Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

UDAQ: Unified Dynamic and Adaptive Q-iteration for Unknown Environment Path Planning and Exploration

Nasr Abdalmanan
Universiti Malaysia Perlis
Kamarulzaman Kamarudin
kamarulzaman@unimap.edu.my

Universiti Malaysia Perlis
Muqri Zinal
Universiti Malaysia Perlis
Mohd Rizal Manan
Universiti Malaysia Perlis
Victor Bennetts
Örebro University

Research Article

Keywords: Dynamic programming, Path planning, Autonomous exploration, Mobile robots

Posted Date: July 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9974234/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

UDAQ: Unified Dynamic and Adaptive Q-iteration for Unknown Environment Path Planning and Exploration

Nasr Abdalmanan[a], Kamarulzaman Kamarudin[a, *], Muqri Muazam Zinal[a], Mohd Rizal Manan[a], and Victor Hernandez Bennetts[b]

[a] *Faculty of Electrical Engineering &, Universiti Malaysia Perlis, Arau, Perlis, 02600, Malaysia*

[b] *School of Science and Technology, Örebro University, Örebro, SE-701 82, Sweden*

* Corresponding author.

Email address: kamarulzaman@unimap.edu.my

Keywords: Dynamic programming, Path planning, Autonomous exploration, Mobile robots

A B S T R A C T

Autonomous navigation in unknown environments requires both efficient point-to-point path planning and reliable full environment exploration. This paper presents UDAQ, a Unified Dynamic and Adaptive Q-iteration framework that achieves local path planning, complete environment exploration, and oriented exploration behavior using a single reward-driven formulation. Unlike conventional approaches that rely on separate planners or task-specific heuristics, UDAQ adapts its behavior through reward configuration within a unified value iteration framework. A key feature of UDAQ is its ability to operate under dynamically evolving state spaces, as encountered in SLAM-based mapping, where the environment expands and changes over time. To ensure stable convergence, an adaptive reward scaling mechanism is employed, enabling effective value propagation as the state space grows and preventing policy degradation. UDAQ functions as a fully autonomous closed-loop system, continuously integrating SLAM updates, adapting its state representation and reward parameters, performing Q-iteration to convergence, and generating navigation commands without human intervention. The method is evaluated in three simulated environments of increasing structural complexity and compared against the ROS 2 NavFn planner for point-to-point navigation, as well as frontier-based exploration, wall-following, and random walk for exploration tasks. In point-to-point navigation, UDAQ reduced path length by approximately 8–12% and travel duration by 20–30% compared to NavFn, approaching global optimal solutions despite operating without prior map knowledge. In exploration tasks, UDAQ outperformed frontier-based exploration across all environments, reducing travel distance by 36–44% and exploration time by 48–59%, while also substantially exceeding the performance of simpler reactive methods. Directional exploration experiments further demonstrated that oriented reward configurations can significantly improve exploration efficiency and room visitation order. These results confirm that UDAQ provides a robust, efficient, and flexible solution for autonomous navigation in unknown environments, unifying multiple navigation behaviors within a single reward-driven framework and operating entirely as a fully autonomous closed-loop system.

1. Introduction

Autonomous navigation in unknown environments remains a central challenge in mobile robotics [1]. Robots operating without prior map information must incrementally build a representation of the environment while simultaneously planning collision-free trajectories using onboard sensing and mapping systems [2]. This problem is particularly relevant in applications such as search and rescue, autonomous inspection, logistics, and service robotics, where environments are partially structured and only partially observable [3].

In unknown environments, where the robot has no prior knowledge of the surroundings, exploration is necessary to gather spatial information. Exploration is the process of gradually transforming an unknown environment into a partially or fully known one by directing the robot toward unvisited regions [4]. It can serve as a standalone objective, such as complete environment coverage, or as a means of acquiring information to support navigation, enabling the robot to optimize its path as the map is incrementally built [3]. Path planning is the task of computing a collision-free route between two points in the environment [5], [6]. When the environment is fully known, global path planning methods can compute the optimal path over the complete map. In unknown environments, local path planning approaches instead build toward an optimal solution as new information becomes available [2]. Optimality in path planning is generally defined in terms of travel distance or duration, though in specific contexts other criteria may take priority. Safety-aware planners may trade path length for increased clearance from obstacles [7], while energy-conscious designs may reduce the number of turns to minimize high motor torque demand and lower overall power consumption [8].

To perform exploration in unknown environments, simultaneous localization and mapping are essential. Sensors such as LiDAR provide dense geometric measurements of the surroundings and serve as the primary source for map construction [9]. By fusing LiDAR measurements with inertial data and wheel odometry, a consistent foundation for both mapping and localization can be established. Simultaneous Localization and Mapping (SLAM) is among the most widely adopted

frameworks for this purpose, enabling the robot to incrementally build a map while estimating its own pose within it [2].

Traditional autonomous exploration architectures decompose the navigation problem into two largely independent components: exploration, which determines where the robot should go, and path planning, which determines how to reach that target. Frontier-based exploration methods, first formalized by Yamauchi [10], identify boundaries between known free space and unknown regions, referred to as frontiers, and direct the robot toward them to maximize spatial information gain. While frontier-based approaches are computationally tractable and have demonstrated strong performance across a range of environments, they evaluate candidate targets based primarily on geometric accessibility and expected coverage, without explicitly accounting for the cost or feasibility of the trajectory required to reach them [11]. Path planning is subsequently performed as a separate step using graph-search or sampling-based algorithms such as A* [12]. This decoupled architecture introduces a structural limitation: the exploration layer selects goals without awareness of path complexity or map uncertainty, while the planning layer operates on a snapshot of the map that may already be outdated by the time the robot begins execution. The result is a pipeline that frequently produces suboptimal navigation behavior, particularly in environments with narrow passages or dynamic frontiers [13].

