Eng Int Syst (2026) 2: 273–284

Engineering
Intelligent Systems

Robot Information Intelligent Perception and Navigation Based on Multi-Sensor Fusion

Xibin Li[1,4,a*], **Yiyang Luo**[2,b], **Mingsong Bao**[1,c] **and Haoming Sun**[3,d]

[1] *School of Safety Engineering, China University of Mining and Technology, Xuzhou 221116, Jiangsu, China*
[2] *Purchasing Department, Sinopec International Business Tianjin Co., Ltd., Tianjin 300042, Tianjin, China*
[3] *Technical Research and Development Department, Shandong Tesla Robot Co., Ltd., Yantai 264006, Shandong, China*
[4] *Technical Research and Development Department, Shandong Guoxing Smartech Co., Ltd., Yantai 264006, Shandong, China*

Various sensors are different in terms of time synchronization, data dimension, and sampling frequency, which makes the deep fusion of heterogeneous data difficult. In addition, the existing high-precision fusion algorithms rely heavily on computational resources and cannot meet the needs of lightweight robots with limited computing power. To address these issues, this research work applies a fusion method based on a multimodal convolutional neural network (2D-ResNet-50 + 3D-CNN) and a cross-modal attention mechanism to ensure the synchronization and unified format of multi-sensor data by means of data preprocessing and time alignment technology. Then, a convolutional neural network is used to extract features from visual image data, laser radar point cloud data, and inertial measurement unit (IMU) data, and the information from different sensors is fused through a cross-modal attention mechanism. A modular architecture is used to optimize the computational efficiency of the system. The system is divided into multiple independent modules, each responsible for a specific task. The event trigger mechanism is used to dynamically activate and schedule related modules to enhance the system's intelligence. This method is deployed on NVIDIA's Jetson Xavier NX platform, and the experiments are conducted under the Robot Operating System framework. Experiments show that the robot's control error does not exceed 0.25 when performing path tracking tasks. The path planning time in various environments does not exceed 150 milliseconds. This method can improve perception precision while maintaining high real-time performance and efficiency with limited computational resources, significantly optimizing the robot's navigation performance in complex dynamic environments.

Keywords: multi-sensor fusion, robot navigation, deep learning, cross-modal attention mechanism, modular architecture

1. INTRODUCTION

In recent years, robots have been widely used in intelligent manufacturing, smart logistics, urban inspection, etc., especially in dynamic and complex environments where the demand for high-precision, intelligent perception and autonomous navigation continues to rise [1–2]. According to Statista data, the annual growth rate of the global service robot market exceeds 25%, and the requirements for environmental perception precision and system real-time performance are increasing simultaneously. Traditional robot systems that rely on a single sensor (such as a camera or laser radar) often have perception blind spots or error accumulation when faced with lighting changes, obstructions, uneven terrain, etc., which seriously affects the reliability and efficiency of task execution [3–4]. For example, robots often rely on laser radar for positioning and obstacle detection in logistics and warehousing scenarios. However, the ranging precision of laser radar sensors is affected by ambient light. In environments with intense light, the laser radar cannot effectively reflect light back to the sensor, causing the robot to lose accurate environmental data and thus fail to position itself. Therefore, integrating multiple heterogeneous sensors to improve information completeness and robustness has become a key direction for research on robot intelligent perception and navigation [5–6].

*Corresponding Author. [a]Email: lixibin1980@126.com, [b]wzluoyy@sinopec.com, [c]msb@suprobot.com, [d]hmsun@tslrobot.com

The current research on multi-sensor fusion systems faces two core issues. The first is the difficulty of strong collaboration due to sensor heterogeneity. Different sensors (images, point clouds, inertial data) have natural differences in regard to sampling frequency, data structure, and time synchronization [7–8]. Traditional filtering or simple concatenating methods cannot achieve truly effective data fusion [9–10]. Second, the algorithm computing power bottleneck limits the practicality of the fusion system [11–12]. For example, although the extended Kalman filter and graph optimization algorithm perform well in terms of precision, they have high requirements for central processing unit (CPU) or graphics processing unit (GPU) resources. They cannot be deployed in real time on resource-constrained lightweight mobile robots [13–14]. These problems make it difficult for existing systems to balance stability and deployment efficiency in complex environments, becoming a core obstacle to their engineering implementation [15–16]. Although existing research has made some progress, there is still a discrepancy between lightweight and high precision. That is, under high precision requirements, the system often requires a large number of computational resources, which makes it impossible to apply it in scenarios with high requirements for low power consumption and real-time performance. Reducing the algorithm's demand for computational resources while ensuring fusion precision is a significant challenge in current research.

To address the above challenges, academia and industry have proposed various multimodal data fusion methods [17–18]. For example, some studies use deep learning to build fusion perception models, such as fusion convolutional neural networks to extract image and point cloud features or Transformer structures to achieve cross-modal information alignment [19–20]. Tao et al. [21] combined visual and inertial information for precise positioning through a graph optimization framework. Although these methods have made some progress in specific scenarios, they still have problems such as complex structure, low inference efficiency, and weak model generalization ability [22–23]. In particular, in embedded deployment scenarios that require low delay and high real-time performance, there is still a lack of general, efficient, and lightweight solutions [24–25]. Therefore, it is urgent to build a fusion perception framework that can balance precision, real-time performance, and resource efficiency [26–27]. Specifically, it can meet robots' perception and navigation needs in dynamic and complex environments while maintaining high performance despite limited computational resources [28–29].

To address the above problems, this paper studies a perception and navigation system that integrates a multimodal convolutional neural network and a cross-modal attention mechanism to solve the problems of difficulty in coordinating heterogeneous sensor data and high computing power dependence. This paper first unifies the multi-source information format through data preprocessing and a time alignment mechanism to ensure the temporal consistency of input data. Then, the visual convolutional neural network (CNN), PointNet point cloud network and the long short-term memory network (LSTM) inertial network are used to extract sensor features, and the deep collaborative fusion of heterogeneous information is achieved by means of the cross-modal attention mechanism. Finally, a dynamic scheduling system is constructed by combining a modular architecture and an event-triggering mechanism to reduce the number of invalid calculations. Experimental verification shows that this method can maintain high real-time performance and efficiency despite limited computational resources while improving perception precision. The engineering application of this technology can promote the widespread application of robotics in multiple industries, such as intelligent manufacturing, smart logistics, and urban inspection, and provide a feasible solution for autonomous robotic navigation in complex dynamic environments.

