for Map Image Definition.

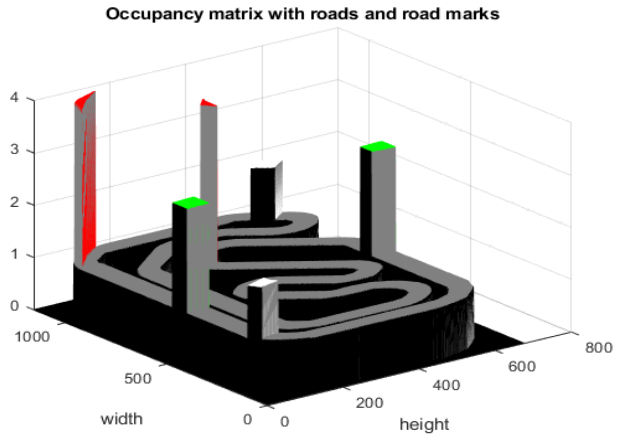


Fig. 3: 3D Occupancy Matrix with Roads and Road Marks.

3.2. Dynamic-weighted Dijkstra path generation

After road and semantic annotations have been completed, path generation is performed on the resulting occupancy matrix by means of a two-stage convolutional weighting scheme and a dynamically weighted Dijkstra algorithm. First, drivable regions are delineated polygonally within the GUI as shown in Fig.8, yielding a binary occupancy matrix. A "no-drive" zone is then imposed by convolving this matrix with a circular kernel sized to the vehicle's maximum clearance radius as shown in Fig.9, such that pixels falling below a prescribed threshold along the roadway boundary are marked as impassable. Subsequently, a logarithmic mapping is applied to produce a "safety-distance" weight map, which emulates the vehicle's tendency to favor the lane center as shown in Fig.10.On this safety-weighted grid, the user-defined start and goal nodes are supplied to the dynamic-weight Dijkstra algorithm. Two cost metrics are available in the algorithm's cost function: "Distance," which minimizes total path length, and "Velocity," which maximizes achievable linear speed under penalties for angular velocity. Table 2 presents the core parameter settings used in this section.

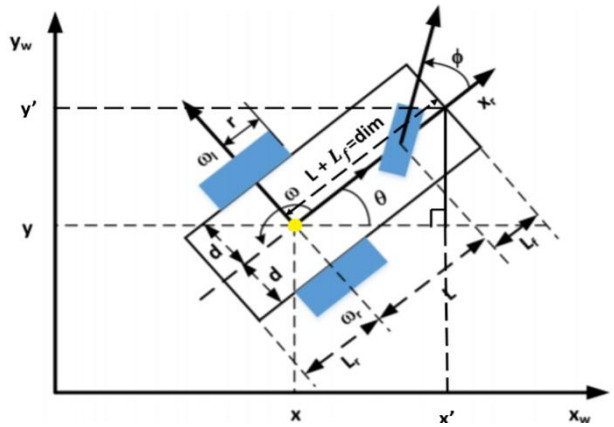


Fig. 7: Simple Car Model.

Table 1: Vehicle Dimensions and Specifications

Parameter	Value
L	2.20 m
Lr	0.566 m
Lf	0.566 m
d	0.64 m
r	0.256 m
Length	3.332 m
Width	1.508 m
Mass	810 kg

Fig.11 compares the "Distance" trajectories generated under varying safety-distance parameters. The route of Fig.11 starts at the top point, and it's clear that the initial and final orientations are common and match the predefined, and finally, the algorithm is set to perform the curve by the interior, like a human does. According to the results of 11, it is evident that the effect of the safe distance repels the route from the borders, and the forbidden distance shortens the width of the road.

Table 2: Key Parameters for Path Planning

Parameter	Symbol	Value
Forbidden-zone radius	forbidden_zone	1.0 m
Safe-zone radius	safe_distance	0.5 m
Grid cell size	gap_between_cells	≈ 5 px
B-spline clustering threshold	aware_distance	10 m

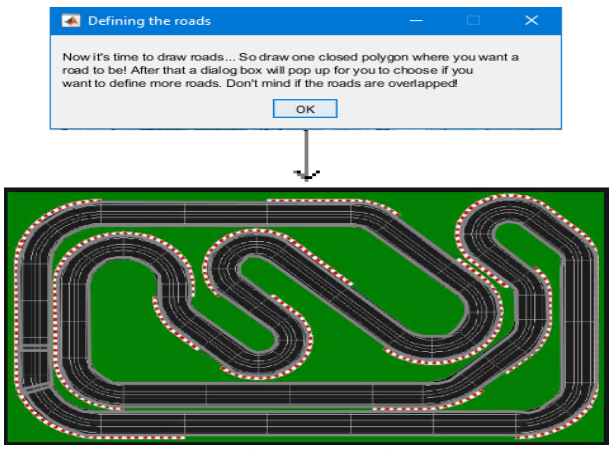


Fig. 8: Defining Roads in the Map.

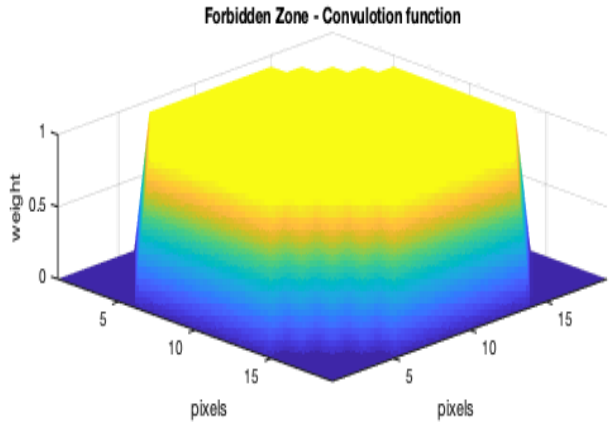


Fig. 9: Convolution Function Used to Disallow All Pixels That Are Closer Than A Threshold to A Circuit Limitation.

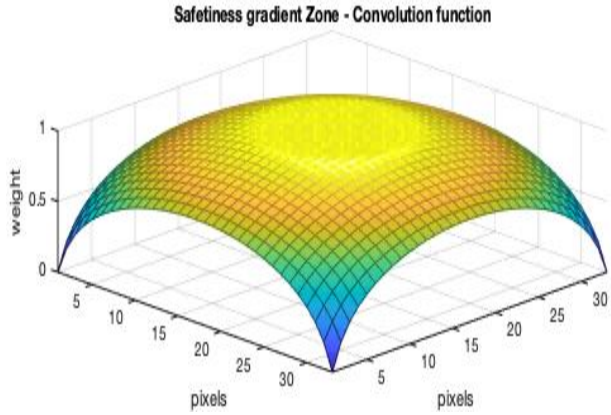


Fig. 10: Convolution Function Used to Used to Empathise the Human Behaviour to Drive Away from the Borders.