Classical approaches to exploration and local navigation have addressed these challenges through a variety of geometric and reactive strategies. Frontier-based methods remain the dominant paradigm for coverage-driven exploration, guiding the robot incrementally toward the boundary of the known map [10]. Recent extensions have improved upon the basic frontier utility function by incorporating path cost estimates, sensor coverage models, and information-theoretic criteria to reduce redundant travel and improve overall coverage efficiency [14], [15]. Reactive strategies offer computationally efficient alternatives that operate without explicit map reasoning. Wall-following algorithms navigate by maintaining proximity to environmental boundaries, enabling the robot to systematically traverse structured enclosures without constructing or maintaining a global map [16], [17]. Random-walk

planners guide the robot through stochastic motion selection combined with reactive obstacle avoidance, allowing exploration to proceed in the absence of any prior spatial knowledge [18], [19]. Graph-search-based local planners such as NavFn , the default global planner in the Robot Operating System (ROS 2) Navigation Stack 2, compute collision-free paths on occupancy grids using Dijkstra's algorithm, propagating a wavefront of accumulated travel cost from the goal until the robot's current position is reached [20], [21]. NavFn provides deterministic, optimal paths with respect to grid-distance cost, but, like other classical planners, it treats exploration and path planning as separate concerns.

Reinforcement learning (RL) offers a principled framework for navigation by formulating the problem as a Markov Decision Process (MDP), in which an agent selects actions to maximize cumulative reward over time [22]. This reward-based optimization perspective is particularly attractive for exploration and path planning because it allows travel cost, coverage, and safety to be jointly encoded in a scalar objective, rather than handled by separate subsystems. Dynamic programming (DP) provides the theoretical foundation for solving MDPs exactly, and value iteration, one of its core algorithms, computes the optimal value function by iteratively applying the Bellman optimality operator until convergence [22]. In the context of grid-based path planning, value iteration propagates cost-to-go information backward from the goal across the navigable state space, yielding a globally consistent policy that maps every reachable cell to its optimal action [23], [24]. Unlike graph-search methods such as Dijkstra's algorithm, which compute a single shortest path, value iteration produces a dense value landscape from which locally optimal decisions can be extracted at any robot position, making it well-suited for reactive replanning as the map is incrementally updated during exploration. However, applying value iteration to autonomous exploration in unknown environments introduces non-trivial challenges. Because the navigable state space expands dynamically as new regions are observed, maintaining a fixed-size state-space representation incurs unnecessary computational overhead and restricts scalability in large environments [25]. Furthermore, static reward magnitude designs, in which goal and obstacle penalties are fixed a priori,

become ineffective when the value function must propagate across a state space of growing and unpredictable extent, resulting in poor Bellman backup quality, slow convergence, and degraded policy performance [26]. Addressing these limitations requires adaptive formulations of both the state-space representation and the reward system, capable of maintaining sound value propagation properties as the environment is progressively revealed.

In this work, we propose UDAQ, a Unified Dynamic and Adaptive Q-iteration framework for autonomous navigation in unknown environments. UDAQ introduces three primary technical contributions. First, it provides a unified multi-behavior path planning framework based on multiple reward configurations that tightly integrate exploration and path planning within a single formulation. Second, it employs a dynamic state space representation that evolves naturally with exploration progress. Third, to address the challenge of reward design in evolving environments, UDAQ introduces an automatic reward scaling strategy based on geodesic diameter estimation, ensuring stable value function propagation across environments of varying size and complexity.

The UDAQ framework operates as a fully autonomous closed-loop system that continuously incorporates SLAM updates, adapts its state space and reward parameters, performs Q-iteration to convergence, and generates navigation commands without human intervention. Experimental results obtained in ROS 2 simulation across multiple indoor environments demonstrate that UDAQ achieves improved navigation efficiency and consistently strong performance compared to established baseline methods, highlighting its potential suitability for autonomous exploration and local path planning.

2. Proposed Method

This section presents the proposed Unified Dynamic and Adaptive Q-iteration (UDAQ) framework for autonomous exploration and navigation in unknown environments. The method formulates both local path planning and exploration as a Markov Decision Process (MDP) and employs a dynamic state space representation derived from SLAM occupancy grids. Furthermore, the approach introduces configurable reward functions designed to induce multiple navigation behaviors within a single unified closed-loop planning framework. The solution is obtained through a Q-iteration procedure that accommodates a dynamically changing state space and incorporates adaptive reward tuning to improve convergence and policy stability.

2.1. Markov Decision Process Formulation

The local path planning and exploration task is formulated as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ, π). Here, S denotes the set of states representing possible robot positions, A denotes the set of actions the agent can perform to transit from current state s to future state s', P denotes the state transition probabilities, R denotes the reward function, and $\gamma \in [0, 1]$ is the discount factor that balances immediate and future rewards. The objective is to find a policy π that maps each state $s \in S$ to an action $a \in A$ such that the expected cumulative reward is maximized.

2.2. State Space Representation

The state space is derived from the occupancy grid map generated by the SLAM system. The dense occupancy grid is further segmented into a grid of square tiles, where each tile corresponds to a state $s \in S$. The tiling resolution δ, measured in meters square per tile, determines the granularity of the state space and provides a tunable parameter for balancing computational efficiency against spatial precision.

