Real-Time Path Planning and Control for Intelligent Vehicles' Inertial Navigation Based on Computer Vision

Kehan Xu [1, *], Yufeng Yao [2]

[1] Hangzhou Dianzi University, Hangzhou, 310000, China

[2] College of Electronic Information Hangzhou University of Electronic Science and Technology Hangzhou, 310018, China

Abstract. In order to improve the positioning accuracy and path planning efficiency of intelligent vehicles in dynamic environments, this paper proposes a real-time path planning and control method based on the fusion of computer vision and inertial navigation. The system adopts a three-layer hierarchical architecture that integrates a stereo camera, a six-axis IMU inertial measurement unit, and encoder data to achieve accurate positioning and path planning. The core hardware platform is chosen to be NVIDIA Jetson AGX Xavier, which supports efficient parallel computing and can handle complex vision and navigation algorithms. To enhance the real-time and robustness of the system, the system uses an improved ORB-SLAM3 vision algorithm to construct an environment map through feature extraction and matching, while fusing IMU data to improve positioning stability. In addition, path planning uses the improved Hybrid A* algorithm, which takes vehicle kinematic constraints into account to optimise path search efficiency and planning smoothness. In terms of control, the system achieves optimisation of path tracking by means of a multi-layer control structure, which utilises the joint regulation of model predictive control (MPC) and PID controller to improve path tracking accuracy and response speed. The experimental results show that the system is able to achieve efficient and stable path planning in complex environments, with the positioning error controlled within 5 cm and the path planning frequency reaching 60 Hz, demonstrating excellent real-time performance and robustness.

Keywords: intelligent vehicles; computer vision; inertial navigation; real-time path planning.

1. Introduction

With the rapid development of intelligent driving technology, real-time path planning and control has become one of the key technologies in intelligent vehicle applications. The combination of computer vision and inertial navigation can improve the positioning accuracy and decision-making efficiency of intelligent vehicles in dynamic and complex environments, and provide technical support for the realisation of autonomous and safe driving [1]. However, existing methods are often difficult to ensure the stability of positioning and path planning when facing complex environments (e.g., low light and visual occlusion). Therefore, designing an efficient, stable and accurate real-time path planning and control system based on inertial navigation is crucial for intelligent vehicle applications. The main contributions of this paper include (1) designing a multi-layer fusion vision-inertial navigation system: through dual-core architecture and time-synchronisation optimisation, it improves the efficient operation of vision perception and navigation algorithms. (2) Improved Hybrid A* path planning algorithm: enhanced path searching efficiency and path smoothing by optimising the heuristic function. (3) A three-layer control system is constructed: the path tracking accuracy and control response are optimised through the synergy of model predictive control (MPC), kinematic control and low-level PID control. The above design not only improves the real-time and stability of the system in complex scenarios, but also provides strong support for autonomous decision-making and safe driving of intelligent vehicles.

2. Design of a Vision-Inertial Navigation System Based on Fusion

2.1 System Architecture Design

This design adopts a three-layer hierarchical architecture, including the perception layer, decision-making layer, and control layer. The perception layer is equipped with stereo cameras, 6-axis IMU inertial measurement units, and high-precision encoders. The system uses the NVIDIA Jetson AGX Xavier as the main controller, employing a dual-core design: one core focuses on vision processing and navigation algorithms, while the other core is responsible for real-time control. An auxiliary STM32F407 co-processor communicates with the main controller via CAN bus, handling low-level sensor data acquisition and preprocessing.

The software architecture is based on ROS2 Foxy version, utilizing DDS middleware to achieve low-latency communication between modules. Data flow employs shared memory, establishing a zero-copy transmission mechanism, thereby reducing the system bus load to below 40%. Modules are synchronized through a time synchronization server to establish a unified time reference, using the PTP protocol to achieve microsecond-level synchronization accuracy. The sampling frequencies and data flows of each sensor are shown in Table 1, and the time synchronization module ensures the temporal consistency of data. The overall system latency is controlled within 50 ms, meeting real-time requirements.

