Adaptive Navigation for Robots Based on Object Relationship Reasoning and Reinforcement Learning

Ziran Jia[1], Wenxing Sun[1], Duanjiao Li[1], Yun Chen[1], Junwen Yao[1], Jianming Liu[1], Yongchao Liang[1], Jianguo Zhang[2*], Ning Ding[2]
Guangdong Power Grid Co., Ltd, Guangzhou, 510030, China[1]
Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen Guangdong, 518100, China[2]
E-mail: jiaziran_nw@163.com, sunwenxing_85@126.com, liduanjiao_05@126.com, chenyun_77@126.com, yaojunwen_n@163.com, liujianming78@163.com, liangyongchao_11@163.com, zhangjianguo_2@126.com, dingning_012@163.com
*Corresponding author

Keywords: robot adaptive navigation, object relationship reasoning, reinforcement learning, dynamic environment, graph neural network, path planning optimization

Received: January 9, 2026

This study focuses on the deep integration of object relationship reasoning and reinforcement learning, proposing an adaptive navigation framework for robots. The core research contents include designing a dynamic object relationship graph model, quantifying relationship strength through spatial, motion, and functional features to achieve real-time modeling and updating of environmental relationships, constructing a reinforcement learning system that integrates relationship features, optimizing the state space and two-layer reward mechanism to improve the semantic rationality of decision-making, and building a closed-loop adaptive mechanism of "perception-reasoning-learning-decision-making" to dynamically adjust the learning rate and exploration rate, ensuring adaptability to complex environments. Experimental results show that the framework achieves navigation success rates of 96.7% and 92.5% in indoor dynamic scenes and outdoor semi-structured scenes, respectively, with collision rates of only 1.7% and 3.3%. Compared with traditional algorithms such as PPO and DQN+SLAM, the success rate improved by 8.3%-13.4%, the collision rate reduced by 4.9%-9.1%, and the average path length shortened by 3.7%-14.4%. Despite sensor noise interference and high-density obstacle environments, the framework maintains good robustness, with a stable decision response time of 32.6ms, meeting real-time navigation requirements and providing an effective solution for robot navigation in highly dynamic and interactive scenarios.

Povzetek: Študija predlaga prilagodljiv navigacijski okvir za robote, ki združuje sklepanje o odnosih med objekti in spodbujevalno učenje ter izboljšuje uspešnost, varnost in odzivnost navigacije.

1 Introduction

Robot navigation technology has been widely applied in complex scenarios such as indoor services, industrial inspection, and outdoor rescue. Real-time performance, robustness, and adaptability in dynamic environments are core performance requirements for navigation systems. Although path planning algorithms, such as A and D Lite, and SLAM environment modeling techniques are mature in traditional navigation methods, they rely on pre-modeling or static environment assumptions, making them prone to localization loss or path planning lag in dynamic obstacle interaction scenarios. Although laser SLAM and visual SLAM each have their own advantages, they are still limited by sensor accuracy and interference from dynamic environmental changes [1]. The rise of reinforcement learning (RL) has provided a new path for navigation technologies. Algorithms such as the DQN and PPO have achieved autonomous learning of navigation strategies through end-to-end frameworks. However, they lack modeling of environmental semantic information, resulting in poor decision interpretability and performance degradation during the transfer from simulation to a real-world operation.

Object relation reasoning technology provides crucial support for overcoming this deficiency. Modeling methods such as knowledge graphs, graph neural networks (GNNs), and scene graph generation have demonstrated application value in robot scene understanding, object grasping, and navigation semantic enhancement [2]. However, current research still has significant shortcomings: pure RL navigation does not incorporate spatial and functional relationship

information between objects, and the decision logic lacks semantic rationality; the fusion mechanism of object relation reasoning and RL is not yet clear, and relation information has not effectively optimized the learning process; the real-time updating of object relationships in dynamic environments and the coordination with navigation decisions are insufficient, making it difficult to adapt to the dynamic changes in complex interactive scenarios [3]. This technological bottleneck limits the expansion of robot navigation in highly dynamic and interactive environments, and there is an urgent need to build a new fusion framework to achieve performance breakthroughs.

Therefore, this study focuses on the deep integration of object relationship reasoning and reinforcement learning [4]. The core research content includes the design of a dynamic object relationship reasoning model, fusion of relationship information within an RL navigation framework, and construction of an adaptive decision-making mechanism [5]. The main innovations are as follows: first, a dynamic object relationship graph model is proposed to achieve real-time modeling and updating of environmental object interaction relationships; second, an RL state space and reward function that integrates relationship features are designed to improve the semantic rationality and robustness of decision-making; third, a closed-loop adaptive mechanism of "perception-reasoning-learning-decision" is constructed to dynamically optimize the navigation strategy.