Each state is classified according to the occupancy values of the SLAM occupancy-grid cells within its corresponding tile. Five state classifications are defined: (i) free states, corresponding to unoccupied navigable space; (ii) occupied states, representing tiles that contains any obstacle cells;

(iii) frontier states, which are partially unknown but not obstructed and represent boundaries between explored and unexplored regions; (iv) goal states, when a specific target location is specified; and (v) unknown states, representing fully unexplored regions.

2.3. Action Space and Transition Function

The action set A comprises eight discrete movements corresponding to transitions to neighboring states in the cardinal and inter-cardinal directions: North, Northeast, East, Southeast, South, Southwest, West, and Northwest. Each action a ∈ A represents a one-step movement from the current state to an adjacent state in the specified direction.

The transition function P(s'|s, a) defines the probability of arriving at state s' at time t+1 given the current state s and executed action a at time t. Under the assumption of accurate robot localization provided by the SLAM system, transitions are modeled as deterministic. Specifically,

$$P(s'|s,a) = \begin{cases} 0 \; if s' is occupied \\ 1 \; otherwise \end{cases} \quad (1)$$

Where, P(s'|s, a) = 0 if s' is the neighboring state in direction a and s' is classified as occupied; otherwise, P(s'|s, a) = 1.

2.4. Reward Function Design

The reward function R(s, a) specifies the immediate reward received when the agent executes action a in state s. Since the transition function is assumed to be deterministic, the reward can be equivalently defined over state transitions as R(s, a, s') where s' is uniquely determined by (s, a). The design of this function is critical for shaping the agent's behavior to achieve the desired local path planning and exploration objectives. Three reward configurations are proposed, each yielding distinct mode.

2.4.1. Reward Function 1: Point-to-Point Local Path Planning

This configuration is designed for navigation to a specified goal location in an unknown environment. High positive rewards are assigned for transitions that lead to frontier states, with reward magnitude scaled by proximity to the goal location using a Euclidean distance ratio. The goal state receives the maximum positive reward when it is visible to the robot (i.e., the goal falls within a free state);

otherwise, the goal state is rewarded as an unknown state. Small negative rewards known as step rewards are assigned for transitions through free space to discourage unnecessary movement, while large negative rewards penalize attempts to enter occupied states and create a natural wall inflation effect by reducing the average rewards of free states adjacent to walls. Unknown states receive neutral rewards as they cannot be reliably classified.

$$R_1(s,a,s') = \begin{cases} r_{step}\ if s' \in S_{free} \\ r_{obstacle}\ if s' \in S_{occupied} \\ d_f(s')\ if s' \in S_{frontier} \\ r_{goal}\ if s' \in S_{goal} \\ 0\ if s' \in S_{unknown} \end{cases}, where: r_{goal} > d_f(s') > r_{step} > r_{obstacle} \quad (2)$$

Where $d_f(s)$ is the normalized proximity function between the input state and the goal state defined as:

$$d_f(s) = \left(1 - d_{norm}(s, goal)\right), d_{norm}(s, goal) = \frac{\sqrt{\left[(s_i - goal_i)^2 + \left(s_j - goal_j\right)^2\right]}}{d_{max}} \quad (3)$$

with d_{max} representing the maximum possible distance in the environment, defined as the diagonal length of the map. The inverse factor makes higher values correspond to closer to goal. This reward structure encourages the agent to explore toward the goal direction with the minimum number of moves while maintaining persistence and avoiding obstacles. As a result, the agent is expected to dynamically re-plan its path when encountering local maxima. As the SLAM map is updated and new regions or previously unknown obstacles are revealed, blocked pathways are identified and the policy redirects the robot toward alternative open routes. These alternative routes may not have been selected initially due to lower reward values at earlier stages of exploration, as illustrated in Fig. 1.

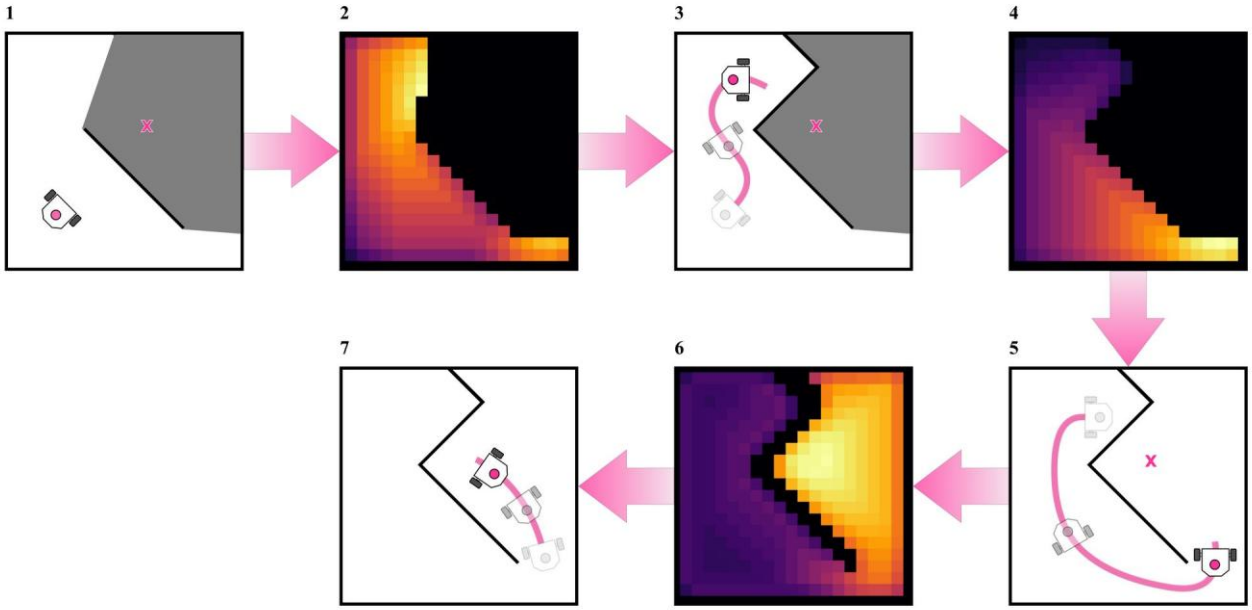


Fig. 1. Illustration of robot navigation behavior under reward function 1, showing dynamic path re-planning in response to updated SLAM map information.