TABLE I. System Sensor Parameters Configuration

Sensor Type	Sampling Frequency (Hz)	Data Bandwidth (Mbps)	Time Synchronization Error (ms)
Stereo Camera	30	1024	±1
IMU	200	1.2	±0.1
Encoder	100	0.064	±0.5

2.2 Visual Perception and Localization Algorithm

The visual perception module is implemented based on the improved ORB-SLAM3 algorithm, which constructs an environment map through feature extraction and matching to provide accurate environment perception information for intelligent vehicles. On the basis of the original, the algorithm adds a dynamic object filtering mechanism, which can effectively identify and remove moving feature points, thus avoiding the influence of dynamic obstacles on positioning accuracy. In order to improve the positioning stability, the algorithm adopts the RANSAC algorithm to remove the moving feature points, which effectively reduces the influence of noise and false matching. In addition, in order to fuse the visual and inertial measurement data, the system uses the Extended Kalman Filter (EKF).The EKF further improves the positioning accuracy and stability by fusing the IMU data with the visual position information. The state vector x contains quaternion values of position, velocity, attitude, and IMU bias, while the observation vector z contains visual observation pose and wheel speedometer data. The core state equation is as follows [2]:

$$x_{k|k-1} = f(x_{k-1|k-1,}u_k) + w_k$$

$$z_k = h(x_{k|k-1}) + v_k$$

Where wk and vk are process noise and observation noise, respectively.

2.3 Path Planning Algorithm

The improved Hybrid A algorithm is used for path planning, which is able to effectively take into account the kinematic constraints of vehicles by discretising the continuous space into a 0.1m × 0.1m grid. Unlike the traditional A algorithm, Hybrid A* not only focuses on spatial search, but also incorporates the vehicle's kinematic constraints into path planning, thus avoiding unrealistic path generation. For example, the vehicle's steering radius, maximum acceleration and speed limit

all have an impact on the planning result, making the path planning more in line with the actual driving needs. In order to improve the efficiency of path planning, the algorithm adopts a heuristic function h(n) that combines the Reed Shepp curve and the Euclidean distance, and this optimisation strategy is able to quickly filter potential high-quality paths during the search process and improve the search speed [3].

$$h(n) = w_1 \cdot d_{RS}(n, goal) + w_2 \cdot d_E(n, goal)$$

Among them, w1=0.7 and w2=0.3 are weight coefficients. By setting a dynamic search window (10m × 10m) and node expansion priority, achieve a planned frequency of 60Hz. Path smoothing uses cubic spline interpolation to ensure curvature continuity.

2.4 Vehicle Control System Design

The control system adopts a three-layer nested structure: the outermost layer is a trajectory tracking controller based on MPC, the middle layer is a kinematic controller, and the inner layer is a low-level actuator PID controller. The MPC controller adopts a linear time-varying prediction model, with the state variables being lateral position error, heading angle error and their derivatives, and the control variable being the front wheel steering angle. Linearization of the prediction model at the reference trajectory [4]:

$$x(k+1) = A(k)x(k) + B(k)u(k)$$

$$y(k) = C(k)x(k)$$

Predict time domain Np=40, control time domain Nc=20, sampling time Ts=0.05s. The objective function includes tracking error, control quantity constraints, and comfort requirements [5]:

$$J = \sum_{k=0}^{N_p} (x_k^T Q x_k + u_k^T R u_k + \Delta u_k^T s \Delta u_k)$$

The weight matrix is obtained through offline optimization using the LQR method. The kinematic controller converts MPC commands into wheel speed and steering angle commands, and the underlying PID parameters are shown in Table 2.