2. MULTI-SOURCE SENSOR DATA ACQUISITION AND SPATIOTEMPORAL SYNCHRONIZATION PREPROCESSING

2.1 Multi-Source Sensor Data Acquisition

The system integrates three types of heterogeneous sensors: camera, laser radar, and IMU, which are used to obtain visual images, spatial point clouds, and inertial motion information, respectively. Each sensor is asynchronously connected through the robot operating system (ROS) node, and a multi-threaded asynchronous mechanism is used to complete parallel acquisition. The camera obtains image frames through GStreamer combined with OpenCV, with a sampling rate of 30 fps and a resolution of 640×480. The image frames are accompanied by hardware timestamps for subsequent alignment. The laser radar is set to scan at a frequency of 10Hz, and the output format is a standard point cloud data (PCD) frame, which is accessed in real time through the device driver in the form of an event callback. The IMU sensor acquires acceleration and angular velocity at a frequency of 200Hz, and transmits data using serial communication. All data is accompanied by sampling clock information. Various sensor data are cached in a shared queue and accessed by the processing module according to the timestamp index, achieving a high-throughput and low-coupling data scheduling mechanism.

Table 1 shows the data acquired from the inertial measurement unit. The timestamp shows the time when the data was acquired, which is based on the time when the event occurred in the robot system. It helps to synchronize and align the data from different sensors. The acceleration data (ax, ay, and az) indicates the acceleration of the robot in three directions. The acceleration of the Z-axis is closely related to gravity. If the robot is stationary, the acceleration of the Z-axis is approximately equal to 9.81m/s^2. The accelerations in the other two directions help determine the robot's movement state. The angular velocity data (gx, gy, and gz) describes the robot's rotation rate on each axis. Through these angular velocity data, the robot's posture changes (roll, pitch, and yaw) can be calculated. These data are crucial for dynamic navigation and path planning, helping the robot

Table 1 Data acquired from the IMU sensor.

ID	Timestamp (ms)	Acceleration ax (m/s^2)	Acceleration ay (m/s^2)	Acceleration az (m/s^2)	Angular Velocity gx (rad/s)	Angular Velocity gy (rad/s)	Angular Velocity gz (rad/s)
1	1000	0.12	0.05	9.81	0.02	0.01	0
2	1005	0.1	0.04	9.82	0.03	0.01	0.01
3	1010	0.11	0.05	9.8	0.02	0	0.01
4	1015	0.1	0.06	9.82	0.03	0.01	0
5	1020	0.09	0.05	9.81	0.01	0	0.02
6	1025	0.12	0.05	9.81	0.02	0.02	0
7	1030	0.13	0.06	9.8	0.01	0.01	0.01
8	1035	0.11	0.05	9.83	0.02	0	0
9	1040	0.1	0.04	9.82	0.01	0.01	0.01
10	1045	0.09	0.05	9.81	0	0	0.02

understand its motion state in three-dimensional space in real time.

Filtering algorithms are designed to remove acquisition errors and sensor noise from the three types of raw data. Edge-preserving adaptive bilateral filtering is used for image data, and the filter kernel is dynamically adjusted in combination with the local gradient statistical characteristics to effectively suppress background noise while maintaining key structures. The point cloud data is first downsampled by VoxelGrid voxel filtering (with a voxel size of 0.1m) to reduce the number of points, and then the statistical outlier removal algorithm is applied. The minimum number of neighbors is set to 20, and the standard deviation threshold is set to 1.0. Isolated invalid points are removed. The IMU data input six-dimensional vector is processed by a third-order Butterworth low-pass filter with a cutoff frequency set to 20Hz to filter out jitter and high-frequency electromagnetic interference. The filtering operation is preprocessed in the data acquisition thread, and the single processing delay is controlled within 2ms.

2.2 Spatiotemporal Synchronization Preprocessing

All sensor data needs to be standardized to unify the numerical scale of various modal features. Image features are normalized independently using RGB channels. The mean and variance use ImageNet statistics $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$ to maintain consistency with the pre-training model. The point cloud coordinates are normalized by the unit sphere; that is, each frame of the point cloud uses the center of mass as a reference, and all coordinates are mapped to the range of $[-1, 1]$ to keep the geometric shape unchanged. A sliding window normalization strategy is used for the IMU data, and the mean and standard deviation of each dimension in the last second are counted to perform dynamic normalization to eliminate the numerical jump caused by sudden changes in movement.

The temporal consistency of the sensor data is ensured by the following strategy: the image frame timestamp (30Hz) is used as the main alignment reference. Within the time window corresponding to each frame of the image, the most recent laser radar point cloud frame is selected. If the deviation exceeds 33ms, linear interpolation is used to reconstruct the key points in time [30–31]. For IMU data, 5 data points are acquired during the time window corresponding to the image. Bilinear interpolation and then sliding average are performed to generate the inertial state at the corresponding moment. If any modal deviation exceeds the threshold of 50ms, the data frame discard mechanism is triggered, and the cache queue is automatically polled to pull the next batch of data. The alignment operation is completed based on the message synchronization filter in ROS. The interpolation and window matching logic uses time as the unique index to ensure that multi-source data is precisely matched in semantic time.

3. MULTIMODAL FEATURE LEARNING AND CROSS-MODAL FUSION

3.1 Multimodal Feature Extraction

After normalization, the input image is sent to the deep convolutional neural network Residual Network-50 (ResNet-50). The network adopts a five-level residual structure. The first four levels of convolution blocks are used for low-level feature extraction (edge, texture, and contour), and the fifth level outputs 1024-dimensional semantic features. Global average pooling is used instead of the fully-connected layer to reduce parameter redundancy and improve generalization ability. The output features are L2-normalized in the channel dimension to suppress local strong responses and ensure global consistency. The feature tensor dimension is finally $B \times 1024$, where B is the batch size. Figure 1 presents image feature extraction.