2.4.2. Reward Function 2: Environment Exploration

For systematic environment exploration without a specified goal, the reward function is modified by removing goal-related components. Frontier states receive uniform high positive rewards independent of any directional preference, encouraging comprehensive exploration of the environment. Small negative rewards for free space transitions and large negative rewards for occupied state transitions are retained.

$$R_2(s,a,s') = \begin{cases} r_{step}\ if s' \in S_{free} \\ r_{obstacle}\ if s' \in S_{occupied} \\ r_{frontier}\ if s' \in S_{frontier} \\ 0\ if s' \in S_{unknown} \end{cases}, where: r_{frontier} > r_{step} > r_{obstacle} \quad (4)$$

This configuration promotes efficient exploration by rewarding discovery of new nearby frontiers while penalizing redundant traversal of known regions. The frontier rewards propagate through value iteration, creating an inherent preference toward the nearest frontiers and effectively sorting exploration priorities by proximity.

2.4.3. Reward Function 3: Oriented Exploration

This configuration introduces directional preferences to the exploration process by placing an imaginary goal location outside of the environment boundary. The reward function retains the proximity-based scaling for frontier states as $R_1(s, a, s')$ in (2), effectively guiding exploration in the direction of the imaginary goal. Eventually, the full environment will be explored as the robot continues searching for the unreachable goal, resulting in different exploration paths depending on the goal location. This oriented exploration mode allows operators to specify preferred exploration directions, which may be beneficial when prior knowledge suggests areas of interest or when exploration priority in a particular direction is desired.

2.5. Solution Procedure

In this work, the previously formulated MDP is solved using Q-iteration, which based on the value iteration that computes state values according to the Bellman optimality principle. The optimal value function V*(s) satisfies the Bellman optimality equation:

$$V*(s) = \max_a [R(s,a) + \gamma V*(s')] \tag{5}$$

Where the maximum is taken over all actions a ∈ A, and s' denotes the state resulting from executing action a in state s. This recursive formulation expresses the optimal value of a state as the immediate reward plus the discounted optimal value of the successor state, assuming optimal action selection. To enable action-specific value estimation, the formulation is extended to the Q-function, which assigns values to state-action pairs. The relationship between the optimal value function and the Q-function is given by:

$$V*(s) = \max_a Q(s,a) \tag{6}$$

The Q-values are updated iteratively according to the temporal difference update rule:

$$Q(s,a) = Q(s,a) + \left[R(s,a) + \gamma \max_a Q(s',a') - Q(s,a)\right] \tag{7}$$

where $Q(s, a)$ is the current Q-value of performing action a in state s , and $R(s, a)$ is the immediate reward received for that action. The term γ max Q(s', a') represents the best expected future return

from the next state s' discounted by the factor γ. The expression in brackets is known as the temporal difference (TD) error, which quantifies the gap between the current estimate and the updated return. This update is applied iteratively for each state-action pair following the infinite horizon formulation, where iterations continue until convergence is achieved. Convergence is determined when no significant changes occur in the policy map between successive iterations, indicating that the Q-values have stabilized and the optimal policy has been identified. Upon convergence, the optimal policy π* is derived by selecting, for each state, the action that maximizes the Q-value:

$$\pi*(s) = \underset{a}{argmax} Q(s,a) \tag{8}$$

Path generation proceeds by following the policy from the robot's current state, as determined by its pose on the state map, until a terminal state is reached.

2.6. Dynamicity and Adaptation

Full convergence is required to extract an optimal policy for the robot's local path planning. Convergence quality can be assessed using the state-action value function, which represents the normalized Q-values across the state space. Under good convergence, the value function exhibits a smooth and well-contrasted gradient, clearly distinguishing low- and high-value states, and yielding a stable and interpretable policy map. On the other hand, poor convergence is characterized by weak or low-contrast transitions between states, particularly in regions where values are heavily discounted, resulting in unstable policy decisions.

Unstable convergence may arise from improperly scaled reward magnitudes, an unsuitable discount factor, or a combination of both. For instance, excessively large negative step penalties can cause the cumulative discounted cost of moving toward the goal to exceed the terminal reward, biasing the learned policy toward suboptimal actions. While the discount factor is typically fixed to emphasize either short-term or long-term rewards, reward magnitudes must be carefully tuned to match the size of the state space.

In dynamic SLAM observation-based environments, the state space evolves as new regions are discovered and previously free areas may be reclassified as occupied. Fixed reward magnitudes may

be adequate in smaller early-stage environments, but as the state space expands, the previously configured reward values become insufficient, leading to weak value propagation and degraded policy quality in some regions. Moreover, even if the robot starts near the initial high-reward zone, the propagation of high-value states can shift to newly discovered regions, leaving the agent in areas with poor policy guidance. This behavior is illustrated in Fig. 2: at T1, in a smaller environment, fixed rewards propagate effectively, creating a clear value function gradient and stable policy; at T2, in a larger environment, the same reward configuration is insufficient, producing weak value gradients and unstable policy behavior.

These challenges motivate the use of an adaptive reward scaling mechanism within UDAQ, which dynamically adjusts reward magnitudes relative to the evolving state space, ensuring robust value propagation and consistent policy quality regardless of environment size or structural changes.