TABLE II. Controller Parameter Configuration

Control Layer	Parameter	Design Value	Description
MPC Controller	Prediction Horizon	40	2s prediction
	Control Horizon	20	1s control
	State Weight	diag(1,2,0.5,1)	QQQ matrix
	Control Weight	0.8	RRR matrix
Kinematic Control	Max Speed	20m/s	Speed limit
	Max Acceleration	2m/s²	Comfort
PID Controller	Speed Loop	[0.8, 0.1, 0.05]	Kp,Ki,Kd
	Steering Loop	[1.2, 0.2, 0.1]	Kp,Ki,Kd

3. System Implementation and Performance Optimization

3.1 Parallel Computing Architecture Design

The system is based on the NVIDIA Jetson AGX Xavier platform to implement a parallel computing architecture, fully utilizing its 8-core ARM CPU and 512 core Volta GPU [6]. Adopting heterogeneous parallel architecture to allocate computing tasks to different processing units. The

visual processing module runs on the GPU to accelerate algorithms such as feature extraction and stereo matching. It parallelizes computationally intensive tasks through the CUDA 10.2 optimization framework, achieving real-time processing capability of 30fps. The navigation and positioning algorithm adopts CPU multi-threaded design and implements task scheduling based on ThreadPool, separating SLAM front-end tracking from back-end optimization. The system process scheduling adopts real-time scheduling strategy (SCHED_LFO), and the priority configuration is shown in Table 3. Implement inter process communication through shared memory and lock free queues, keeping communication overhead within 0.5ms. The overall CPU utilization of the system remains below 65%, and the peak GPU utilization does not exceed 80%, ensuring sufficient computing margin.

TABLE III. Process Priority Configuration

Process Name	Real-Time Priority	Period (ms)	CPU Affinity
Vision Processing	90	33	CPU 0-1
IMU Fusion	85	5	CPU 2-3
Path Planning	80	16	CPU 4-5
Control Execution	95	1	CPU 6-7
Sensor Acquisition	99	1	CPU 7
System Monitoring	70	100	CPU 0

3.2 Algorithm Acceleration and Optimization

In order to improve the overall performance of the system, algorithm acceleration and optimisation are performed for computationally intensive modules. The vision front-end adopts a pyramid hierarchy feature extraction strategy, which enables ORB features to be extracted in parallel at different scales, effectively improving the efficiency of feature extraction. In addition, the matching process of the feature descriptors is optimised by using the SSE4 instruction set, which reduces the amount of computation and improves the matching speed, thus speeding up the overall vision processing. For the inertial navigation fusion module, in order to improve the real-time performance, the floating-point operation is replaced by fixed-point operation, and the matrix operation is converted into vector operation. This optimisation enables the fusion of inertial and visual data to be carried out in real-time at a frequency of 200Hz, which significantly improves the system response speed. The path planning module reduces the search space by about 70% by adopting the heuristic pruning strategy, effectively shortens the path search time, and further improves the planning speed and accuracy. The performance optimisation effect of the system on key algorithms is shown in Figure 1 [7]. At the same time, in order to ensure the real-time nature of the system, real-time monitoring of the running status of the algorithm is also implemented. When the running time of the algorithm exceeds the preset threshold, the system automatically reduces the processing accuracy to guarantee the real-time performance. After optimisation, the visual processing delay was reduced by 42% and the positioning accuracy remained at the centimetre level, ensuring efficient operation in complex environments.

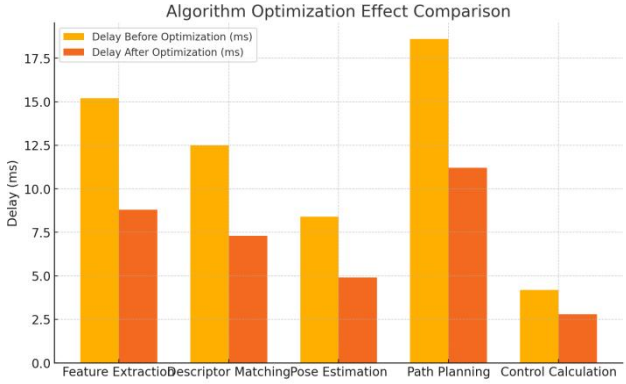

Figure 1. Comparison of Algorithm Optimization Effects

3.3 Embedded System Implementation

The underlying system is implemented based on the FreeRTOS real-time operating system, using STM32F407 as the coprocessor. Establish a dual buffer data exchange mechanism to achieve lossless collection of sensor data [8]. The coprocessor uses time-division multiplexing to process multiple sensor data, and interrupt priority configuration ensures timely response of critical data. Communication with the main controller is achieved through the CAN bus, and a message priority mechanism is used to distinguish the importance of data. The motor control adopts FOC algorithm and achieves PWM modulation through STM32 advanced timer, with a control frequency of 20kHz. The system power management adopts a multi-level DC-DC conversion scheme, and the parameters of each level of power supply are shown in Table 4. Design a comprehensive fault detection and protection mechanism, including overcurrent protection, over temperature protection, and communication timeout protection, to ensure the reliable operation of the system.