Figure 1 shows the process of image input being processed by the ResNet-50 deep convolutional neural network. First, the original image is input into the network. The image is normalized before entering the network, scaling the pixel values to a certain range so that the neural network can better process the data. The normalized image enters the ResNet-50 network. ResNet-50 is a 50-layer convolutional neural network that uses a residual structure to help avoid the gradient vanishing problem when training deep networks. Convolutional layers 1 to 4 comprise the low-level feature extraction stage. Through multiple convolution operations,

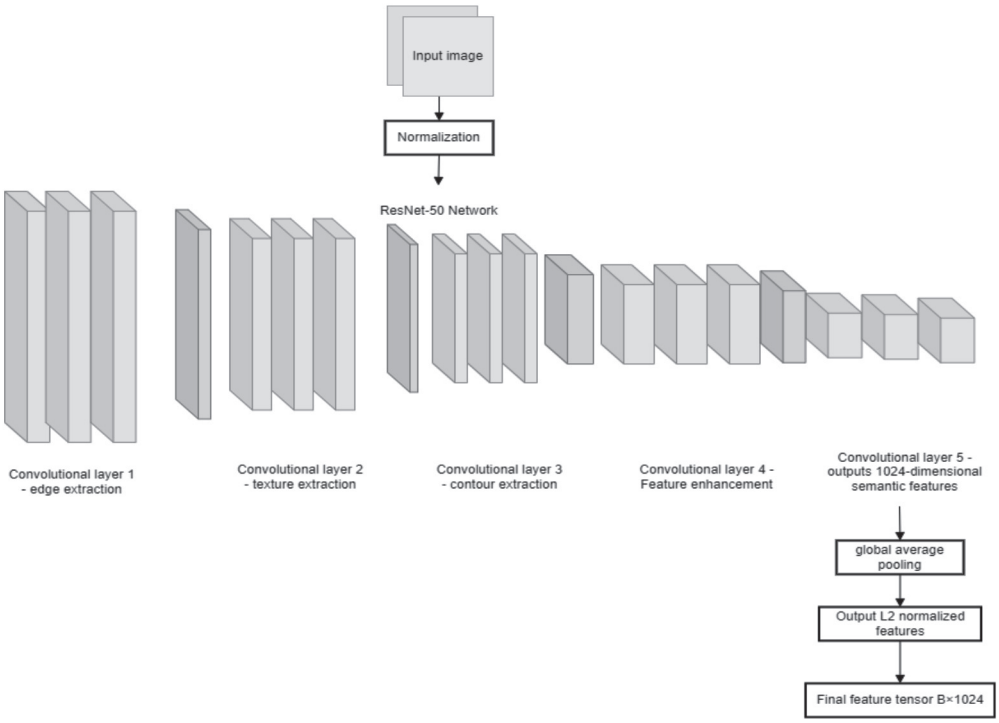

Figure 1 Deep convolutional neural network processing.

the network extracts different low-level features in the image, such as edges, textures, and contours.

The point cloud data is first sampled and reordered, and then divided into a fixed number of points N ($N = 1024$) per frame. The PointNet++ architecture is used for three-stage spatial feature learning. The Set Abstraction module constructs a local point set through farthest point sampling (FPS) and Ball Query grouping. The PointNet subnetwork extracts geometric features within the region. The Feature Propagation module performs feature upsampling and fusion between different scales. Finally, the global semantic representation of each frame point cloud is output with a dimension of $B \times 1024$. During the feature extraction process, the spatial relationship between points is preserved, and the boundary expression is enhanced, thereby improving the robot's ability to model the shape and spatial distribution of objects in sparse structures [32–34]. The IMU data is divided into fixed-length sequences according to the time window ($T = 100$ frames), and the input dimension is $B \times T \times 6$ (three-axis acceleration and three-axis angular velocity). The bidirectional gated recurrent unit (bidirectional GRU) network is used to model the time series. After the two hidden states of the forward and backward directions are merged, they are compressed into $B \times 256$ dynamic state features through the fully connected layer. Dropout ($p = 0.2$) is used to prevent overfitting. LeakyReLU is used for the activation function. This structure enhances the time perception ability of the inertial mode and effectively characterizes the continuous action characteristics of the robot, such as acceleration and steering.

The three types of features—image, point cloud, and IMU—are projected to a unified dimension ($D = 512$) by means of a linear layer. During this transformation, the semantics of the features remain unchanged, and only dimension compression and channel unification are completed to facilitate subsequent aggregation. The projection layer parameters participate in back propagation during the training phase to ensure that the projection results maintain high information density in the shared representation space. The three types of features are set to $rmf_{\text{img}}, \text{f}_{\text{pc}}, \text{f}_{\text{imu}} \in \mathbb{R}^{\text{B}\times 512}$, which are represented as $\tilde{\text{f}}_{\text{img}}, \tilde{\text{f}}_{\text{pc}}, \tilde{\text{f}}_{\text{imu}}$, respectively, after linear mapping.

3.2 Dynamic Allocation of Cross-modal Attention Weights

The cross-modal attention mechanism is used to weight the three types of features through learnable weights. The specific process is as follows: each modal feature is used as Query (Q), and the other two are used as Key (K) and Value (V).

The scaled dot-product attention is executed to calculate the attention distribution:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where $d_k = 512$ is the dimension of the Key vector. A three-modal fusion operation is performed on each modality to form enhanced feature representations f^*_{img}, f^*_{pc}, and f^*_{imu}.

Using the gated fusion strategy, the three enhanced features are concatenated and sent to the gated network (two layers of full connection + Sigmoid), and the final fusion weight vector $\omega \in \mathbb{R}^3$ is output, satisfying $\sum_i \omega_i = 1$.

The final fusion feature is expressed as:

$$f_{fusion} = \omega_{\text{img}} \cdot f^*_{\text{img}} + \omega_{\text{pc}} \cdot f^*_{\text{pc}} + \omega_{\text{imu}} \cdot f^*_{\text{imu}} \tag{2}$$

The attention mechanism is adopted in cross-modal dependency modeling, and the contribution of different modalities

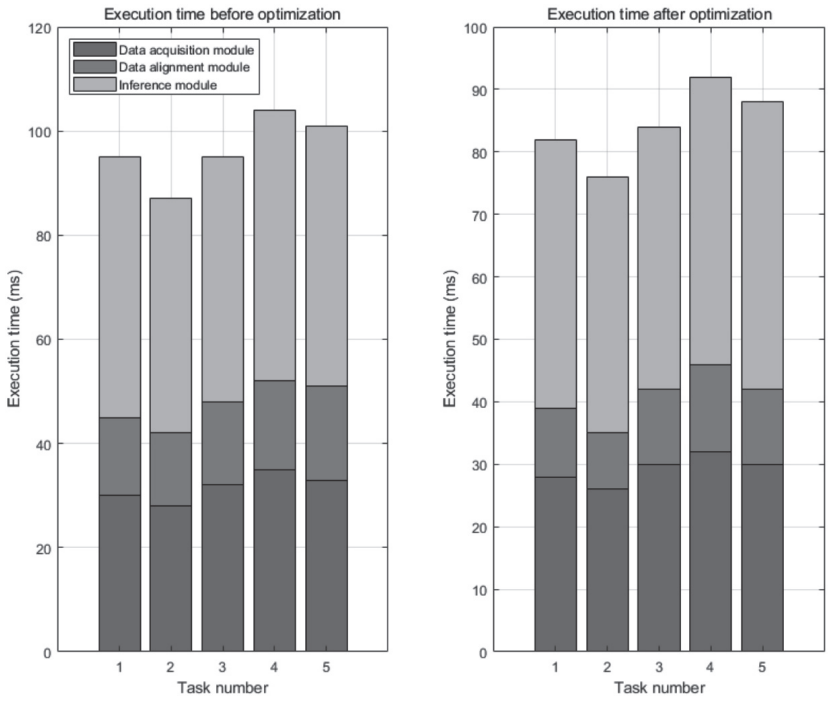

Figure 2 Comparison of execution time with parallel inference optimization.

is dynamically evaluated in a task-driven manner to achieve information reconstruction and difference compensation.