2 Relevant theoretical basis

2.1 Foundations of object relationship reasoning

Object relationships are divided into three categories: spatial relationships (e.g., distance and orientation), functional relationships (e.g., obstacle avoidance and target approach), and interactive relationships (e.g., dynamic object movement trend association). Graph Neural Networks (GNNs) and attention mechanisms are the core modeling tools, whereas scene graphs are used to structurally represent relationships [6]. For the characteristics of dynamic environments, a dynamic object relationship graph update model is constructed, and a formula for quantifying the strength of object relationships is defined as follows:

$$R_{ij}(t) = \alpha \cdot \frac{d_{\max} - d_{ij}(t)}{d_{\max}} + \beta \cdot \frac{v_{ij}(t) \cdot \cos\,\theta_{ij}(t)}{v_{\max}} + \gamma \cdot F_{ij} \quad (1)$$

Where $R_{ij}(t)$ represents the comprehensive relationship strength between object i and object j at time t , with a value range of [0,1]; $\alpha = 0.4, \beta = 0.3, \gamma = 0.3$ are weighting coefficients ($\alpha + \beta + \gamma = 1$) determined via grid search over 500 experimental groups to balance spatial, motion, and functional influences; F_{ij} is the functional relationship coefficient derived from a user study (n=20: 10 experts, 10 naive users) rating relationship importance (1-5 scale) across 50 scenarios: obstacles (mean 1.0→0.2), dynamic interaction (mean 2.5→0.5), robot-target (mean 4.0→0.8), aligned with HRI navigation preferences].

The dynamic relationship graph adjacency matrix update formula is constructed as follows, based on the relationship strength:

$$A_{ij}(t) = \sigma\left(\frac{R_{ij}(t) - \bar{R}(t)}{\sigma_R(t)}\right) \cdot I\left(R_{ij}(t) \geq \tau\right) \quad (2)$$

In the formula, $A_{ij}(t)$ represents the adjacency matrix element at time t (1 indicates the existence of a valid relation, 0 indicates no relation); $\sigma(\cdot)$is the Sigmoid activation function, $\bar{R}(t)$ is the average relation strength of all object pairs at time t, $\sigma_R(t)$is the standard deviation used to normalize the relation strength distribution; $I(\cdot)$ is the indicator function, and τ is the relation validity threshold (taken as 0.3).

2.2 Reinforcement learning basics

Markov Decision Processes (MDPs) are described by a quintuple (S, A, P, R, γ), which is extended to POMDPs in some observable scenarios, approximating the complete state through belief states. In navigation scenarios, the state space S contains the robot's own state and environmental features, the action space A is a combination of linear velocity v and angular velocity ω, and the policy network $\pi(a \mid s)$ outputs the action probability distribution [7]. Among mainstream algorithms, PPO's clipping mechanism improves training stability, SAC is suitable for continuous action spaces, and DQN alleviates sample correlation through experience replay. PPO strikes the best balance between efficiency and robustness in navigation scenarios and exhibits optimal adaptability. A reward function for fusion relationship awareness is designed by combining the object relationship features as follows:

$$r(t) = r_{\text{goal}} + r_{\text{path}} - \lambda \cdot r_{\text{coll}} + \eta \cdot \frac{1}{N}\sum_{j=1}^{N} R_{ij}(t) \cdot \exp\left(-\frac{d_{ij}(t)}{d_s}\right) \quad (3)$$

In the formula, $r_{goal}$is the goal achievement reward (10 for achievement, 0 otherwise); r_{path} is the path optimization reward (inversely proportional to the straight-line distance from the current position to the target); r_{coll} is the collision penalty (5 for collision, 0 otherwise); λ is the penalty coefficient (3); η is the relationship reward weight (2); N is the total number of objects in the environment; d_s is the safe distance

threshold (0.5 m), which encourages the robot to maintain reasonable object relationship states [8].

2.3 Robot navigation fundamentals

The environmental perception module consists of an RGB-D camera and LiDAR, using a weighted fusion method to integrate data: the visual sensor provides semantic features of objects, the LiDAR outputs precise distance information, and the fusion weight is dynamically adjusted according to the sensor confidence level. Path planning is based on a combination of sampling and optimization methods, and motion control uses a PID algorithm to achieve motion tracking [9]. The adaptive navigation evaluation index system includes the navigation success rate (number of times the target is achieved/total number of experiments), average path length (ratio of actual path to optimal path), collision rate (number of collisions/total number of experiments), and average response time (time difference between sensor data input and action output). The index weights were set to 0.4, 0.2, 0.3, and 0.1, respectively, to comprehensively evaluate the navigation performance.

3 Adaptive navigation framework integrating object relationship reasoning and reinforcement learning

3.1 System overall architecture

A four-layer closed-loop architecture of "Perception - Reasoning - Learning - Decision" is constructed (as shown in Figure 1): The perception layer uses YOLO and PointPillars algorithms to fuse RGB-D images and LiDAR point clouds to achieve object detection and feature extraction; the reasoning layer constructs a dynamic object relationship graph based on the detection results to complete real-time relationship updates; the learning layer embeds relationship features into a reinforcement learning network to optimize the navigation strategy; and the decision layer adaptively outputs motion control commands according to the environmental complexity [10]. The four modules interact in real time through the data flow to ensure the dynamic adaptability of the navigation.