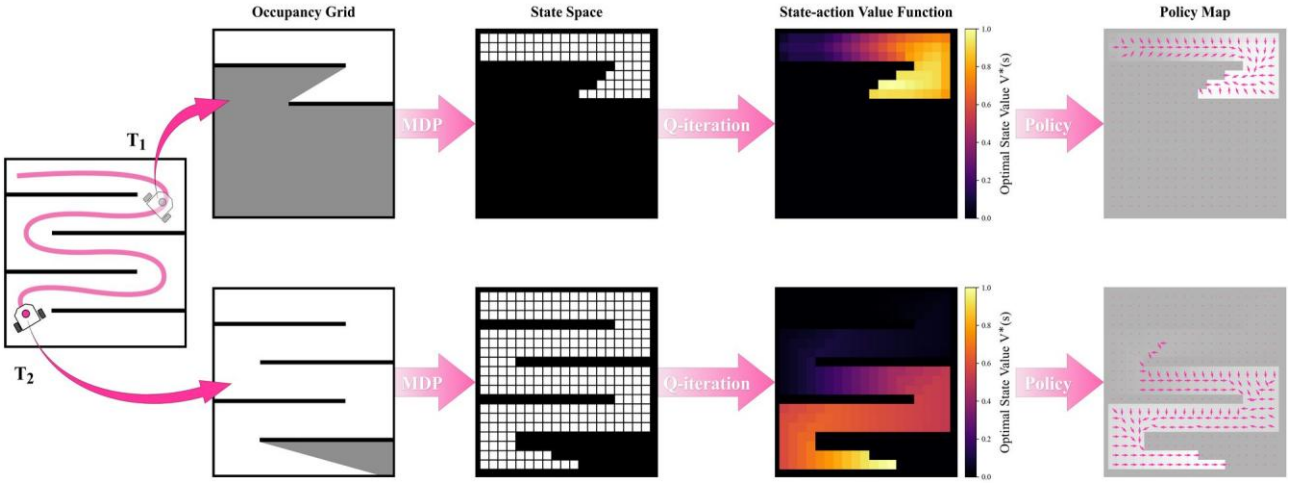


Fig. 2. Effect of environment size on value function propagation. Fixed reward magnitudes generate a clear value gradient and stable policy in smaller environments (T1) but produce weakened gradients and unstable policy in larger environments (T2).

In this work, we propose an adaptive reward tuning approach that adjusts reward magnitudes according to the current state space size. This method ensures consistent value function propagation throughout the environment and maintains reliable convergence regardless of dynamic state space evolution.

The adaptive reward magnitude tuner estimates the longest traversable trajectory in the state space by explicitly modeling its geometry and approximating its geodesic diameter. Let the discrete dynamic state space be denoted by $F \in [S_{Free}, S_{Frontier}]$. A binary mask $M:S \rightarrow \{0,1\}$ is constructed such that $M(s) = 1$ if $s \in F$ and $M(s) = 0$ otherwise. A Chebyshev distance transform is then applied to compute distance for each state to the nearest obstacle:

$$d_{Chebyshev}(s) = \min_{o \in s} \|s - o\|_{\infty}, s \in F \tag{9}$$

The medial-axis skeleton $K \in F$ is extracted as the set of local maxima of $d_{chebyshev}(s)$ and represented as an undirected graph:

$$G = (V, E) \tag{10}$$

where vertices correspond to skeleton cells and edges encode adjacency. The geodesic distance between two vertices is defined as the shortest path length $d_g(u, v)$ in G, and the geodesic diameter L is approximated as:

$$L = max_{u,v \in V}[d_g(u, v)] \tag{11}$$

A two-pass breadth-first search (BFS) is employed by first identifying the farthest vertex from an arbitrary seed and then measuring the maximum distance from that vertex. The resulting longest meaningful traversable trajectory provides an estimate of the state space geodesic diameter which forms the basis for adaptively scaling reward magnitudes to ensure consistent value-function propagation as the environment evolves. Using this diameter, the penalties are set proportional to the environment scale,

$$r_{step} = \frac{1}{L}, r_{obstacle} = \frac{2}{L} \tag{12}$$

Ensuring that the cumulative cost along the longest path remains meaningful, while obstacle penalties are defined to dominate the step penalty discouraging collisions regardless of environment size. Reward scaling for goal-specified path planning or exploration scenarios is then handled automatically. When a goal is provided, the maximum reward is assigned to the goal location and scaled to dominate cumulative discounted step penalties along the geodesic diameter.

$$r_{goal} = \frac{1}{(1-\gamma^{L})} \tag{13}$$

Frontier rewards are subsequently assigned based on their proximity to the goal $d_f(s)$, where s is the given frontier state.

$$r_{frontier} = \frac{\frac{1}{(1-\gamma^{L})} - d_f(s)}{L} = r_{goal}\left(\frac{1-d_f(s)}{L}\right) \tag{14}$$

producing a gradient of incentives that encourages movement toward the goal while preserving the reward ordering $r_{goal} > r_{frontier} > r_{step} > r_{obstacle}$. In scenarios where no goal is specified, all frontiers are given a uniform maximum reward,

$$r_{frontier} = \frac{1}{(1-\gamma^{L})} \tag{15}$$

while step and obstacle penalties remain the same. This ensures consistent behavior in exploration tasks, maintaining the ordering $r_{frontier} > r_{step} > r_{obstacle}$ without requiring manual tuning of reward constants. By tying all reward magnitudes to the geodesic diameter and discount factor, the system automatically adapts to the dynamic state space as it grows or refined during exploration , guaranteeing meaningful scale and value function propagation Fig. 3.