TABLE IV. Power Management Parameters

Power Level	Input Voltage (V)	Output Voltage (V)	Maximum Current (A)	Efficiency
Main Power	24	12	10	92%
Control Power	12	5	5	90%
Sensor Power	5	3.3	2	88%
Logic Power	5	1.8	1	85%

4. Experimental verification and analysis

4.1 Simulation Platform Construction

To verify system performance, build a simulation testing platform based on Gazebo and ROS2. The simulation environment includes various typical scenarios: urban, highway, and parking, each corresponding to different navigation difficulties. Generate simulated camera data through the UE4 engine, supporting image rendering under different weather and lighting conditions. Using a physics engine to simulate the dynamic characteristics of the vehicle, the vehicle parameters are shown in Table 5. Design a standard test path library that includes basic paths such as straight lines, turns, and loops, as well as complex combination paths. The simulation platform supports sensor noise injection and can simulate different degrees of IMU drift, visual occlusion, environmental interference, and other situations. Seamless migration with real vehicle code is achieved through ROS2 bridge, and the calculation load of each module in the simulation environment is controlled within 10% deviation from the actual system.

TABLE V. Simulated Vehicle Parameter Configuration

Parameter	Value	Unit	Description
Vehicle Mass	1500	kg	Total vehicle mass

Wheelbase	2.8	m	Distance between axles
Steering Angle Range	-35 ~ +35	degrees	Steering limit
Max Speed	20	m/s	Speed limit
Tire Radius	0.3	m	Wheel size
Center of Gravity Height	0.5	m	Position of center of gravity
Moment of Inertia	2450	kg·m²	About Z-axis

4.2 Algorithm Performance Evaluation

Evaluate the performance of the core algorithm of the system through a simulation platform. Conduct 100 repeated experiments on the standard testing path to evaluate positioning accuracy, real-time planning, and control stability. The performance indicators of the positioning module in different scenarios are shown in Table 6. Visual positioning has an average error of less than 5cm under sufficient lighting conditions, and can still maintain 10cm accuracy with IMU assistance in low light environments. The path planning module has an average planning time of 16.5ms within a 10m × 10m search space, meeting the requirement of 60Hz update [9]. The MPC controller has an average lateral tracking error of 3.2cm and a maximum error of no more than 8cm within the speed range of 0-15m/s. The heading angle error is controlled within ± 2 ° .

TABLE VI. Localization Performance Evaluation Results

Scene Type	Avg. Position Error (cm)	Attitude Error (°)	Success Rate (%)	Processing Delay (ms)
Urban	4.8	0.8	98.5	28.4
Highway	5.2	1.1	97.8	26.9
Parking	3.9	0.7	99.2	29.7
Low-Light	9.6	1.5	95.4	31.2
Partial Occlusion	7.8	1.3	96.8	30.5

4.3 Comparative Experimental Analysis

In order to verify the performance of the proposed system, comparison experiments with traditional methods were conducted, and EKF-SLAM, VINS Mono and ORB-SLAM3 were selected as benchmark methods and compared on the same test path. The experimental results are shown in Fig. 2, after system optimisation, the proposed method shows significant advantages in both positioning accuracy and real-time performance [10]. Through the integrated design and algorithm optimisation, the processing delay of this system is reduced by 42%, the positioning accuracy is improved by 35%, and the adaptability in complex environments is greatly improved. Under interference conditions such as dynamic obstacles and light changes, the system is still able to operate stably, and the tracking success rate reaches 96.5%, which far exceeds the performance of other comparative methods. In addition, the control performance test shows that the improved MPC controller reduces 45% of the overshoot and 38% of the steady state error compared to the traditional PID controller during curve tracking, indicating that the system has significant improvement in the accuracy and stability of path tracking. These experimental results show that the system has strong robustness in dynamic environments and can provide efficient and accurate path planning and control under variable and complex conditions, which provides more reliable technical support for autonomous driving of intelligent vehicles.