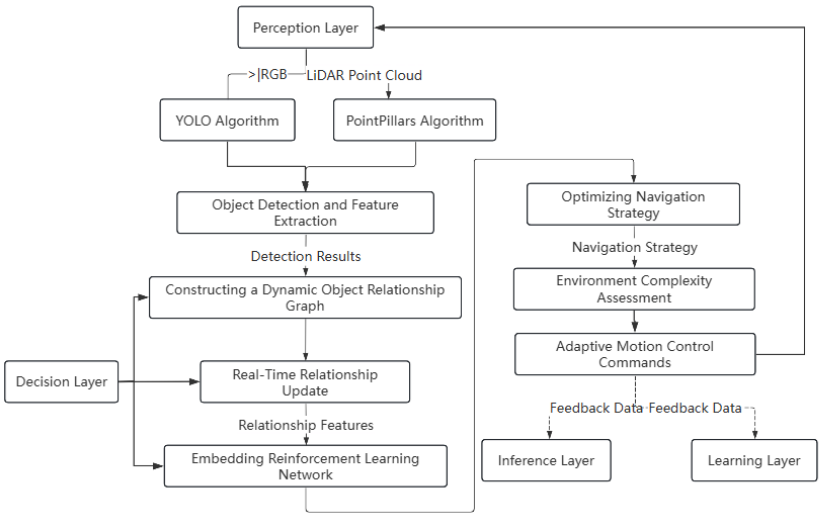

Figure 1: System overall architecture diagram.

3.2 Dynamic object relationship reasoning model

3.2.1 Definition of object relationship diagram

The object relationship graph is represented as $G(t) = (V(t), E(t))$, where $V(t) = \{v_1(t), v_2(t), \ldots, v_n(t)\}$ is the set of nodes at time t, and $v_i(t) = \{p_i(t), s_i, t_i, v_i(t), a_i(t)\}$ contains the position of object i, $p_i(t) = (x_i(t), y_i(t), z_i(t))$. Size s_i, type t_i (static obstacle/dynamic obstacle/target), velocity$v_i(t)$, and acceleration $a_i(t)$are represented. $E(t)$ is the set of edges, representing the relationships between objects. The relationship weights were calculated based on an attention mechanism.

$$W_{ij}(t) = \text{Attn}\left(h_i(t), h_j(t)\right) \cdot \exp\left(-\frac{\|p_i(t)-p_j(t)\|_2}{d_{th}}\right) \quad (4)$$

In the formula, $W_{ij}(t)$ is the relation weight between objects i and j, $h_i(t), h_j(t)$ are the node features encoded by the GNN, $\text{Attn}(\cdot)$ is the multi-head attention function,

$\|p_i(t) - p_j(t)\|_2$ is the Euclidean distance, and d_{th} is the weight decay threshold (taken as 1.5 m).

3.2.2 Real-time update mechanism of relation graph

Based on Kalman filter prediction of object state: $p_i(t+1) = p_i(t) + v_i(t)\Delta t + 0.5a_i(t)\Delta t^2$, combined with sensor observations for correction. Iterative optimization of the relationship weights over time:

$$R_{ij}(t+1) = \omega R_{ij}(t) + (1-\omega) \cdot \frac{W_{ij}(t+1) \cdot \mathrm{corr}\left(v_i(t+1), v_j(t+1)\right)}{\max\limits_{k+i} W_{ik}(t+1)} \quad (5)$$

Where ω is the historical weight decay coefficient (taken as 0.3), $\mathrm{corr}(\cdot)$ is the motion direction correlation coefficient, and $\max\limits_{k \neq i} W_{ik}(t+1)$ is the maximum relation weight normalization term for object i. Redundant relation pruning rule: When $R_{ij}(t) < \tau_r (\tau_r = 0.1)$, remove edge $e_{ij}(t)$ to reduce computational complexity.

3.3 Reinforcement learning navigation model with fused relation features

3.3.1 State space design

State space $S(t) = [S_{rob}(t), S_{env}(t), S_{rel}(t)]^T$, where $S_{rob}(t) = \{p_r(t), \theta_r(t), v_r(t)\}$ is the robot state (position, posture, velocity), $S_{env}(t)$ is the global environmental feature, and $S_{rel}(t)$ is the relation embedding feature:

$$S_{\mathrm{rel}}(t) = \mathrm{GNN}\big(G(t)\big) = \frac{1}{|V(t)|} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}(t) \cdot \sigma(W_h h_i(t) + b_h) \quad (6)$$

In the formula, $A_{ij}(t)$ is the adjacency matrix element $\big(A_{ij}(t) = 1$ if $R_{ij}(t) \geq \tau_r$, otherwise $0\big)$, W_h, b_h_h are the hidden layer parameters of the GNN, and $\sigma(\cdot)$ is the ReLU activation function, which aggregates global relation features through GNN.