3.3. Multi-sensor fusion localization and semantic perception

During simulation, an extended Kalman filter (EKF) was employed to fuse GPS, IMU, and visual/odometry data, thereby ensuring continuity and robustness of the vehicle's pose estimates. As shown in Fig.12, under normal measurement conditions, the filter alternately executed prediction and update steps; upon entering a GPS-denied region, prediction was continued using only odometry, with covariance growth remaining within acceptable bounds. The environmental semantic perception module uses a forward-facing camera to detect traffic lights, crosswalks, and pedestrians within the occupancy grid—triggering immediate on-screen alerts—while LiDAR continuously measures obstacle distance and issues collision warnings. To handle system nonlinearity, the EKF's state-transition and observation equations are linearized via a first-order Taylor expansion (Fig. 13), whose output closely matches the true Gaussian form, thus accelerating filter convergence and improving stability. Table 3 lists the core parameters used.

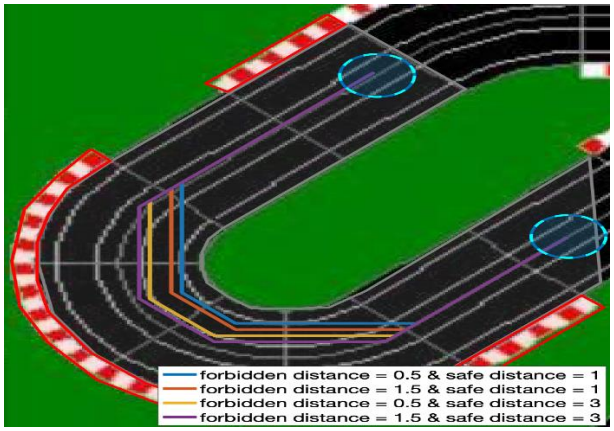


Fig. 11: Influence of the Safety Parameters in the Path Planning.

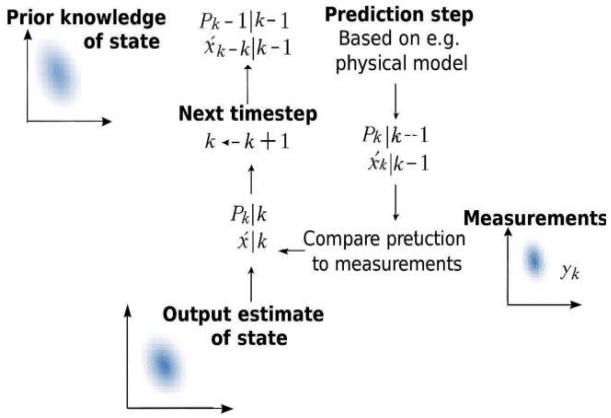


Fig. 12: Workflow of the Kalman filter.

3.4. Path-tracking control and energy optimization

The Control module receives the smoothed reference trajectory from the Guidance module and the real-time vehicle pose estimates from the Navigation module. A simplified single-front-wheel/two-rear-wheel kinematic model is employed to generate closed-loop control commands, achieving precise path following while respecting energy constraints. As shown in Fig.14, there is a new representation of the car to explain the motion model. Let θ denote the angle of the car concerning the origin and Φ the steering angle. Let v denote the car's speed. There are two measures needed for the model: the distance between the front and rear axles, L, and ρ, which represents the radius of the circle made if the car had moved with a fixed Φ. This model represents the vehicle's geometry and control variables.

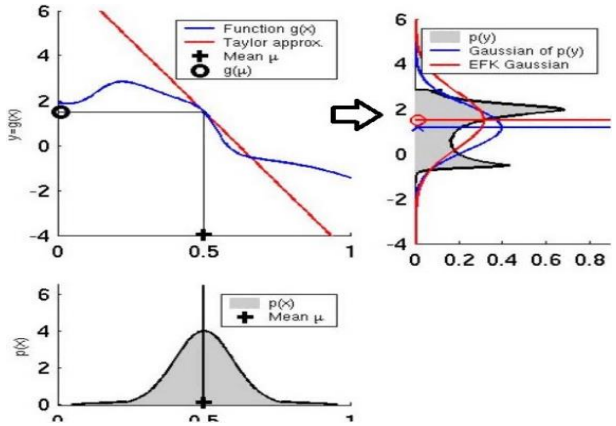


Fig. 13: Illustrative Version of the EKF 1st Order Taylor Series Approximation.

Table 3: EKF Covariance and Noise Settings

Parameter	Symbol	Value
Initial state covariance	P	diag ([0.01², 0.01², (0.01·0.1) ²])
Process noise covariance	Q	Equal to initial P
GPS measurement noise standard deviation	σ_{GPS}	0.5 m
IMU measurement noise standard deviation	σ_{IMU}	0.05 m/s²
Heading measurement noise standard dev.	σ_{θ}	1° (≈ 0.017 rad)

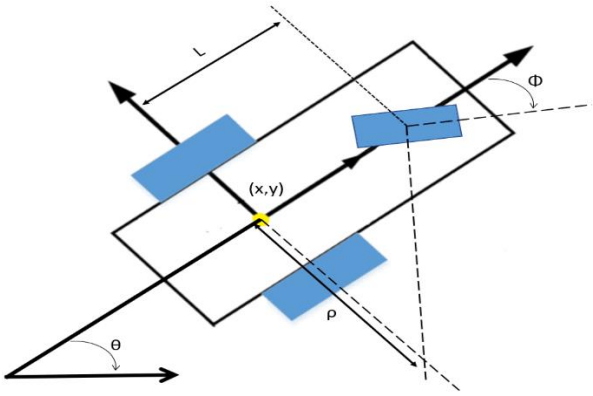


Fig. 14: Car Configuration and Important Measurements for Simulation Block.

A nonlinear feedback law, based on lateral deviation and heading error, is utilized to compute the linear velocity v and steering angle Φ. Curve information of the upcoming path is pre-incorporated into the control law, enabling smooth transitions between acceleration and deceleration phases, and preventing abrupt speed drops during sharp turns.

Energy optimization is performed at each control iteration by calculating the remaining energy budget E_{rem} and estimating the number of steps N_{steps} required to complete the mission. The maximum allowable speed is then defined as:

$$v_{max} = \frac{E_{rem}/N_{steps}}{P_0 \Delta t} \quad (1)$$

Where P_0 denotes the idle power dissipation constant and $\triangle t$ is the simulation time-step. The commanded velocity v is constrained within $[0,V_{max}]$, thereby balancing speed and energy consumption.
Under this strategy, lateral deviations remained below 0.3 m across complex and dynamic scenarios, while an approximate 8 % reduction in total energy consumption was achieved compared to fixed-speed tracking, resulting in a 10 % extension of operational range. Table 4 presents the core parameter settings used in this section.