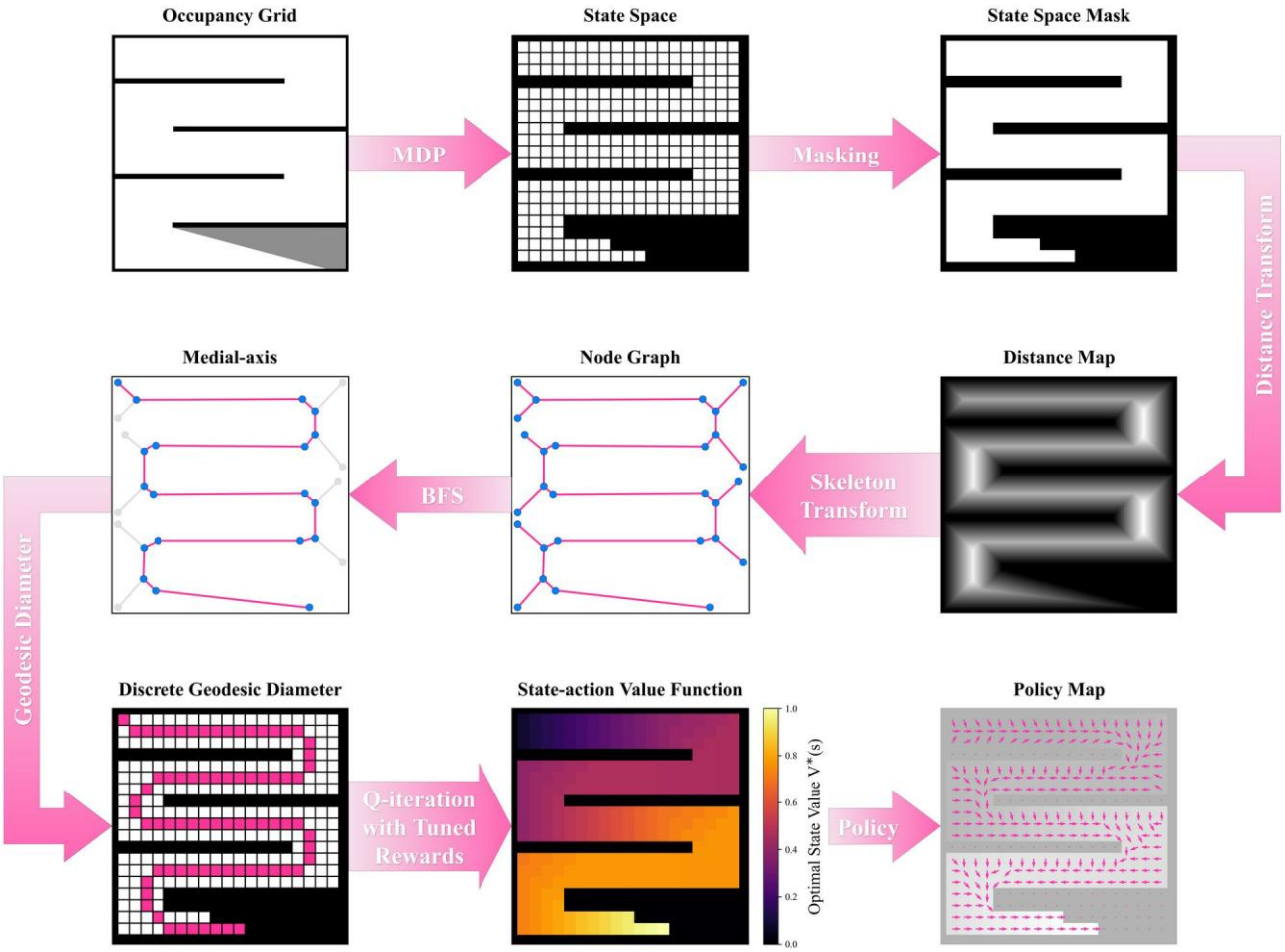


Fig. 3. Adaptive reward tuning process.

As the robot explores and the SLAM map evolves, the state classifications, reward function, and Q-values must be updated to reflect newly discovered environmental information. The proposed method implements periodic re-computation of these components to enable adaptive behavior. When significant map changes are detected, the state classification is updated based on new occupancy information, the reward function is recalculated for affected states, and Q-iteration is performed to update the policy. This adaptive mechanism ensures that the robot's navigation decisions remain consistent with the most current environmental knowledge, enabling robust navigation in dynamically discovered environments. This architecture constitutes a closed-loop system that continuously feeds updated information back into the decision-making process and automatically terminates when the task is completed Fig. 4.