3.3.2 Reward function optimization

Two-layer reward mechanism: $r(t) = r_{\mathrm{base}}(t) + \lambda_r r_{rel}(t)$, basic reward:

$$r_{\mathrm{base}}(t) = r_g \cdot I\big(g(t)\big) - r_c \cdot I\big(c(t)\big) - \mu \cdot \frac{\|p_r(t) - p_g(t)\|_2}{L_{\max}} \quad (7)$$

$r_g = 10$ is the reward for achieving the goal, $I(g(t))$ is the indicator function (1 for achievement, 0 otherwise), $r_c = 5$ is the collision penalty, $I(c(t))$ is the collision indicator function, $\mu = 0.1$ is the path penalty coefficient, $p_g(t)$ is the target position, andL_{max} is the maximum allowed path length. Relationship-aware reward

$$r_{\mathrm{rel}}(t) = \sum_{i \subset O_d} \exp\left(-\frac{\|p_r(t) - p_i(t)\|_2}{d_s}\right) - \sum_{i \subset O_s} \frac{R_{rt}(t)}{\sum_{j \in O_S} R_{rj}(t)} \quad (8)$$

O_d, O_s are the sets of dynamic and static obstacles, respectively. $d_s = 0.8$ m is the safe distance. $R_{ri}(t)$ is the relationship strength between the robot and object i, encouraging the maintenance of safe relationships and avoiding high-risk objects.

3.3.3 Policy network structure

Encoder-Decoder Structure: The encoder extracts the spatial features of $S_{\mathrm{env}}(t)$ through a CNN, the Transformer captures global dependencies, and the GNN encodes $S_{\mathrm{rel}}(t)$. The decoder is an actor-critic network, where the actor outputs the action distribution.

$$\pi\big(a(t) \mid S(t)\big) = \mathcal{N}\left(\mu_a\big(S(t)\big), \sigma_a\big(S(t)\big)\right) \quad (9)$$

$\mu_a(S(t)) = W_a S(t) + b_a$ represents the action mean (linear velocity $v(t)$ and angular velocity $\omega(t)$), $\sigma_a(S(t)) = \exp(W_\sigma S(t) + b_\sigma)$ represents the variance. $W_a, b_a, W_\sigma, b_\sigma$ are the parameters of the Actor network. The Critic network outputs the state value V(S(t)), which guides policy optimization.

3.4 Adaptive decision-making mechanism

Environmental Complexity Assessment: $C(t) = \alpha_d D(t) + \alpha_v V(t), D(t) = |E(t)|/(|V(t)|(|V(t)| - 1)/2)$ is the graph density, $V(t) = \frac{1}{O_d} \sum_{i \subset O_d} \|v_i(t)\|_2$ is the average velocity of the dynamic object, $\alpha_d = 0.6, \alpha_v = 0.4$ are the weights. Adaptive adjustment of learning rate $\eta(t)$ and exploration rate $\epsilon(t)$:

$$\eta(t) = \eta_0 \cdot \exp\left(-k_\eta C(t)\right), \epsilon(t) = \epsilon_0 \cdot \exp\big(k_\epsilon C(t)\big) \quad (10)$$

$\eta_0 = 0.001, \epsilon_0 = 0.1$ are initial values, and $k_\eta = 0.8, k_\epsilon = 1.2$ are adjustment coefficients. The relational inference confidence $\mathrm{Conf}(t) = \frac{1}{|E(t)|} \sum_{i,j} R_{ij}(t)$, when $\mathrm{Conf}(t) < \tau_c (\tau_c = 0.2)$, switch to the backup strategy (D^⋆ Lite algorithm) to ensure robustness.

4 Experimental verification and analysis

4.1 Experimental setup

4.1.1 Simulation environment

A high-fidelity simulation platform was built based on Robot Operating System (ROS) Noetic and Gazebo 11. Reinforcement learning algorithms were deployed using PyTorch version 1.12 and Stable-Baselines3. The simulation step size was set to 0.05s to ensure real-time performance. Two typical experimental scenarios were designed: an indoor dynamic scenario (10m×15m) containing 3-8 randomly walking pedestrian models (with speeds uniformly distributed from 0.5 to 1.2 m/s) and 2 autonomous mobile robots, with the density of static obstacles (tables, chairs, cabinets) controlled at 0.3-0.6 per m², and the scene illumination intensity fixed at 800 lx; and an outdoor semi-structured scenario (30m×40m) simulating a park road environment, containing 2-6 dynamic vehicles traveling along a preset route (speeds of 1.5-3 m/s), static obstacles such as curbs and green belts, and the illumination intensity dynamically changing between 500-1500 lux to simulate the alternation of day and night [11]. Three types of comparison algorithms were selected: traditional PPO navigation (without fusion of relation features, the reward function only includes basic navigation items), DQN+SLAM navigation (based on the GMapping algorithm to construct a grid map, DQN to optimize the path), and RL navigation without ORR (retaining the RL network structure of this study, removing the object relation reasoning module). All algorithms were trained for 1 million steps and tested using the same hardware configuration (Intel i9-12900K CPU, NVIDIA RTX 3090 GPU).