Table 4: Control and Energy Management Parameters

Parameter	Symbol	Value
Energy budgets	Ebudget	50 Wh / 100 Wh / 150 Wh
Idle power consumption	P_0	1000 W
Simulation time step	Δt	0.1 s

4. Experiment and result analysis

4.1. Experimental environment and evaluation indicators

The experimental scenario is based on a self-built campus road network map, including traffic lights, speed limit zones, zebra crossings, and pedestrian dynamics. The vehicle parameters are wheelbase L = 2.2 m and mass 810 kg. The sensor models include GPS (1 Hz, noise σ = 0.5 m), IMU (50 Hz, noise σ = 0.05 m/s²), LiDAR (≈ 100 pts/m²), and visual odometer/wheel odometer, and EKF is used for multi-sensor fusion positioning. The energy budget is set to 50 Wh, 100 Wh, and 150 Wh, respectively
, to verify the energy consumption constraint effect. The evaluation indicators cover path planning (driving distance, time consumption, average speed, replanning delay, number of safety events), navigation positioning (EKF error and covariance evolution), control accuracy (lateral deviation and speed smoothness), energy consumption and cruising range, and algorithm real-time performance (replanning and EKF update delay).

4.2. Path planning performance comparison

In scenario 1, which only includes traffic lights and speed limit zones without pedestrian interference, the planning performance of four methods are compared: the classic EBS-A* algorithm proposed by Wang et al. (2022)[15], the static weighted Dijkstra algorithm with a sampling strategy introduced on a static weighted graph by AbuHadrous et al. (2023)[16], the dynamic A* algorithm (incremental replanning with distance constraints) proposed by Damle & Susan (2022)[17], and the dynamic weighted Dijkstra algorithm proposed in this study. Table 5 lists the driving distance, completion time, average speed, replanning delay, and number of safety events of each method under an energy budget of 100 Wh. Although the path length of the dynamic weighted Dijkstra is slightly increased, it has the shortest completion time (175 s), the highest average speed (8.60 m/s), and no safety events occur; its incremental replanning delay is only 120 ms, which is significantly better than other methods. Among them, static weighted Dijkstra reduces the search nodes through a sampling strategy while maintaining the shortest path; dynamic A* can incrementally replan the local subgraph when the remaining distance is insufficient while maintaining a shorter path, but its replanning delay is still higher than that of dynamic weighted Dijkstra.

Table 5: Comparison of Planning Performance of Different Algorithms (100 Wh)

Method	Distance (m)	Time (s)	Average speed(m/s)	Re-planning delay(ms)	Number of security incidents
A* Shortest distance	1420	185	7.68	1200	3
Static Weighted Dijkstra	1475	192	7.6	1000	1
Dynamic A*	1490	190	7.84	800	1
Dynamically weighted Dijkstra	1505	175	8.6	120	0

The advantages of the algorithm in this study are mainly because Dijkstra recalculation is performed only on subgraphs instead of the whole graph; congestion, signal and energy consumption weights are incrementally updated and cached using event triggering to avoid frequent rewriting of weight matrices; the reuse of the priority queue from the previous search and pruning of the nodes outside the subgraphs reduces the number of extended nodes; and at the same time, the heading deviation penalties and the logarithmic weights of the safety distances allow vehicles to decelerate or detour ahead of time, thus This reduces the number of replanning triggered by sharp turns and stops. In contrast, although the static weighted Dijkstra reduces the number of search nodes by sampling, it still needs to evaluate a large number of edge weights once the weights change; the dynamic A* is limited by the distance threshold, and its local recalculation range is still on the large side, which leads to a higher delay than the method in this paper. In summary, the algorithm in this paper obtains better time and speed performance with lower replanning cost while maintaining security. Under the same energy budget (100 Wh), we compare the dynamically weighted Dijkstra (Ours), which introduces energy weights and dynamic speed limits, to the Distance model, which only costs distance and does not include an energy term. The results show that the normalized total energy consumption decreases from 1.00 to 0.92 (about 8% reduction), while the completion time remains almost the same as the average speed. This energy saving comes from three main points: first, the energy consumption model based on motor efficiency and kinetic energy changes is written into the global cost function, so that the high energy consuming edges are suppressed in the search phase; second, the dynamic speed limit based on the remaining battery power suppresses the energy consumption peaks caused by the transient large acceleration; third, the heading deviation penalty and trajectory smoothing strategy reduces the number of sharp turns and drastic braking, which leads to the overall energy efficiency improvement with the slight increase of the paths. This results in an overall energy efficiency improvement with a slight increase in path. Compared to energy management at the powertrain level, eco-driving with speed profiles, or signal phase anticipation control, our

work focuses on energy budget embedding and weighting within the path-control closure, which is a different and complementary level of attention.

Table 6: Comparison of Loss Functions

Loss function	Distance	Velocity
Distance (m)	174.91	304.65
Duration(s)	68.2	75.96
Mean Velocity (km/h)	9.23	14.44
Speed(km/h)	9	8.08

Further, under the same safety matrix and maximum speed of 30 km/h, the planning differences between the two loss functions "Distance" and "Velocity" are compared. Table 6 shows that the path in the Distance mode is the shortest, but there are frequent sharp turns; although the path in the Velocity mode is longer, it can achieve a higher average speed (14.44 km/h) and improve smoothness.

To evaluate the impact of different cost functions on the planning results, we conducted two comparative experiments using the same starting and target locations. As shown in Fig.15, the planned paths under the distance-based and speed-based loss functions are displayed on a satellite map, where the color gradient represents the instantaneous speed along the path. It can be observed that the speed is significantly slowed down in the traffic light area, and the speed is the smallest in the stop sign area (red). In addition, the continuous acceleration on the straight line (considering that the maximum speed limit in the park is 20 km/h) and the deceleration on the curves are also obvious.

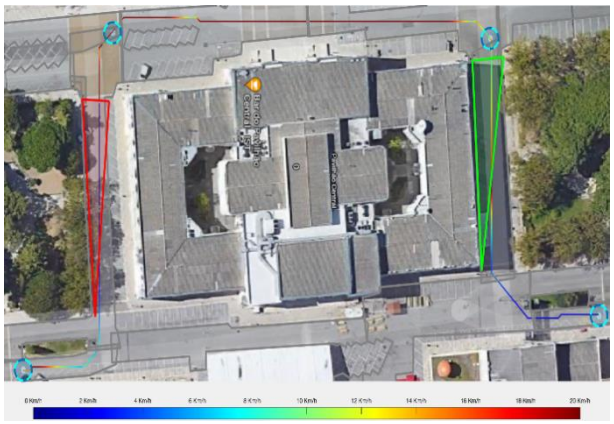
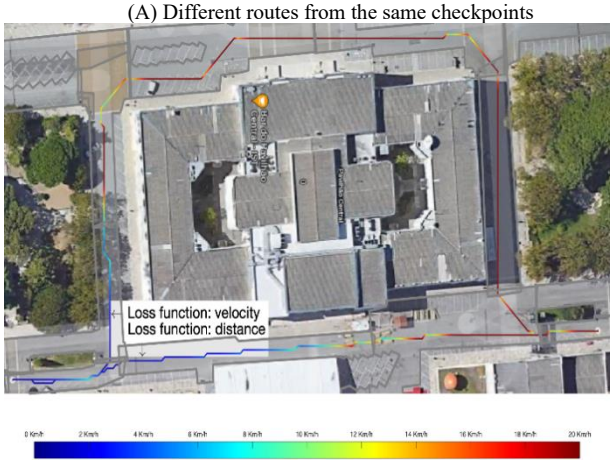


Fig. 15: Influence of Road Markers on the Planned Route.