4. LIGHTWEIGHT ARCHITECTURE DESIGN AND EVENT-DRIVEN TASK SCHEDULING

4.1 Lightweight Architecture Design

The overall operation of the robot comprises four independently running functional sub-modules: the data acquisition module, the data alignment module, the inference module, and the navigation module. Each module communicates asynchronously through a unified interface to avoid operation blockage caused by coupling between modules, and supports on-demand parallel scheduling. The data acquisition module is responsible for encapsulating various sensor drivers and receiving camera, laser radar, and IMU data streams in parallel using multi-threading. Each type of data is initially cached and timestamped in an independent thread and communicates with downstream modules through a ring buffer. The data alignment module is responsible for denoising and standardizing the data after it is acquired. Then, according to the unified timestamp under the master clock, a linear interpolation strategy is used to complete the heterogeneous data alignment. The module has a fixed operation cycle and is decoupled from the acquisition frequency to ensure that time synchronization is not affected by frame rate fluctuations. The inference module is responsible for receiving preprocessed data and scheduling parallel inference of three types of neural networks in the GPU environment. Through compute unified device architecture (CUDA) flow control, each type of model is executed in an independent flow to improve hardware resource utilization. Figure 2 presents specific effects.

Figure 2 shows the time difference before and after parallel optimization. The X-axis represents the task cycle, showing the time from task commencement to execution. The Y-axis represents in milliseconds the execution time of each task. The left sub-figure shows the results before optimization, and the execution time of the system is relatively long. Each module needs to be completed in sequence during execution, and there is a strong sequence dependency between tasks, which results in a longer completion time for each task. The execution time of all tasks is greater than or equal to 45 milliseconds. Without parallel processing, the waiting and dependencies between modules significantly extend the overall execution cycle. The right sub-figure shows the optimized results, with shorter execution time. The execution time of the third task is reduced from 47 milliseconds to 42 milliseconds. This is due to the application of parallel inference that allows multiple neural network tasks to be performed simultaneously, thus speeding up the overall inference process. When multiple modules are executed in parallel, the waiting time between tasks is significantly reduced. The optimized execution time shows a more balanced time distribution, and hardware resources are used more efficiently.

The output feature vectors are uniformly entered into the shared representation space and, after being fused by the cross-modal attention mechanism, they are stored in the shared memory area for the navigation module to call. The navigation module is responsible for running perception updates and path replanning at a fixed frame rate based on the fused features. The improved A* algorithm is used to implement incremental path updates, and the deep reinforcement learning strategy network is integrated to adjust the control signal in real time. This module monitors the changes in the front-end perception results through state subscription and triggers the activation of replanning.

4.2 Event-driven Task Scheduling

An event-triggered mechanism is used instead of the continuous operation mode to reduce energy consumption and computing load. The specific strategies are as follows: each type of perception result is bound to a specific change indicator, and the corresponding threshold is set for state change detection. When the structural similarity between two consecutive frames of images is lower than 0.85, it is assumed that the scene has changed significantly; when the mean change rate of the Euclidean distance between point cloud frames is greater than 15%, it is indicative of a change in the obstacle structure; when the angular velocity or linear acceleration exceeds the set threshold (3σ), it is judged as a sudden change in posture. The detection process is completed inside the perception module, which smooths historical features by means of a sliding window, and generates an event signal when the change exceeds a threshold. After receiving the event signal, the controller allocates module execution permissions according to the event type and priority through the task scheduler: high-priority events (such as path blocking) trigger the path module to interrupt the current execution flow and insert emergency replanning; medium-priority events (such as IMU abnormalities) schedule the perception module to reacquire short-term feature sequences; low-priority events (such as slow changes in image scenes) only update the perception cache and do not trigger the inference process. The scheduler maintains the event queue and thread pool resource state internally, dynamically allocates computational resources based on event priority, controls the number of concurrent executions, and avoids competition for resources. To avoid false triggering, a cooling time window is set. Repeated triggering of the same type of event within the window is discarded, and only the first valid event is retained. After the task is completed, the context resources are recycled. GPU memory, thread pool handles, and temporary cache are released to ensure task scheduling stability and resource control. In addition, a module lifecycle monitoring mechanism is added. The execution time limit of each task is monitored by Watchdog. If the timeout is exceeded, it automatically terminates and logs to prevent unresponsive processes from blocking global scheduling.

5. PERCEPTION-DRIVEN PATH PLANNING AND NAVIGATION DECISIONS

5.1 Perception-driven Path Planning

The fusion features output by the perception module are first mapped to an occupancy grid map. Based on this map, a dynamic passable area identification is constructed. The improved A* algorithm is used as the main engine for path planning, and the core optimization strategies include heuristic function adjustment, cost function weighting, and adjacent extension restriction. The heuristic function adjustment is reflected in the Manhattan distance of the classic A*, adding an obstacle density penalty and direction offset factors to make the heuristic function more consistent with the path coherence and steering stability in complex environments. The cost function weighting comprehensively considers three indicators: grid risk weight (degree of obstacle proximity), path curvature change (for smooth control), and platform motion characteristics (minimum turning radius limit). Each item participates in the path cost accumulation through a weighted coefficient to form a non-uniform cost model. The adjacent extension restriction adds dynamic obstacle occlusion detection when the path node is extended. If there is a high-density point cloud in the target direction, the direction is automatically skipped to reduce redundant node expansion and improve planning efficiency. Map updates are maintained in real time using a sliding window. Every time new perception information is received, the local sub-map area is refreshed, and only the local path containing the changed area is reconstructed to avoid the computational overhead caused by global replanning. For real-time calls, the path data is cached in the priority queue by the control strategy module.