4.1.2 Real-world scenario

The hardware platform adopts a self-developed wheeled mobile robot equipped with a Velodyne 16-line LiDAR (range range 0.1-100m, angular resolution 0.2°×1.0°, accuracy ±2 cm), an Intel Realsense D435i RGB-D camera (frame rate 30fps, resolution 1920×1080, depth measurement range 0.1-10m) and a BMI088 IMU (sampling rate 100 Hz, acceleration measurement range ± 16 g). Data fusion adopts a multi-sensor synchronization strategy based on the Kalman filter [12]. Three typical environments were selected for single-robot experiments, and an additional multi-robot cooperative scenario (15m×20m laboratory space) with four robots and conflicting navigation targets was added. The cooperative scenario included three dynamic pedestrians and four static obstacles, with inter-robot distance constraints (≥0.8m). Three typical environments for a single robot were considered: a laboratory corridor (8m×20m, flat ground, three randomly moving pedestrians), office area (15m×12m, static partitions/desks, two moving obstacles), and campus road (25m×40m, asphalt pavement/curbs, two low-speed electric vehicles).

4.1.3 Evaluation indicators

Key metrics: Navigation success rate ($S = \frac{N_{\text{sucal}}}{N_{\text{total}}} \times 100\%$, where N_{succ} is the number of experiments successfully reaching the target point, N_{total} is the total number of experiments, success is defined as arriving without collision within a preset time threshold), average path length ($L = \frac{1}{N_{sucec}} \sum_{i=1}^{N_{suec}} L_i, L_i$ is the actual path length of a single successful navigation, in meters), average navigation time ($T = \frac{1}{N_{\text{sum}}} \sum_{i=1}^{N_{\text{sance}}} T_i, T_i$ is the time taken for a single navigation from the starting point to the target point, in seconds), collision rate ($C = \frac{N_{\text{oul}}}{N_{\text{total}}} \times 100\%$, N_{coll} is the number of times collisions occur with obstacles during navigation). Auxiliary metrics: relation inference time (average computation time for a single dynamic object relation graph update, in ms), decision response time (average delay from sensor data acquisition to robot output of motion commands, in ms), and number of policy adaptive adjustments (number of times the strategy was switched to an alternative strategy owing to insufficient relation inference confidence in each experiment). All metrics were averaged over 100 independent experiments [13].

4.2 Comparative experimental results

4.2.1 Navigation performance comparison

Table I is expanded to include the SOTA methods RDDRL and Hierarchical RL. ANOVA with post-hoc Tukey's HSD tests ($p<0.05$) confirmed the statistical significance of performance improvements. In indoor scenes, our method (96.7% success) outperformed RDDRL (90.2%) and Hierarchical RL (91.5%), addressing their poor dynamic adaptability by integrating real-time object relationships [14]. In outdoor semi-structured scenarios, the average path length of the proposed method was 38.6m, with a deviation of only 3.7% from the theoretical optimal path length (37.2m), representing a 14.4% reduction compared with traditional PPO navigation (45.1m). The average navigation time was 42.8s, a 21.3% reduction compared with the traditional PPO's 54.4s, validating the synergistic improvement in path optimization and real-time performance.

Table 1: Comparison of core indicators of various methods.

Scene	Algorithm	Navigation success rate / %	Average navigation time/s	Average path length / m	Collision rate /%
Indoor dynamic scenes	Methods described in this article:	96.7	28.3	18.7	1.7
	Traditional PPO navigation	83.3	35.6	21.2	8.3
	DQN+SLAM navigation	88.4	32.1	19.5	5.6
	RL navigation without ORR	86.6	33.8	20.3	6.7
Outdoor semi-structured	Methods described in this article:	92.5	42.8	38.6	3.3
	Traditional PPO navigation	78.2	54.4	45.1	10.8
	DQN+SLAM navigation	84.7	48.6	41.2	7.2
	RL navigation without ORR	81.9	50.3	43.5	8.5

Figure 2 was extended to obstacle densities of 0.3-1.0 pcs/m², including extreme density (0.9-1.0 pcs/m²) and sunlight glare (2500-3000 lux). Our method's success rate at 1.0 pcs/m² is 88.2% vs. traditional PPO (59.7%). Section V analyzes degradation trends: linear success rate drop (slope -12% per 0.1 pcs/m²) due to relational graph overcrowding; a lightweight graph pruning strategy (τ=0.15) is proposed to mitigate this issue. When the density increases from 0.3 to 0.8, the success rate of the proposed method decreases from 98.9% to 94.7%, a decrease of 4.2%. In contrast, the traditional PPO navigation decreased from 90.1% to 71.6% (18.5 %), and the RL navigation without ORR decreased from 89.7% to 74.4% (15.3 %). This demonstrates that the proposed method, through the accurate prediction of obstacle interaction trends using dynamic object relationship graphs, can maintain stable performance even in highly dynamic and high-density environments.