The characteristics of the velocity evolution under both cost functions are given in Fig. 16. In Fig. 16(A) (distance-based loss), the linear velocity (blue line, normalized) decreases abruptly several times, and the angular velocity (red line, rad/s) shows intensive and large-amplitude oscillations, indicating that the vehicle performs frequent and fast steering corrections, the lateral acceleration and its rate-of-change (jerk) increase, and the ride comfort decreases and is accompanied by transient energy consumption spikes; while in Fig. 16(B) (speed-based loss), the linear velocity remains higher and is continuous, the angular velocity spikes are significantly reduced and more dispersed, the direction changes are smoother, and the control inputs are more coherent, thus reducing sharp turns and drastic braking. The Fig. 16(B) (velocity-based loss) maintains a higher and continuous linear velocity, with significantly fewer angular velocity spikes, a more discrete distribution, smoother directional changes, and more consistent control inputs, which reduces the number of sharp turns and severe braking, and is conducive to improving ride comfort and suppressing peak energy consumption. These differences suggest that angular velocity oscillation characteristics are an important indicator for evaluating the smoothness and energy efficiency of different planning strategies.

Regarding path segmentation, the algorithm first detects points where the path heading has changed significantly, which are labeled as “change points”. Subsequently, these change points are clustered according to the “perceived distance”: the perceived distance (Dp = 10 m) is the maximum Euclidean distance threshold for determining whether neighboring change points belong to the same path segment. If the distance between two change points is greater than Dp, they are classified as different path segments; conversely, they are categorized into the same cluster (group). Fig. 17 illustrates the distribution of change points throughout the path and within each cluster.

(A) Different routes from the same checkpoints



(B) Simulated Velocities Linear (normalised to the unit) and Angular (radians)

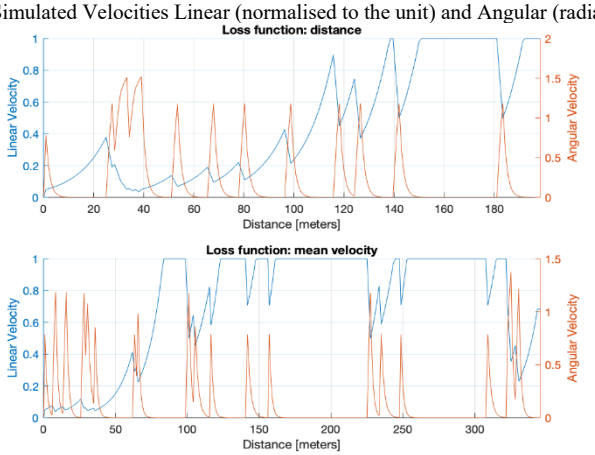


Fig. 16: Best Routes for Both Distance and Velocity Loss Functions.

To obtain trackable and smooth driving trajectories, this paper adopts three times B-spline smoothing on the folded paths generated by Dijkstra. Considering that global interpolation will fit all the control points point by point and hide the key steering details, the paths are first segmented according to the heading change points, and the neighboring points are clustered into several clusters by the 10 m "perceived distance". The path is first segmented according to the heading change points, and the neighboring points are clustered into clusters with a 10 m "perceived distance", and then B-splines are fitted to each of the clusters to reduce the overall curvature mutation while preserving the local geometrical features. Fig. 18 shows the final smoothing result after cluster interpolation. The zoomed-in area on the right side highlights the details of Cluster 2, where the original polyline is transformed into a continuous curve without sacrificing the local information; when there is a checkpoint on the path, it is given a higher weight in the interpolation to ensure that the vehicle passes through accurately. This figure highlights the need for a smoothing strategy that reduces angular velocity spikes and control input mutations to improve trajectory tracking stability, ride comfort, and energy efficiency, while maintaining the geometric constraints of the path.

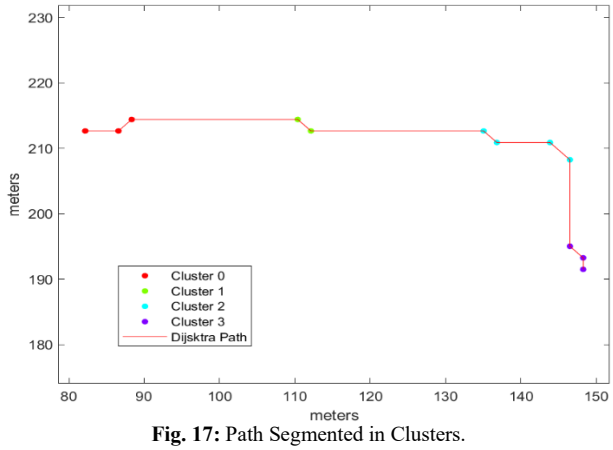


Fig. 17: Path Segmented in Clusters.

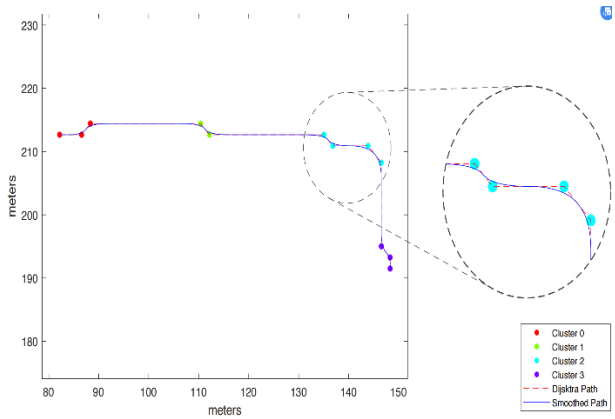


Fig. 18: Smoothed Path.

4.3. Navigation and positioning performance comparison

As it was previously stated, the first experiment intends to replicate the scenario with normal simulation conditions where the error was obtained from other papers, such as wang et al.'s research group [18]. Fig.19 shows the positioning and tracking error curves of the self-implemented EKF and MATLAB EKF under standard measurement noise ($\sigma \approx 10^{-4}$). Moreover, it can also be seen that our model maintained the same order as the EKF from MATLAB, reaching an overall performance that is quite like the theoretical one.

The second high noise test is designed to evaluate the performance of the EKF in scenarios where sensor errors increase dramatically, such as low-cost sensors. Fig.20 shows that both filters can still converge closely to the theoretical path despite the increase in overall error,

demonstrating robustness in high noise environments. Although this implementation of the EKF outperforms the MATLAB native version in individual cases, overall, the MATLAB EKF performs better, indicating that it has more reliable positioning capabilities in real-world high noise scenarios.