5.2 Navigation Decision

In the control stage, the perception-estimation joint state is used as input to make navigation decisions. The core inference module uses a pre-trained deep reinforcement learning network, and the strategy structure is implemented based on the deep deterministic policy gradient (DDPG) algorithm, with the ability to output continuous actions. The state representation is constructed. The current state includes the position coordinates, orientation angle, velocity vector, historical path residual (the deviation trajectory between the current position and the expected path), and dynamic obstacle characteristics in the local perception area. Each state quantity is normalized and concatenated into a unified vector input to the strategy network. Action output structure: the strategy network consists of a two-layer fully connected structure, and the output action is a three-dimensional control instruction: linear speed, angular velocity, and emergency stop weight. The angular velocity part uses tanh compression activation to constrain the steering range and ensure smooth control. The emergency stop weight is normalized by a sigmoid and used to determine control interruption. In the strategy execution mechanism, the control module performs feedforward inference at a frequency of 20Hz. The decision results of each frame are converted into specific motor control commands in combination with the proportional-integral-derivative (PID) adjustment mechanism. In the case of historical residual deviation exceeding the threshold or close intrusion of obstacles, the strategy weight is shifted in the direction of obstacle avoidance or deceleration to achieve adaptive control switching.

Three types of safety constraint mechanisms are deployed to avoid path conflicts and control anomalies. These are 1) the speed limiter: when the perception result shows that the obstacle approach distance is less than the set threshold (0.5m), the linear speed is actively reduced to 0 according to the exponential decay function to prevent collision due to inertia; 2) the heading deviation corrector: when the heading angle deviates from the expected path direction by more than the set threshold (15°), the heading priority control is enabled, and the linear speed is frozen. Only the angular

Figure 3 Safety constraint mechanism triggering.

speed is adjusted until the direction is consistent; and 3) the strategy fuse: when the DDPG output is inconsistent with the historical control instructions for three consecutive frames, or the sudden change amplitude of state input exceeds the learning range (angular velocity suddenly increases by more than 2 rad/s), the strategy network is immediately fused and switched to the static emergency braking module to prevent the strategy network from outputting out-of-control instructions under abnormal conditions. The control process combines the event trigger signal of the perception module to respond to dynamic environmental changes in real time. Path reconstruction and control update are independent threads, and a shared state buffer is used for asynchronous reading and writing to ensure the real-time performance and robustness of the control process. Figure 3 shows the safety constraint mechanism.

Figure 3 shows the triggering state of different safety constraint mechanisms in each cycle. The horizontal axis represents the different time periods of system operation. The left Y-axis represents the value of continuous data, obstacle distance, heading deviation, and sudden change of angular velocity. The right Y-axis uses binary values (0 or 1) to represent the triggering state of the safety mechanism (speed limiter, heading deviation corrector, and strategy fuse), with 1 representing that the mechanism is triggered, and 0 representing that it is not triggered. If the obstacle is too close, the speed limiter is triggered to reduce the robot's speed to avoid collision. In the T3-T6 period, the speed limiter is triggered to reduce the running speed. When the heading deviation is greater than the set threshold, the heading deviation corrector is triggered to adjust the robot's heading angle. In the T5-T7 cycle, the heading deviation corrector is triggered, indicating that the heading deviation is greater than the set threshold. If the control instructions for three consecutive frames are inconsistent with the historical instructions, or the angular velocity surge exceeds the threshold, the strategy fuse is triggered and switches to emergency braking mode.

6. EXPERIMENTAL EVALUATION

The experimental process includes data acquisition, sensor synchronization, path planning and control, and evaluation. The experiment is conducted in a semi-open environment that contains dynamic obstacles (such as pedestrians, other robots, and movable objects) and complex terrain (such as curves, ramps, narrow passages, etc.). All sensor devices, including cameras, laser radar, and IMU, are started to ensure that all sensors are working properly. The robot platform and its control module are initialized, and initial parameters (speed limit, heading control angle, IMU data sampling frequency, etc.) are configured. The data collection frequency and resolution of each sensor are set: the camera resolution is 640×480; the laser radar scanning frequency is 10Hz; the IMU sampling frequency is 200Hz, etc. Additionally, the event trigger threshold of the perception module is configured, including the threshold of the obstacle approach distance and the threshold of the heading deviation angle.

6.1 Control System Performance Evaluation

The purpose of the control error experiment is to determine the deviation between the current position of the robot and the desired target when performing the path tracking task. A fixed target trajectory is set, and the robot's path planning and tracking control modules are enabled. The robot starts to run along the set path. The control system receives sensor data (IMU, odometer, laser radar, etc.) in real time and performs path tracking. Sampling is performed at a fixed point in time, and multiple experiments are performed on the same trajectory.

In Figure 4, the deviation between the target trajectory and the actual trajectory is analyzed through the control errors of six different experiments, and the control precision of the robot in each experiment is further shown through the error vector. The black plus sign represents the target path, and the robot should walk along this trajectory. The target path is made up of points with fixed x and y coordinates. The blue dots represent the actual trajectory. When the robot is executing movements, due to control errors (such as sensor errors, calculation errors, etc.), its movement path deviates from the target trajectory. Each target trajectory point has an error vector that indicates the deviation between the target point and the actual position. The error vector points from the target position to the actual position. The length of the arrow represents the magnitude of the error, and the direction