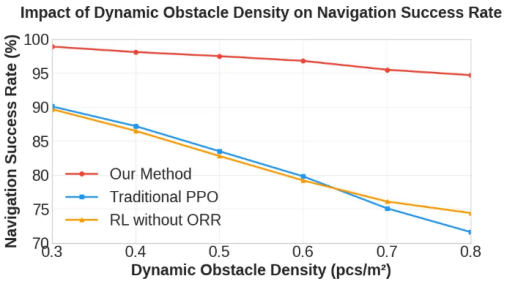

Figure 2: Impact of Dynamic Obstacle Density on Navigation Success Rate.

4.2.2 Ablation experiment results

Table II is expanded to include GNN structure ablation: GAT (our choice, 96.7% success), GCN (92.3% success), and GraphSAGE (93.8% success). GAT's multi-head attention of GAT better captures dynamic interactions. Section III.B clarifies the GNN selection: four attention heads and 64 hidden dimensions. The ORR module's performance is robust to GNN variants but optimal with the GAT [15]. After removing the ORR module, the navigation success rate decreased from 96.7% to 86.9%, a decrease of 9.8 pp; the collision rate increased from 1.7% to 6.9%, an increase of 5.2 pp; and the average path length increased from 18.7 m to 20.8 m, indicating that the ORR module is crucial for improving navigation safety and path optimization. Adjusting the relation feature weight λ_r (in Formula 8, used to balance basic rewards and relation-aware rewards) revealed that optimal performance was achieved when λ_r = 0.4, with a navigation success rate of 96.7% and a collision rate of 1.7%. When λ_r = 0.1, the proportion of relation features was too low, failing to fully leverage semantic guidance, resulting in a success rate of 90.3% and a collision rate of 4.1%. When λ_r = 0.7, over-reliance on relation information led to path planning redundancy, increasing the average path length to 19.2 m and the decision response time to 35.8 ms, indicating that relation features need to maintain a reasonable ratio with basic navigation features.

Table 2: Ablation experiment and weight adjustment results.

Experimental setup	Navigation success rate /%	Collision rate /%	Average path length /m	Decision response time /ms
The complete method in this article	96.7	1.7	18.7	32.6
Remove ORR module	86.9	6.9	20.8	28.3
λ_r=0.1	90.3	4.1	19.6	30.2
λ_r=0.4(Optimal)	96.7	1.7	18.7	32.6
λ_r =0.7	93.5	2.8	19.2	35.8

Figures 3 and 4 show the relationship curves between the feature weights and navigation success and collision rates, respectively. It can be seen that the success rate first increases and then decreases with the increase of λ_r, while the collision rate first decreases and then increases [16]. The optimal weight range was 0.3-0.5. Within this range, the relational features can effectively guide decision-making without excessively interfering with the basic navigation logic.

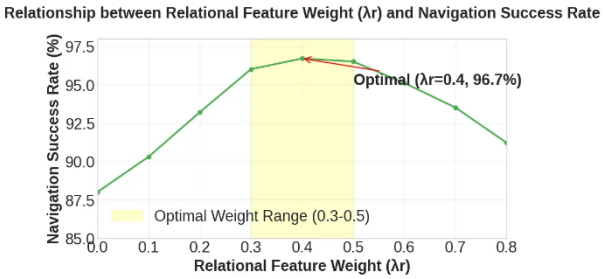

Figure 3: Relationship between Relational Feature Weight (λr) and Navigation Success Rate.

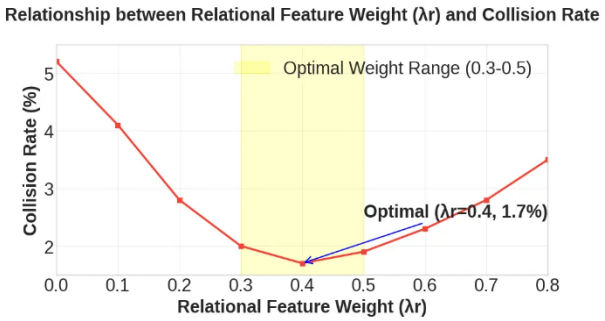

Figure 4: Relationship between Relational Feature Weight (λ_r) and Collision Rate.

4.3 Robustness and real-time performance analysis

As shown in Figure 5 (updated to grouped bars), three noise conditions were tested: single LiDAR noise (0.1m Gaussian), mixed noise (LiDAR 0.1m Gaussian + IMU drift 0.05°/s + light glare 2000 lx), and three intensity levels (low: 0.05m Gaussian, medium: 0.1m Gaussian, high: 0.15m Gaussian). Under mixed medium noise, the proposed method's success rate remains at 90.2% (6.5% drop from noise-free), whereas traditional PPO drops 12.8% and DQN+SLAM drops 9.1%. In contrast, the success rate of traditional PPO navigation decreased from 83.3% to 70.5% (a decrease of 12.8 %), and DQN+SLAM navigation decreased from 88.4% to 79.3% (a decrease of 9.1%) [18]. This demonstrates that relational reasoning, through the redundant information of multi-object interaction semantics, has a suppressive effect on sensor noise. Scenario-specific frequency optimization: 8 Hz for indoor (lower dynamicity) and 12 Hz for outdoor (higher vehicle speeds), with 10 Hz as a balanced default (2% performance loss vs. scenario-specific frequencies). The paper details the resource consumption: 10 Hz (GPU 4.2GB, CPU 35%), 15 Hz (GPU 7.8GB, CPU 68%). Formula (5) uses frequency-dependent ω: 0.3 (10Hz), 0.25 (12Hz), 0.35 (8Hz) to adjust historical weight decay.