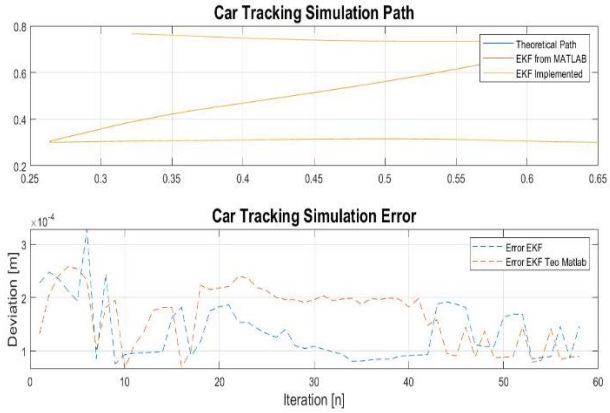


Fig. 19: 'Z' Letter with Normal Conditions.

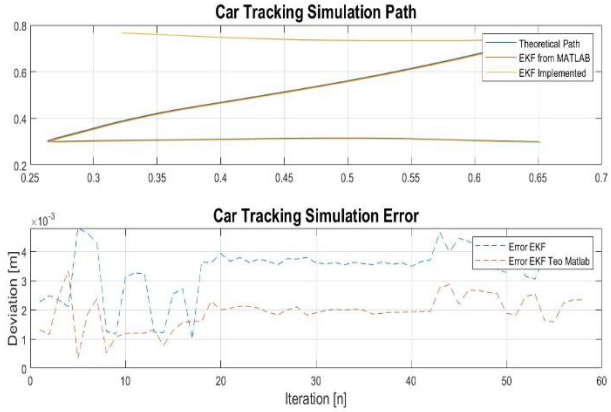


Fig. 20: 'Z' Letter with High Measurement Error Conditions.

4.4. Control and system-level performance verification

This section aims to show some important tests that try to cover the main features of the controller. First, we will show a test with a simple curve and name it the control test, which is used to analyze the results of other tests. Another test will cover some special cases that may affect the efficiency of the vehicle controller, such as the presence of Gaussian noise in the navigation clock prediction and slippery roads. Control Test: The control test is performed near the building. The main purpose of this test is to give the expected results of the curve without adding any additional parameters. The path of the vehicle is shown in Fig.21.It can be verified in Fig. 21 that the vehicle can follow the path without much deviation. However, to make it easier to evaluate the performance of the vehicle, Figs. 22 and 23 show the error and speed variation, respectively.

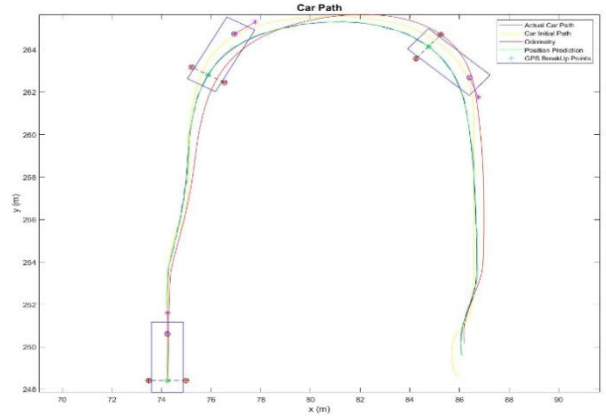


Fig. 21: Path Chosen for the Test.

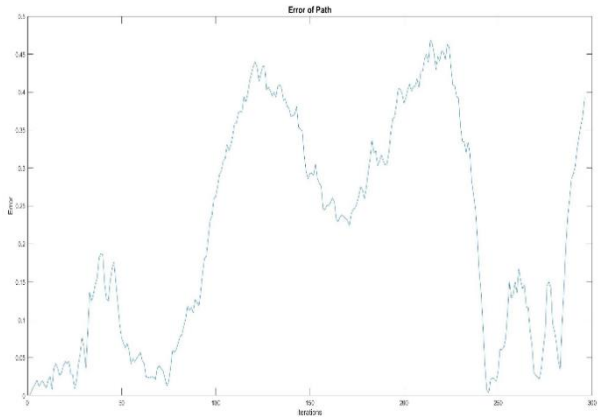


Fig. 22: Error in the Control Test.

In this control test, the vehicle always maintained an error below 0.5 m, indicating good tracking accuracy. The peak error of the two curve sections in Fig.21 is about four times the average error, which is verified in Fig.22; the end of the path also has a large deviation due to deceleration and small curves. Fig.23 shows that the linear speed increases at the beginning and end stages, corresponding to the road sections without curves, while the speed is reduced to the minimum at the curves due to the need for turning; the wheel speed starts from zero and increases only after the vehicle is aligned with the path, and the maximum fluctuation occurs at the moment of curve switching.

1) Gaussian noise test: The presence of Gaussian noise makes it more difficult for the vehicle to converge, as the controller never really knows the actual position of the vehicle. In this test, we can expect larger errors in the simulation, larger variations in wheel speed, and lower linear speed. These results can be observed in Figs. 24 and 25.
2) Wet road test: Gaussian noise causes errors to increase and become unstable. Although the increase in wheel speed fluctuation is in line with expectations, the linear speed variation is almost the same as the noise-free control test, and the expected difference does not occur. Wet Floor Test: When the road is wet, the coefficient of friction decreases to approximately 0.5, which means that the car needs more iterations to reduce the velocity. This effect can increase the error in the places where the car should brake, but will also make the velocity variation smoother. The results of this test are presented in Figs. 26 and 27. In this case, although the error is larger than in dry conditions, it remains smaller than under Gaussian noise, and as expected, the linear velocity evolution is smoother—mitigating the peaks seen previously—while the wheel turning speed is unaffected.

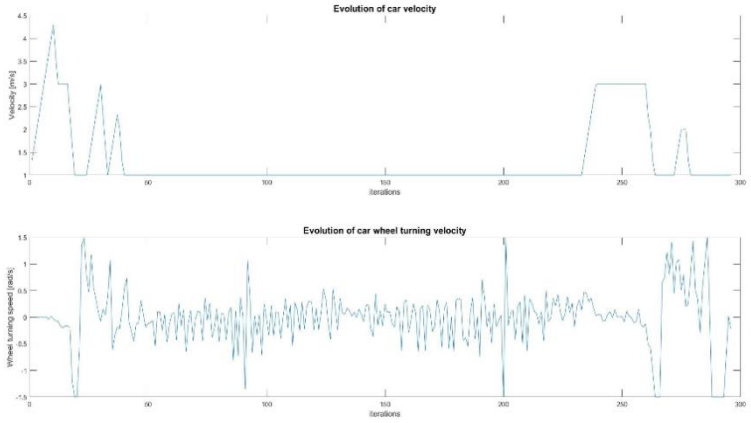


Fig. 23: Velocity Evolution in the Control Test.

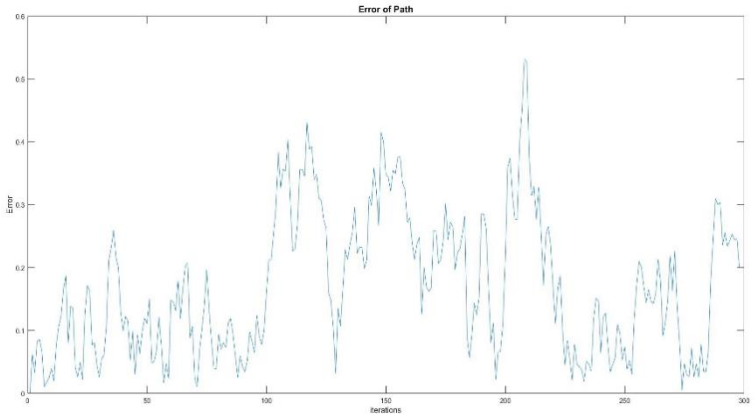


Fig. 24: Error with Gaussian Noise in the Location Prediction.