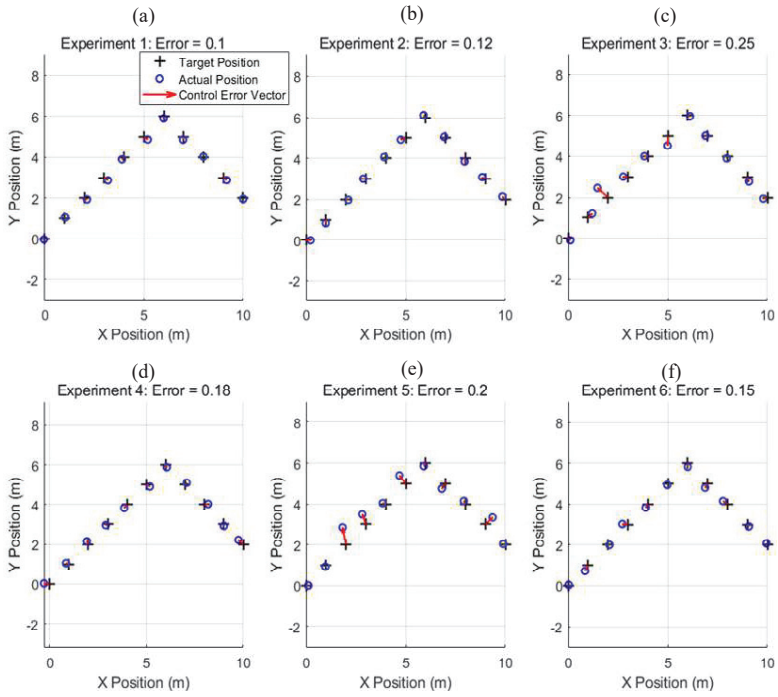

Figure 4 Control errors. Figure 4(a): First experiment with an average error of 0.1. Figure 4(b): Second experiment with an average error of 0.12. Figure 4(c): Third experiment with an average error of 0.25. Figure 4(d): Fourth experiment with an average error of 0.18. Figure 4(e): Fifth experiment with an average error of 0.2. Figure 4(f): Sixth experiment with an average error of 0.15.

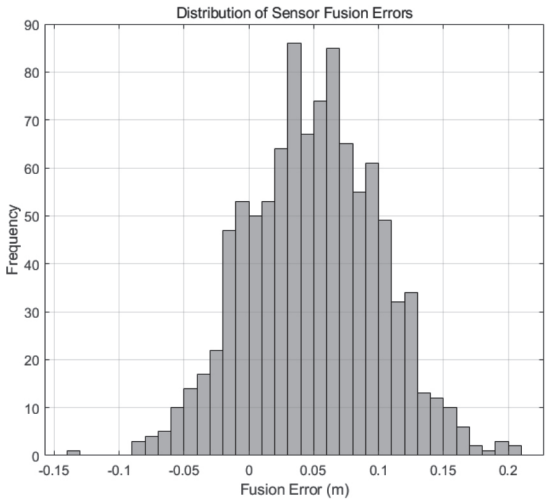

Figure 5 Sensor fusion error distribution.

of the arrow represents the direction of the error. Different error margins reflect the control error in each experiment. A smaller error margin (such as 0.1) means that the actual path of the robot is closer to the target path, and a larger error margin (such as 0.25) means that the robot deviates far from the target path. In this experiment, the error margins are relatively small, with average errors of [0.1, 0.12, 0.25, 0.18, 0.2, 0.15], which are relatively close and none of them exceeds 0.25.

6.2 Data Fusion Error

The data fusion precision of different sensor combinations (such as camera + laser radar, IMU + laser radar, camera + IMU, etc.) is evaluated. The raw data is collected from each sensor (laser radar, camera, IMU, etc.). The collected sensor data is preprocessed to ensure that the timestamps of each sensor data are aligned. According to the fusion algorithm, the data from the camera, IMU, and laser radar are fused to obtain a global positioning estimate or environment map. The fusion error metric is used to evaluate the quality of data synchronization and fusion.

In Figure 5, the X-axis shows the size of the fusion error, which is the difference between the estimated value and the actual value after the data from multiple sensors (camera, IMU, and laser radar) are fused. The error is measured in meters. The Y-axis shows the frequency of each error value interval. Most data points are concentrated within a small range (for example, between 0 and 0.1 meters), indicating that the fusion error of most sensors is small and the fusion effect of the system is relatively stable.

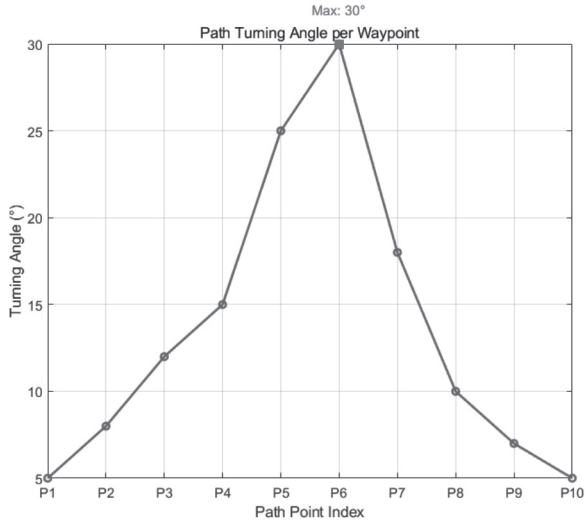

Figure 6 Smoothness of the generated path.

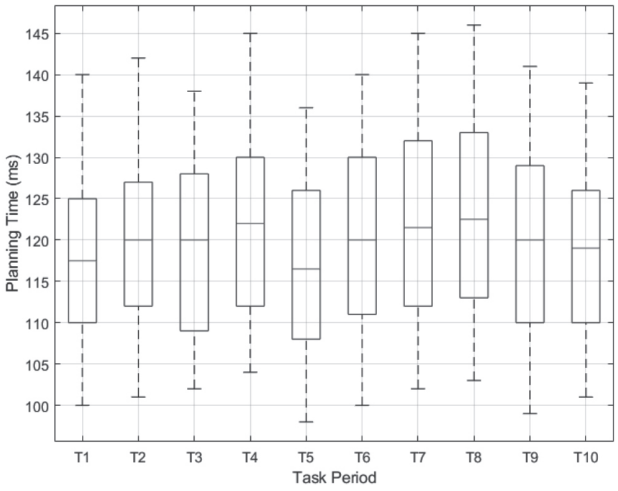

Figure 7 Planning time for different task cycles.

6.3 Path Planning Performance Evaluation

6.3.1 Path Smoothness

The smoothness of the paths generated by different path planning strategies is evaluated to determine whether there are sharp turning angles, sudden changes in curvature, etc. These phenomena may cause unstable robot motion, increased energy consumption, and difficult control. Different starting and end task combinations are set. Point sequences are extracted at equal intervals. The angle formed by every three points is calculated as the turning angle sequence.

In Figure 6, the X-axis indicates the number of 10 consecutive key points in the path. The robot starts from P1 along the path and passes through P2, P3, …, in sequence. The Y-axis indicates the turning angle of the path at the current point. The larger the value, the sharper the turn at the current point. The broken line connects the turning angle values of each point, and its "smoothness" is observed. The red dot marks the maximum turning angle position. At P6, a 30° turning angle appears, showing that the path undergoes a large change in direction in this section, resulting in control difficulties.

6.3.2 Planning Time Assessment

The real-time performance and computational efficiency of path planning in different test scenarios are assessed. The time of each path planning operation is recorded. To ensure the reliability of the experiment, the path planning operation is repeated multiple times during each task cycle, and the time of each calculation is recorded.