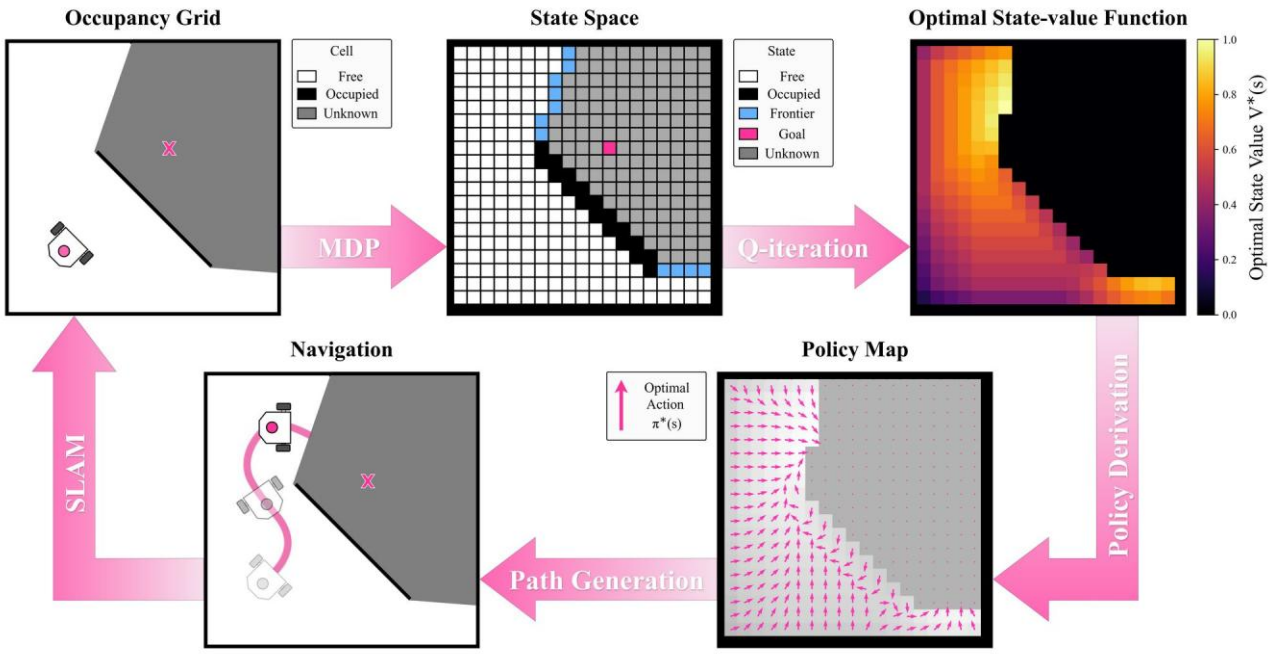


Fig. 4. UDAQ closed-loop system illustration.

3. Experimental Setup

3.1. Simulation Environment

The proposed method was evaluated in ROS 2 simulations using three 15 × 15 m indoor environments of varying structural complexity. Map A (Simple) comprises mostly open spaces, representing warehouses or large halls. Map B (Moderate) contains segmented rooms connected by corridors, reflecting typical office or residential layouts. Map C (Complex) includes multiple rooms and narrow passages, representing dense indoor spaces Fig. 5. UDAQ used a tiling resolution δ equal to the robot footprint 0.21 m to ensure safe navigation and a discount factor of 0.95 to prioritize long-term rewards.

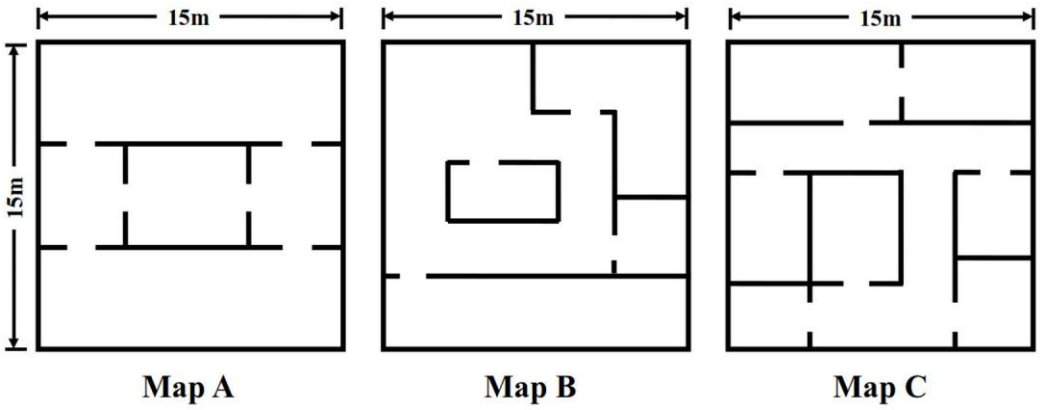


Fig. 5. Simulation environments.

3.2. Comparative Methods

The proposed method was compared against established navigation and exploration algorithms. For local path planning evaluation, the Nav2 NavFn Planner, which implements Dijkstra search based path planning within the ROS 2 navigation stack, served as the baseline. Additionally, the global optimal path computed using Dijkstra's algorithm with complete map knowledge provided an upper bound for performance assessment. For exploration evaluation, three comparative methods were implemented: frontier-based exploration, which identifies and navigates to boundaries between known and unknown regions; wall-following, which maintains contact with environmental boundaries to ensure coverage; and random walk, which employs simple stochastic movement patterns with obstacle avoidance. The selected exploration baselines represent increasing levels of capability. Random walk provides a minimal, non-deliberative lower bound, wall-following introduces reactive structure through environmental contact, and frontier-based exploration represents the strongest classical approach by explicitly reasoning about unknown space. This hierarchy enables evaluation of UDAQ across progressively more capable exploration strategies.

3.3. Evaluation Metrics

Performance was assessed using multiple quantitative metrics. Travel distance measured the total path length in meters from start to completion. Travel time measured the duration in seconds required to complete the navigation or exploration task. For local path planning experiments, the optimal path length computed with full map knowledge provided a reference for path length efficiency assessment. For exploration experiments, the Exploration Index (EI) was introduced to quantify exploration progress. The Exploration Index is defined as:

$$EI = 1 - \left(\frac{unknown\ cells}{total\ cells}\right) \tag{16}$$

where cells are defined in the SLAM occupancy grid. This metric is normalized to the range [0, 1], with EI = 0 indicating no exploration (entire map unknown), EI = 1 indicating complete exploration (all explorable areas discovered), and intermediate values indicating partial exploration progress. The

Exploration Rate was derived as EI/s, quantifying the rate of exploration progress in terms of map coverage per second. This derivative metric provides insight into the temporal efficiency of different exploration strategies.