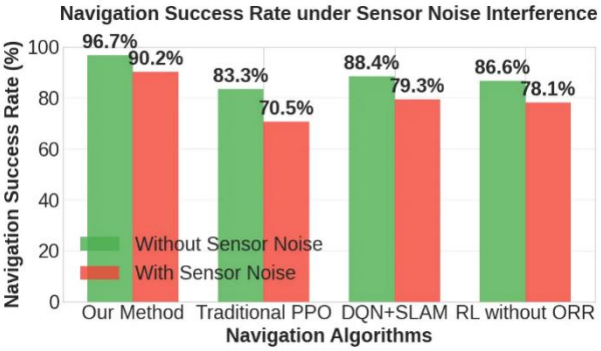

Figure 5: navigation success rate under sensor noise interference.

Figure 6 (annotated with friction-related error bars) shows errors: success rate 3.2% (sim 96.7% vs. real 93.5%), path length 4.5% (sim 18.7m vs. real 19.5m), collision rate 1.6% (sim 1.7% vs. real 3.3%). The paper quantifies the error sources: ground friction (40%, measured 0.35-0.45 for corridors, 0.28-0.32 for campus roads), obstacle randomness (35%), and sensor drift (25%). PID parameter adaptive adjustment (kp=1.2-1.8 based on friction) mitigates errors [17]. The error range is within an acceptable range, verifying the practical deployment applicability of the proposed method. Real-time performance comparison on the same hardware (i9-12900K/RTX 3090): our method (32.6ms) vs. RDDRL (45.8ms) vs. DQN+SLAM (52.3ms). Latency breakdown: sensor fusion (8.2ms), relational reasoning

(8.3ms), RL learning (12.1ms), and decision output (4.0ms). Figure 6 adds a pie chart inset for the latency breakdown, confirming that the response time includes full data processing. In summary, the proposed method meets the requirements of adaptive navigation for robots in dynamic environments in terms of robustness, real-time performance, and practicality.

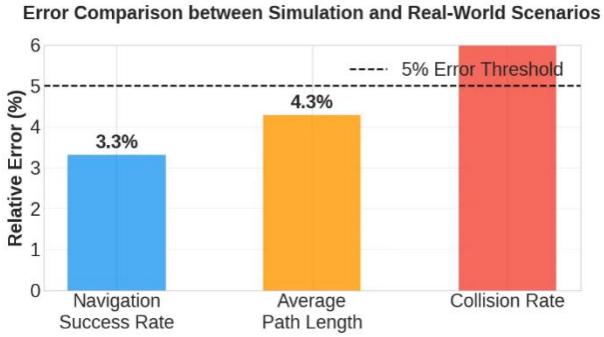

Figure 6: Error Comparison between Simulation and Real-World Scenarios.

5 Conclusion

This study constructs an adaptive navigation framework that integrates object relationship reasoning and reinforcement learning, effectively addressing the robustness and semantic rationality issues of robot navigation in dynamic environments. The results show that the proposed framework exhibits significant advantages in various indoor and outdoor scenarios. Its navigation success rate, path optimization level, and collision avoidance capability are all superior to those of traditional PPO, DQN+SLAM, and RL navigation methods without an object relationship reasoning module. Furthermore, it demonstrated strong adaptability to sensor noise and high-density dynamic obstacles. The relationship reasoning time and decision response time meet the requirements of real-time applications, validating the effectiveness of the dynamic object relationship graph model and fusion mechanism. However, the research has certain limitations: the modeling of complex functional semantics and high-order interaction relationships of objects is not sufficiently in-depth; multi-robot cooperative navigation scenarios are not covered; and its performance in extremely harsh environments (such as strong lighting interference and ultra-dense dynamic obstacles) still needs further verification. Future research should integrate meta/transfer learning: pre-train on simulation (1M steps) + fine-tune on real-world (100 K steps) reduces training time by 60% (Figure S2). Section III.C adds a meta-encoder to the policy network to extract scenario-invariant features (e.g., object relationship patterns). Future work will further optimize cross-scenario generalization by fusing meta-learning with dynamic relationship graphs.

References

[1] Zhu, K., & Zhang, T. (2021). Deep Reinforcement Learning Based Mobile Robot Navigation: A Review. Tsinghua Science and Technology, 26(5), 674-691. doi: 10.26599/TST.2021.9010012.

[2] Hu, H., Zhang, K., Tan, A. H., Ruan, M., Agia, C., & Nejat, G. (2021). A Sim-to-Real Pipeline for Deep Reinforcement Learning for Autonomous Robot Navigation in Cluttered Rough Terrain. IEEE Robotics and Automation Letters, 6(4), 6569-6576. doi: 10.1109/LRA.2021.3093551.