Figure 7 shows the distribution of path planning time for different task cycles. The task cycle corresponds to the test scenario. Multiple experiments are conducted on the same task cycle. The X-axis shows different task cycles, and each cycle corresponds to a set of path planning time data. The Y-axis represents the time required for path planning; that is, the time required to calculate the path in each cycle. The maximum planning time is at T8, with a time of 146 milliseconds, and the minimum planning time is at T5, with a time of 98 milliseconds. Although there are various task cycles and test scenarios, the planning time is not much different, wiht the median being between 115 and 125 milliseconds.

Table 2 Safety performance.

Experiment ID	Collisions	Min Distance (m)	Unsafe Distance Count	Avg Response Time (ms)	Speed Limiter Triggers	Heading Deviation Corrector Triggers	Strategy Fuse Triggers	Strategy Triggering Rate (%)
1	0	0.62	3	35	2	1	0	100
2	0	053	5	48	3	2	1	83.3
3	0	0.56	1	30	1	1	0	100
4	0	0.57	4	42	2	3	0	91.6
5	0	0.55	3	37	2	1	0	100
6	0	0.6	1	34	1	0	0	100

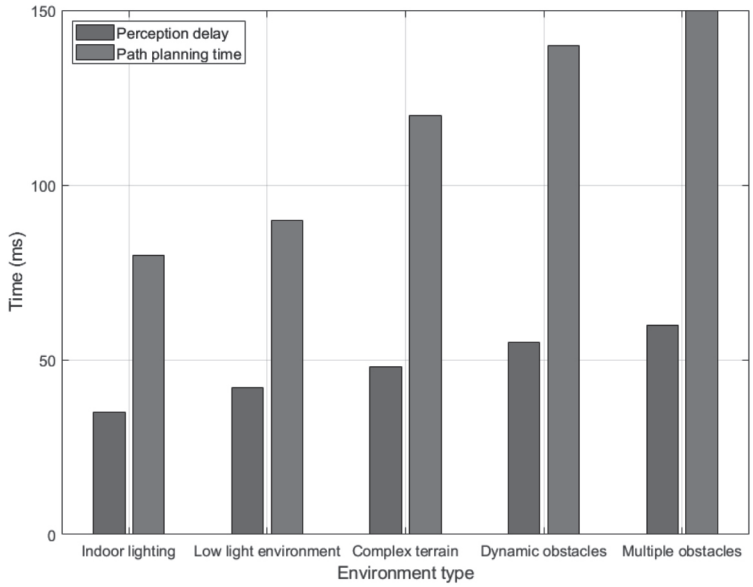

Figure 8 Perception delay and path planning time in different environments.

6.4 Safety Performance

The safety of the robot system is measured by means of a series of key indicators, including the number of collisions, safety distance, safety policy response time, and policy effectiveness. The simulation environment of the robot and obstacles is configured to ensure that the robot can move in the environment and encounter obstacles. The robot's navigation system is set up to ensure that the robot has the ability to avoid obstacles, plan paths, and control. Safety strategies, including speed limiters, heading deviation correctors, etc., are set up to trigger safety mechanisms. Data such as the number of collisions, safety distance, triggering of safety strategies, and response time are recorded. The results of each experiment are organized, and indicators such as average response time and strategy triggering rate are calculated, as shown in Table 2.

In all experiments, the number of collisions is 0, indicating that the obstacle avoidance system can successfully avoid collisions in most cases. The minimum value of the safety distance is greater than 0.5 meters, proving that the robot maintains a good safety distance during driving. The safety strategy response time in each experiment is within a reasonable range, and the strategy triggering rate of most experiments is close to 100%, indicating that the system can trigger the safety strategy in time and effectively execute it.

6.5 Environmental Adaptability

The robot system is started, and the navigation, perception, and path planning modules are loaded. A unified navigation task goal (such as navigating from the starting point to the end point) is configured, and the path length is kept consistent. The experiment is repeated 5 times in each environment (with statistics on average performance and fluctuations). During each run, the delay from the perception module receiving sensor data to outputting obstacle state and the time taken by the planning module from receiving environmental information to completing path generation are recorded.

The X-axis of Figure 8 contains five typical test environments: 1) Indoor lighting: normal bright indoor environment. 2) Low light environment: such as night or insufficient lighting. 3) Complex terrain: such as ramps and uneven surfaces. 4) Dynamic obstacles: such as pedestrians and moving obstacles. 5) Multiple obstacles: such as narrow passages and dense obstacles. The Y-axis represents the time consumption in terms of two types of key performance indicators. The blue column shows the perception delay (the time from sensor input to perception result output), and the orange column shows the path planning time (the time from receiving perception data to generating a path). The perception delay is lowest under indoor lighting conditions

(35ms), indicating that the system runs smoothly under standard conditions. It rises slightly in low light and complex terrain (42ms and 48ms). It is highest in dynamic obstacles and multiple obstacles (55ms and 60ms), probably due to the need to process more targets or more complex point cloud data. The path planning time also increases with the complexity of the environment. The time consumption is the highest when there are dynamic obstacles and multiple obstacles (140ms and 150ms respectively), indicating that the obstacle avoidance strategy and path reconstruction burden are increased. In simpler environments where there is, for example, indoor lighting and low light, it remains at around 80ms–90ms, which meets the real-time performance requirements.

7. CONCLUSION

This paper focuses on the needs of lightweight robots for efficient perception and navigation in complex environments. It adopts a fusion method based on multimodal feature extraction and cross-modal attention mechanism, builds a modular architecture, and applies an event trigger mechanism to achieve efficient preprocessing of multi-sensor data, deep feature fusion, and path planning control. By improving the A* algorithm and linking the DDPG strategy network control, the perception precision, navigation stability, and system real-time performance are significantly improved. Although the method performs well in terms of computational efficiency and perception robustness, it still has problems such as response lag and insufficient strategy generalization ability in extreme dynamic environments. Future work can further expand the application scenarios of the method proposed in this paper. In multi-robot systems, improving the overall performance of the system through the sharing of perception information and collaborative planning, especially the navigation and obstacle avoidance capabilities in complex environments, needs further exploration.