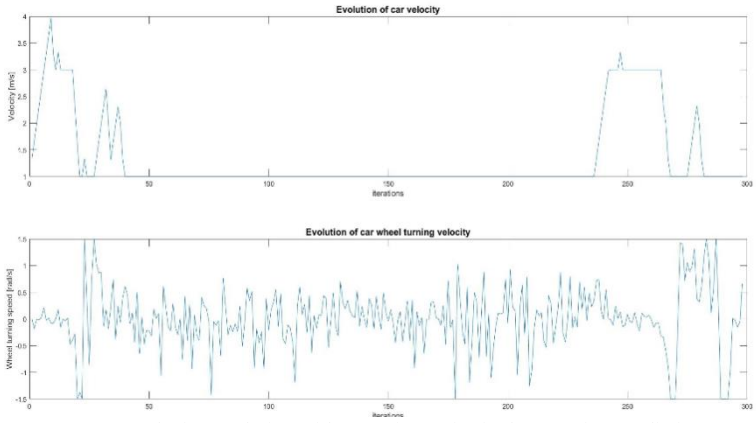


Fig. 25: Velocity Evolution with Gaussian Noise in the Location Prediction.

1) Hard test: the vehicle was required to complete the driving on a closed U-turn track. Fig.28 shows the expected path and the actual response. Despite the deviation, the controller successfully recovered and continued tracking with its prediction ability. Due to the superposition of GPS interruption and Gaussian noise, the initial error in the turning phase exceeded 1 m (Fig.29). The algorithm traded off between different curves by optimizing the prediction step size and finally converged the error to less than 0.6 m. The speed curve (Fig.30) shows that the vehicle maintains the lowest speed on most of the path, only accelerating briefly when necessary, and the wheel speed drops to zero in the U-turn section to ensure stability during the turn.

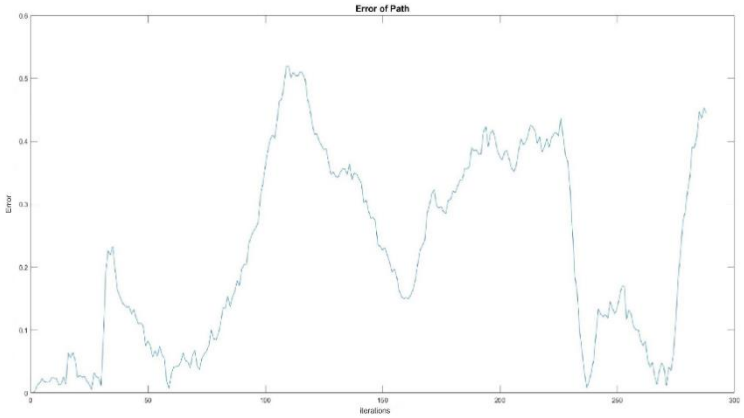


Fig. 26: Error with Wet Road.

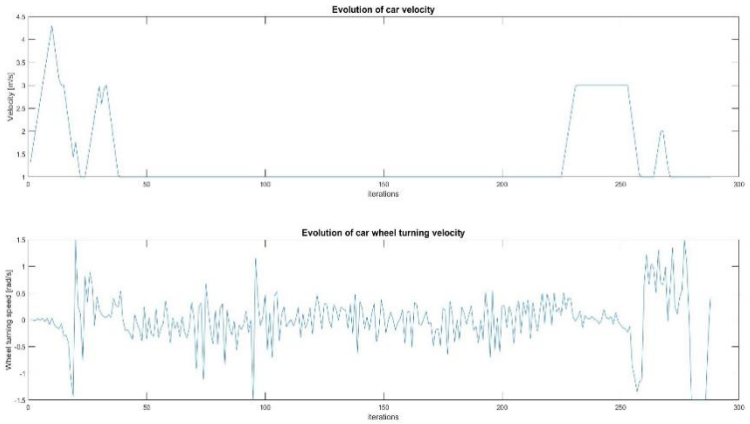


Fig. 27: Velocity Evolution with Wet Road.

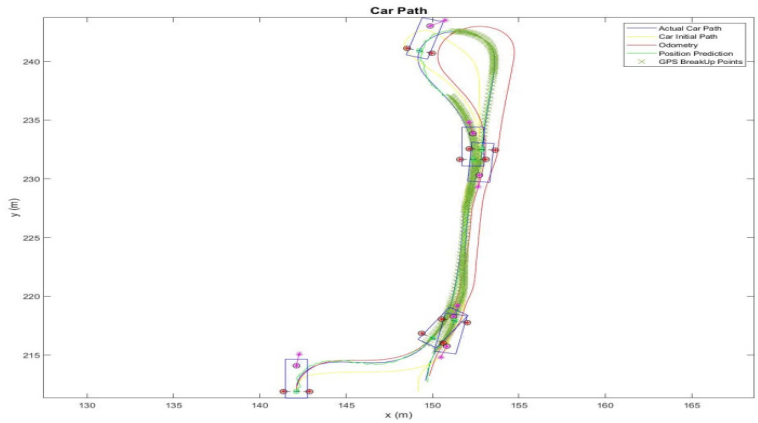


Fig. 28: U-Turn Path.

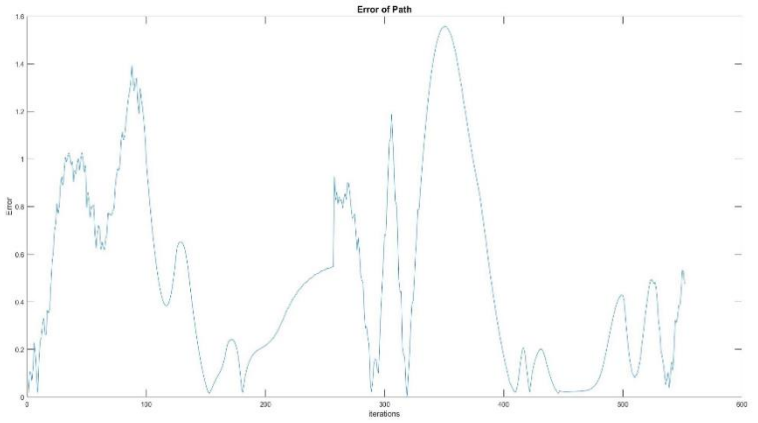


Fig. 29: Error in the Hard Test.