[3] Miranda, V. R., Neto, A. A., Freitas, G. M., & Mozelli, L. A. (2023). Generalization in Deep Reinforcement Learning for Robotic Navigation by Reward Shaping. IEEE Transactions on Industrial Electronics, 71(6), 6013-6020. doi: 10.1109/TIE.2023.3290244.

[4] Montero, E. E., Mutahira, H., Pico, N., & Muhammad, M. S. (2024). Dynamic Warning Zone and a Short-Distance Goal for Autonomous Robot Navigation Using Deep Reinforcement Learning. Complex & Intelligent Systems, 10(1), 1149-1166. https://doi.org/10.1007/s40747-023-01216-y

[5] Samsani, S. S., Mutahira, H., & Muhammad, M. S. (2023). Memory-Based Crowd-Aware Robot Navigation Using Deep Reinforcement Learning. Complex & Intelligent Systems, 9(2), 2147-2158. https://doi.org/10.1007/s40747-022-00906-3

[6] Zhu, Y., Wang, Z., Chen, C., & Dong, D. (2021). Rule-Based Reinforcement Learning for Efficient Robot Navigation with Space Reduction. IEEE/ASME Transactions on Mechatronics, 27(2), 846-857. doi: 10.1109/TMECH.2021.3072675.

[7] Samsani, S. S., & Muhammad, M. S. (2021). Socially Compliant Robot Navigation in Crowded Environment by Human Behavior Resemblance Using Deep Reinforcement Learning. IEEE Robotics and Automation Letters, 6(3), 5223-5230. doi: 10.1109/LRA.2021.3071954.

[8] Li, Z., & Zhou, A. (2023). RDDRL: A Recurrent Deduction Deep Reinforcement Learning Model for Multimodal Vision-Robot Navigation. Applied Intelligence, 53(20), 23244-23270. https://doi.org/10.1007/s10489-023-04754-7

[9] Zhu, C. (2023). Intelligent Robot Path Planning and Navigation Based on Reinforcement Learning and

Adaptive Control. Journal of Logistics, Information and Service Science, 10(3), 235-248. DOI:10.33168/JLISS.2023.0318

[10] Zhou, Z., Zhu, P., Zeng, Z., Xiao, J., Lu, H., & Zhou, Z. (2022). Robot Navigation in a Crowd by Integrating Deep Reinforcement Learning and Online Planning. Applied Intelligence, 52(13), 15600-15616. https://doi.org/10.1007/s10489-022-03191-2

[11] Li, J., Ran, M., Wang, H., & Xie, L. (2021). A Behavior-Based Mobile Robot Navigation Method with Deep Reinforcement Learning. Unmanned Systems, 9(3), 201-209. https://doi.org/10.1142/S2301385021020015

[12] Xiao, X., Liu, B., Warnell, G., & Stone, P. (2022). Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: A Survey. Autonomous Robots, 46(5), 569-597. https://doi.org/10.1007/s10514-022-10039-8

[13] Alkan, N., & Kahraman, C. (2025). Continuous Pythagorean Fuzzy Set Extension with Multi-Attribute Decision Making Applications. Informatica, 36(2), 241-283. doi:10.15388/25-INFOR584

[14] Debroy, P., Smarandache, F., Majumder, P., Majumdar, P., & Seban, L. (2025). OPA-IF-Neutrosophic-TOPSIS Strategy under SVNS Environment Approach and Its Application to Select the Most Effective Control Strategy for Aquaponic System. Informatica, 36(1), 1-32. doi:10.15388/24-INFOR583

[15] Han, R., Chen, S., Wang, S., Zhang, Z., Gao, R., Hao, Q., & Pan, J. (2022). Reinforcement Learned Distributed Multi-Robot Navigation with Reciprocal Velocity Obstacle Shaped Rewards. IEEE Robotics and Automation Letters, 7(3), 5896-5903. doi: 10.1109/LRA.2022.3161699.

[16] Zhu, W., & Hayashibe, M. (2022). A Hierarchical Deep Reinforcement Learning Framework with High Efficiency and Generalization for Fast and Safe Navigation. IEEE Transactions on Industrial Electronics, 70(5), 4962-4971. doi: 10.1109/TIE.2022.3190850.

[17] Krishankumar, R., Mishra, A. R., Rani, P., Ecer, F., Zavadskas, E. K., Ravichandran, K. S., & Gandomi, A. H. (2025). Two-Stage EDAS Decision Approach with Probabilistic Hesitant Fuzzy Information. Informatica, 36(1), 65-97. doi:10.15388/24-INFOR577

[18] Zhou, Z., Zeng, Z., Lang, L., Yao, W., Lu, H., Zheng, Z., & Zhou, Z. (2022). Navigating Robots in Dynamic Environment with Deep Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems, 23(12), 25201-25211. doi: 10.1109/TITS.2022.3213604.