FUNDING

This work was supported by the National Key R&D Program of China (2024YFC3015800); "Wave Future" shortwave infrared photoelectric sensor industry chain collaborative innovation project(2024ZDCX031)

REFERENCES

1. Wu J, Antonova R, Kan A, et al. Tidybot: Personalized robot assistance with large language models. Autonomous Robots, 2023, 47(8): 1087–1102.
2. Raj R, Kos A. A comprehensive study of mobile robot: History, developments, applications, and future research perspectives. Applied Sciences, 2022, 12(14): 6951.
3. Haddadin S, Parusel S, Johannsmeier L, et al. The franka emika robot: A reference platform for robotics research and education. IEEE Robotics & Automation Magazine, 2022, 29(2): 46–64.
4. Chang Y, Ebadi K, Denniston C E, et al. LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments. IEEE Robotics and Automation Letters, 2022, 7(4): 9175–9182.
5. Arents J, Greitans M. Smart industrial robot control trends, challenges and opportunities within manufacturing. Applied Sciences, 2022, 12(2): 937.
6. Zhang C, Chen J, Li J, et al. Large language models for human–robot interaction: A review. Biomimetic Intelligence and Robotics, 2023, 3(4): 100131.
7. Zhang X, Li Y, Wang X, et al. Multi-source interactive stair attention for remote sensing image captioning. Remote Sensing, 2023, 15(3): 579.
8. Bao Y, Huang Z, Wang H, et al. High-resolution quantification of building stock using multi-source remote sensing imagery and deep learning. Journal of Industrial Ecology, 2023, 27(1): 350–361.
9. Xie J, Qi T, Hu W, et al. Retrieval of live fuel moisture content based on multi-source remote sensing data and ensemble deep learning model. Remote Sensing, 2022, 14(17): 4378.
10. Ghorbanian A, Ahmadi S A, Amani M, et al. Application of artificial neural networks for mangrove mapping using multi-temporal and multi-source remote sensing imagery. Water, 2022, 14(2): 244.
11. Yuan Y, Wen Q, Zhao X, et al. Identifying grassland distribution in a mountainous region in Southwest China using multi-source remote sensing images. Remote Sensing, 2022, 14(6): 1472.
12. Ge F, Wang G, He G, et al. A hierarchical information extraction method for large-scale centralized photovoltaic power plants based on multi-source remote sensing images. Remote Sensing, 2022, 14(17): 4211.
13. Stock-Homburg R. Survey of emotions in human–robot interactions: Perspectives from robotic psychology on 20 years of research. International Journal of Social Robotics, 2022, 14(2): 389–411.
14. Inkulu A K, Bahubalendruni M V A R, Dara A, et al. Challenges and opportunities in human robot collaboration context of Industry 4.0-a state of the art review. Industrial Robot: the international journal of robotics research and application, 2022, 49(2): 226–239.
15. Wang J, Chortos A. Control strategies for soft robot systems. Advanced Intelligent Systems, 2022, 4(5): 2100165.
16. Letheren K, Jetten J, Roberts J, et al. Robots should be seen and not heard…sometimes: Anthropomorphism and AI service robot interactions. Psychology & Marketing, 2021, 38(12): 2393–2406.
17. Bhatti M A, Song Z, Bhatti U A, et al. AIoT-driven multi-source sensor emission monitoring and forecasting using multi-source sensor integration with reduced noise series decomposition. Journal of Cloud Computing, 2024, 13(1): 65.
18. Bonci A, Cen Cheng P D, Indri M, et al. Human-robot perception in industrial environments: A survey. Sensors, 2021, 21(5): 1571.
19. Jin Y, Chen C, Zhao S. Multisource data fusion diagnosis method of rolling bearings based on improved multiscale CNN. Journal of Sensors, 2021, 2021(1): 2251530.
20. Zhao J, Zhou Y, Shi B, et al. Multi-stage fusion and multi-source attention network for multi-modal remote sensing image segmentation. ACM Transactions on Intelligent Systems and Technology (TIST), 2021, 12(6): 1–20.
21. Tao X, Zhu B, Xuan S, et al. A multi-sensor fusion positioning strategy for intelligent vehicles using global pose graph optimization. IEEE Transactions on Vehicular Technology, 2021, 71(3): 2614–2627.
22. Yan X, Li J, Smith A R, et al. Evaluation of machine learning methods and multi-source remote sensing data combinations

to construct forest above-ground biomass models. International Journal of Digital Earth, 2023, 16(2): 4471–4491.
23. Zhang P, Gao X, Miao M, et al. Design and control of a lower limb rehabilitation robot based on human motion intention recognition with multi-source sensor information. Machines, 2022, 10(12): 1125.
24. Liu R W, Guo Y, Nie J, et al. Intelligent edge-enabled efficient multi-source data fusion for autonomous surface vehicles in maritime internet of things. IEEE Transactions on Green Communications and Networking, 2022, 6(3): 1574–1587.
25. Zhang S, Liu T, Wang C. Multi-source data fusion method for structural safety assessment of water diversion structures. Journal of Hydroinformatics, 2021, 23(2): 249–266.
26. Dai K, Li Z, Xu Q, et al. Identification and evaluation of the high mountain upper slope potential landslide based on multi-source remote sensing: the Aniangzhai landslide case study. Landslides, 2023, 20(7): 1405–1417.
27. Y. Watanobe; Y. Yaguchi; K. Nakamura; T. Miyaji; R. Yamada; K. Naruse. Architecture and framework for data acquisition in cloud robotics. International Journal of Information Technology, Communications and Convergence, 2021 Vol.4 No.1, pp.1–25.
28. Che C, Liu B, Li S, et al. Deep learning for precise robot position prediction in logistics. Journal of Theory and Practice of Engineering Science, 2023, 3(10): 36–41.
29. Zhu K, Zhang T. Deep reinforcement learning based mobile robot navigation: A review. Tsinghua Science and Technology, 2021, 26(5): 674–691.
30. Cong Y, Gu C, Zhang T, et al. Underwater robot sensing technology: A survey. Fundamental Research, 2021, 1(3): 337–345.
31. Ren Y, Liu X, Zhang B, et al. Sensitivity assessment of land desertification in China based on multi-source remote sensing. Remote Sensing, 2023, 15(10): 2674.
32. Paluch S, Tuzovic S, Holz H F, et al. "My colleague is a robot"–exploring frontline employees' willingness to work with collaborative service robots. Journal of Service Management, 2022, 33(2): 363–388.
33. Chen, X, Zhang, L. Intelligent Monitoring Technology for Traffic Road Construction Quality under Deep Learning. Engineering Intelligent Systems, 2025, 33(1): 77–87.
34. Feng, X. Design of Vehicle Structural Stability and Safety Control System Based on Genetic Algorithm. Engineering Intelligent Systems, 2025, 33(5): 525–534.