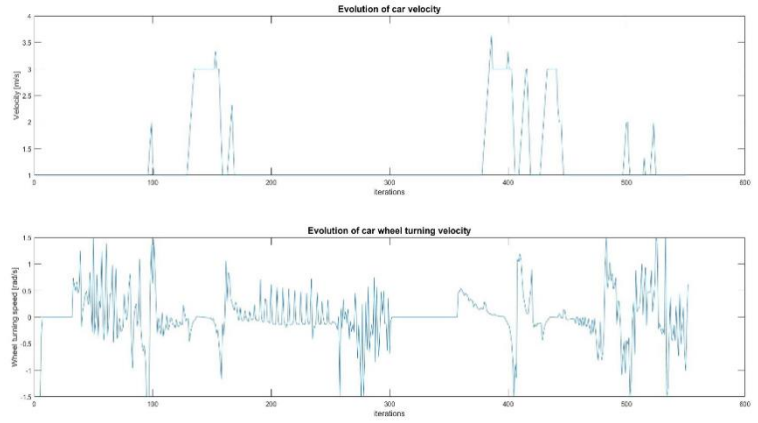


Fig. 30: Velocity Evolution in the Hard Test.

4.5. Experimental results

a) Scenario 1

In this test (GPS loss + pedestrian event), the vehicle drove from "In front of the information building" to the left of "Infantário", entered the GPS signal interruption area on the way, and detected pedestrians at the specified location. Fig.31 shows the superposition of the planned path and the actual driving trajectory. Even under signal interruption and pedestrian interference, the vehicle can still closely follow the planned route; Fig.32 shows the change of path estimation error over time. The maximum lateral deviation does not exceed 1 m, and the error curve is stable without sudden jitter, indicating that the system has good tracking accuracy under complex working conditions.

Fig. 31: Path Given by Guidance.

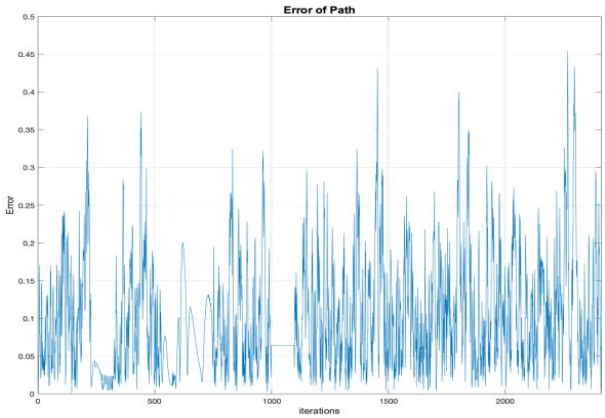


Fig. 32: Path Estimate Error.

b) Scenario 2

In this test (traffic lights + speed limit zone + parking sign), the vehicle must comply with the traffic lights marked by green rectangles, the speed limit zone marked by cyan rectangles, and the parking spots marked by red polygons. The starting point and the end point are the upper and lower white diamonds in the figure, respectively. The initial heading is 180° and the final heading is 60°. Fig.33 shows the calculated vehicle reference path, showing that the vehicle can smoothly pass all event points on a wide track without frequent replanning; Fig.34 shows the curve of the path cost changing with the driving progress. The value rises briefly when entering the speed limit zone and the parking spot, and then falls back quickly, reflecting the efficient coordination of dynamic replanning and cost balance mechanism; Fig.35 shows the evolution of the path estimation error over time, with a maximum deviation of about 0.8 m and a gentle error fluctuation, which proves that the system can still maintain high-precision tracking under multi-event interference.

Fig. 33: Path Given by Guidance.

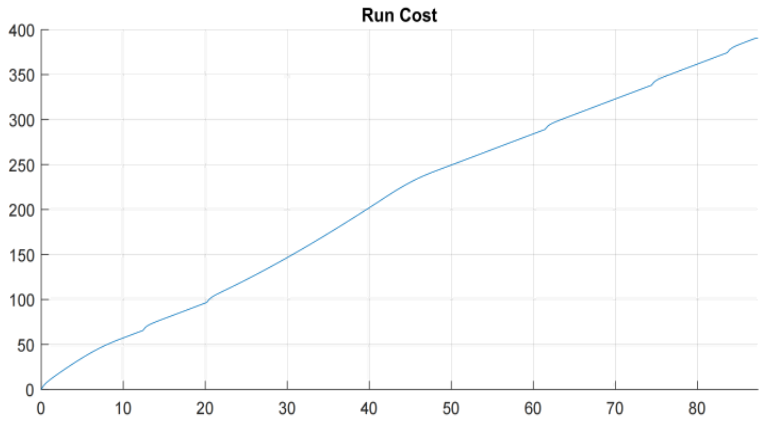


Fig. 34: Cost Throughout the Path.

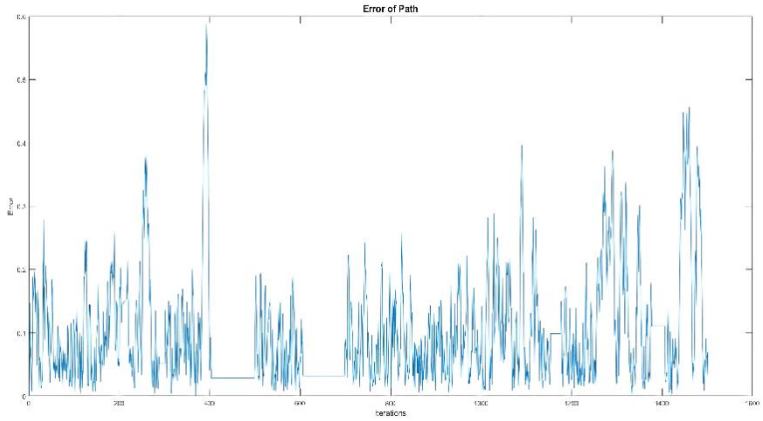


Fig. 35: Path Estimate Error.

5. Conclusion

In this paper, a closed-loop autopilot simulation system composed of guidance, navigation, and control modules is constructed in the MATLAB environment. The guidance module adopts dynamically weighted Dijkstra and B-spline interpolation to ensure the avoidance of off-road areas and trajectory trackability in the planning stage; the navigation module utilizes EKF to fuse GPS, odometer, camera and LiDAR to output vehicle position and traffic semantic labels; the control module generates speed and steering commands based on a simplified single front wheel/dual rear wheels model and energy budget and simulates dry and wet road braking. The control module generates speed and steering commands based on a simplified single front wheel/dual rear wheel model and energy budget, and simulates dry and wet road braking. After the synergy of the three modules, the system maintains low localization and tracking errors in the simulation of typical urban roads, and the planning response delay, trajectory smoothness, and energy consumption are up to the expectation, and the handling of random events such as congestion, signal switching, and GPS loss is also stable.

However, the dynamic weighted Dijkstra requires multiple subgraph recalculations under large-scale road network and high-frequency weight updating, which accumulates computational overhead and affects the real-time performance; the B-sample may have local oscillations in the dense area of curves, and the curvature constraints are still insufficient; and the model of vehicle dynamics and energy consumption is relatively simplified, which does not fully consider the tire-pavement friction changes, dynamics saturation, and energy recovery; The semantic and sensing data mainly come from the simulation settings and lack the support of real traffic flow.