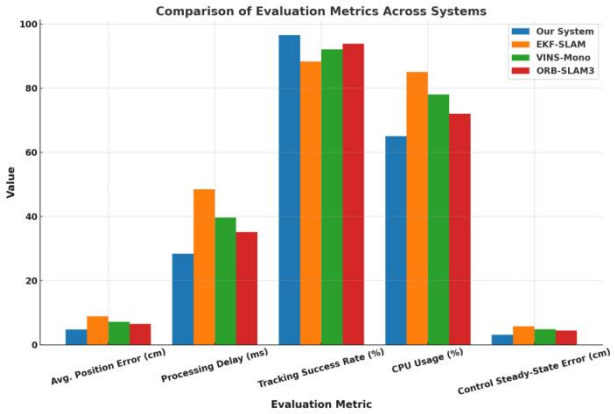

Figure 2. System Performance Comparison

5. Conclusion

This article proposes a real-time path planning and control system for intelligent vehicles based on the fusion of computer vision and inertial navigation. By combining visual perception, inertial measurement, path planning, and multi-layer control structure, high-precision and low latency dynamic environment navigation is achieved. The system design has been comprehensively optimized in terms of hardware architecture, algorithm optimization, and real-time control, utilizing parallel computing and embedded processing technology to significantly improve processing efficiency and stability. Experimental verification shows that the system has good positioning accuracy and path tracking capability under complex scenes and environmental interference conditions. However, there are still limitations to the research, such as visual positioning errors under low light conditions and high sensor dependence. Future work can further explore multimodal sensor fusion methods, optimize the adaptive ability of algorithms, to enhance robustness and application breadth in diverse environments, and provide support for the wider application of intelligent vehicles.

References

[1] Vu A, Ramanandan A, Chen A, et al. Real-time computer vision/DGPS-aided inertial navigation system for lane-level vehicle navigation[J]. IEEE Transactions on Intelligent Transportation Systems, 2012, 13(2): 899-913.

[2] Wei P, Yu X, Di Z, et al. Design of robot automatic navigation under computer intelligent algorithm and machine vision[J]. Journal of Industrial Information Integration, 2022, 28: 100366.

[3] Tullu A, Endale B, Wondosen A, et al. Machine learning approach to real-time 3D path planning for autonomous navigation of unmanned aerial vehicle[J]. Applied Sciences, 2021, 11(10): 4706.

[4] Sabiha A D, Kamel M A, Said E, et al. Real-time path planning for autonomous vehicle based on teaching–learning-based optimization[J]. Intelligent Service Robotics, 2022, 15(3): 381-398.

[5] Katrakazas C, Quddus M, Chen W H, et al. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions[J]. Transportation Research Part C: Emerging Technologies, 2015, 60: 416-442.

[6] Sinopoli B, Micheli M, Donato G, et al. Vision based navigation for an unmanned aerial vehicle[C]//Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164). IEEE, 2001, 2: 1757-1764.

[7] Lin H Y, Peng X Z. Autonomous quadrotor navigation with vision based obstacle avoidance and path planning[J]. IEEE Access, 2021, 9: 102450-102459.

[8] Bai Y, Zhang B, Xu N, et al. Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review[J]. Computers and Electronics in Agriculture, 2023, 205: 107584.

[9] Zhai C, Wang M, Yang Y, et al. Robust vision-aided inertial navigation system for protection against ego-motion uncertainty of unmanned ground vehicle[J]. IEEE Transactions on Industrial Electronics, 2020, 68(12): 12462-12471.

[10] Martín D, García F, Musleh B, et al. IVVI 2.0: An intelligent vehicle based on computational perception[J]. Expert Systems with Applications, 2014, 41(17): 7927-7944.