For practical applications, the framework needs to be adapted to different vehicle models and complex urban road networks: on the vehicle side, the wheelbase, steering limit, mass and energy consumption curves can be replaced parametrically, and a higher fidelity dynamics/energy model can be introduced; on the environment side, a hierarchical or partitioned road map can be used, combined with a high-precision map and a signal phasing database, and weights can be updated incrementally by region, taking into account real-time and computational burdens. In the future, we can introduce machine learning or reinforcement learning for predictive semantic sensing of congestion, signal and pedestrian events, and adjust the weights online to reduce the frequency of replanning, further optimize the incremental planning strategy and more fine-grained dynamics and energy modeling, to improve real-time and scalability in large-scale scenarios, and validate the results on a small-scale real-vehicle platform to evaluate the robustness of the system under different sensor configurations and vehicle platforms, when the conditions allow. If possible, the validation can be carried out on a small-scale real vehicle platform to evaluate the robustness and energy efficiency performance under different sensor configurations and vehicle platforms.

Acknowledgement

Conceptualization, H. G. and J. Q.; methodology, H. G. and J. Q.; software, H. G. and J. Q.; validation, H. G. and J.Q.; formal analysis, H. G. and J.Q.; investigation, H. G. and J. Q.; data curation, H. G. And J. Q.; writing—original draft preparation, H. G. and J. Q.; writing—review and editing, H. G. and J.Q.; visualization, H. G. and J. Q.; supervision, J. Q.; All authors have read and agreed to the published version of the manuscript. J. Q. is the corresponding author.

References

[1] Y. Du, G. Liu, Y. Jiang, S. Cai and J. He, "Adaptive congestion index–based A* algorithm for dynamic vehicle path planning optimization", Physica A: Statistical Mechanics and its Applications, Vol. 672, (2025), Art. no. 130702, https://doi.org/10.1016/j.physa.2025.130702.

[2] G. Liu, M. Bilal, Q. Wang and X. Xu, "Congestion-Aware Path Planning With Vehicle-Road Cooperation in AIoV", IEEE Internet of Things Journal, Early Access, (2024), https://doi.org/10.1109/JIOT.2024.3446699.

[3] C. Li, C. Fan, M. Wang, J. Shen and J. Liu, "A Dynamic Shortest Travel Time Path Planning Algorithm with an Overtaking Function Based on VANET", Symmetry, Vol. 17, No. 3, (2025), p. 345, https://doi.org/10.3390/sym17030345.

[4] Y.-J. Kim, W.-J. Ahn, S.-H. Jang, M.-T. Lim and D.-S. Pae, "A Reinforcement Learning Approach to Dynamic Trajectory Optimization with Consideration of Imbalanced Sub-Goals in Self-Driving Vehicles", Applied Sciences, Vol. 14, No. 12, (2024), Art. no. 5213, https://doi.org/10.3390/app14125213.

[5] Min Luo, Xiaorong Hou and Jing Yang, "Surface Optimal Path Planning Using An Extended Dijkstra Algorithm", IEEE Access, Vol. 8, (2020), pp: 1–12, https://doi.org/10.1109/ACCESS.2020.3015976.

[6] Barbora Siemiątkowska, Rafał Więckowski, Jakub Rapcewicz and Jakub Kowaliński, "Semantic-Aware Path Planning with Hexagonal Grids and Vehicle Dynamic Constraints", Energies, Vol. 16, No. 13, (2023), Art.: 5127, https://doi.org/10.3390/en16135127.

[7] Andrzej Łebkowski, "Energy Consumption Optimization for an Electric Delivery Vehicle", Energies, Vol. 17, No. 22, (2024), Art.: 5665, https://doi.org/10.3390/en17225665.

[8] Truls Nyberg, Christian Pek, Laura Dal Col, Christoffer Norén and Jana Tumova, "Risk-Aware Motion Planning for Autonomous Vehicles with Safety Specifications", Proceedings of the IEEE Intelligent Vehicles Symposium, (2021), pp: 1016 – 1023, https://doi.org/10.1109/IV48863.2021.9575928.

[9] J. Jeong, J. Y. Yoon, H. Lee, H. Darweesh and W. Sung, "Tutorial on High-Definition Map Generation for Automated Driving in Urban Environments", Sensors, Vol. 22, No. 18, (2022), Art.: 7056, https://doi.org/10.3390/s22187056.

[10] H. Gu & J. Qu, "Real-time traffic management system based on YOLOv8 and DeepSORT," in Proceedings of the IEEE International Conference on Business and Industrial Research (ICBIR), to be printed in IEEE explore,2025.

[11] Z. Zhao, S. Gan, B. Xiao, X. Wang and C. Liu, "Three-Dimensional Reconstruction of Zebra Crossings in Vehicle-Mounted LiDAR Point Clouds", Remote Sensing, Vol. 16, No. 19, Art. 3722, 2024, https://doi.org/10.3390/rs16193722.

[12] S. Ding and J. Qu, "Research on Multi-tasking Smart Cars Based on Autonomous Driving Systems", SN Computer Science, Vol. 4, Art. 292, 2023, https://doi.org/10.1007/s42979-023-01740-1.

[13] S. Cao, X. Lu and S. Shen, "GVINS: Tightly Coupled GNSS-Visual-Inertial Fusion for Smooth and Consistent State Estimation", IEEE Robotics and Automation Letters, Vol. 7, No. 1, pp. 1–8, 2022, https://doi.org/10.1109/LRA.2022.3224367.

[14] Y. Li and J. Qu, "A novel neural network architecture and cross-model transfer learning for multi-task autonomous driving", Data Technologies and Applications, Vol. 58, No. 5, (2024), pp: 693–717, https://doi.org/10.1108/DTA-08-2022-0307.

[15] H. Wang, S. Lou, J. Jing, Y. Wang, W. Liu and T. Liu, "The EBS-A* algorithm: An improved A* algorithm for path planning", PLoS ONE, Vol. 17, No. 2, (2022), pp: e0263841, https://doi.org/10.1371/journal.pone.0263841.

[16] A. H. Tanira and I. M. I. AbuHadrous, "An Improved Sampling Dijkstra Approach for Robot Navigation and Path Planning", International Journal of Intelligent Systems and Applications (IJISA), Vol. 15, No. 6, (2023), pp: 51 – 64, https://doi.org/10.5815/ijisa.2023.06.05.

[17] V. P. Damle and S. Susan, "Dynamic Algorithm for Path Planning using A-Star with Distance Constraint", in Proc. 2022 2nd Int. Conf. Intell. Technol. (CONIT), (2022), pp: 1 – 5, https://doi.org/10.1109/CONIT55038.2022.9847869.

[18] C. Hu, Z. Wang, H. Taghavifar, J. Na, Y. Qin, J. Guo and C. Wei, "MME-EKF-Based Path-Tracking Control of Autonomous Vehicles Considering Input Saturation", IEEE Trans. Veh. Technol., Vol. 68, No. 6, (2019), pp: 5246 – 5259, https://doi.org/10.1109/TVT.2019.2907696.