3.4. Experimental Procedures

Three experiments were conducted to evaluate different aspects of the proposed method. Experiment 1 assessed point-to-point local path planning capability by tasking the robot with navigating between specified start and goal locations in unknown environments. Multiple start-goal pairs were tested across all three maps, with results compared to NavFn Planner and the global optimal path. Experiment 2 evaluated the exploration capability of the proposed method by comparing it against frontier-based exploration, wall-following, and random walk across all three environments. Exploration Index, travel time, path length, and exploration rate were recorded for each method in each environment. Experiment 3 investigated the oriented exploration functionality by testing eight directional configurations (East, Northeast, North, Northwest, West, Southwest, South, Southeast) on Map C. This experiment aimed to understand how directional preferences affect exploration efficiency and exploration paths.

4. Results

4.1. Experiment 1: Point-to-Point Local Path Planning

The first experiment evaluated UDAQ's capability for local path planning in unknown environments. The robot was tasked with finding routes between specified locations while minimizing travel time and distance. Although neither NavFn nor UDAQ could identify the true optimal path due to the lack of prior map information, UDAQ generated trajectories that were consistently shorter and smoother than those produced by NavFn, demonstrating more efficient local path planning Fig. 6.

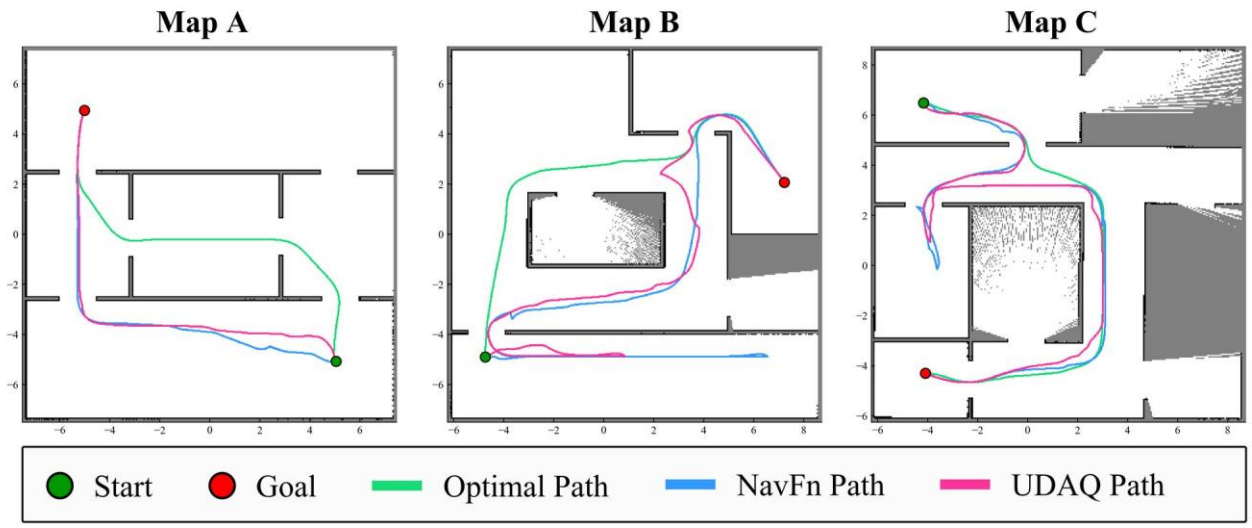


Fig. 6. Comparison of path trajectories in unknown environments. The optimal path is shown for reference. While neither NavFn nor UDAQ could reach the true optimal path without prior map information, UDAQ produced shorter and smoother trajectories than NavFn.

Table 1 presents a comparison of path lengths between UDAQ, NavFn, and the globally optimal path. The optimal path, computed with complete map knowledge, serves as a theoretical lower bound. Both UDAQ and NavFn deviate from this optimum due to the unknown environment, as path planning is performed concurrently with exploration.

Table 1

Comparison of travel durations and distances for UDAQ, NavFn, and the global optimal path across random test cases.

Map	Start	Goal	UDAQ		NavFn		Optimal
			Travel Distance (m)	Duration (s)	Travel Distance (m)	Duration (s)	Travel Distance (m)
A	(5,-5)	(-5,5)	19.01	84	19.84	111	18.08
B	(-5,-5)	(7,2)	32.76	147	42.87	227	20.37
C	(0,0)	(7,7)	33.14	151	31.08	153	24.18
C	(-4,7)	(-4,-4)	33.05	151	37.50	250	21.78
C	(7,-3)	(-4,0)	35.97	163	36.24	195	22.93

Analysis of the results indicates that UDAQ consistently outperformed NavFn in both travel distance and completion time. Across the five test cases, UDAQ reduced average travel distance by approximately 8–12% and decreased average completion time by about 20–30%, relative to NavFn. These improvements reflect UDAQ's ability to more effectively balance exploration and exploitation, minimizing unnecessary traversal while maintaining smooth and efficient local paths in unknown environments. Performance gains vary across environments due to differences in structural complexity, but UDAQ consistently produces shorter and faster paths than NavFn.

4.1.1. Reward Function 1 Hypothesis Test

As described in Section 2.4.1, the reward function 1 for local path planning was hypothesized to encourage persistent navigation toward unknown goal locations while enabling dynamic re-planning. To test this hypothesis, additional obstacles were introduced into Map A across several scenarios, and the robot was tasked with reaching the goal in an unknown environment. The results demonstrate robust planning and re-planning capabilities, with the robot successfully reaching the goal in all trials Fig. 7.

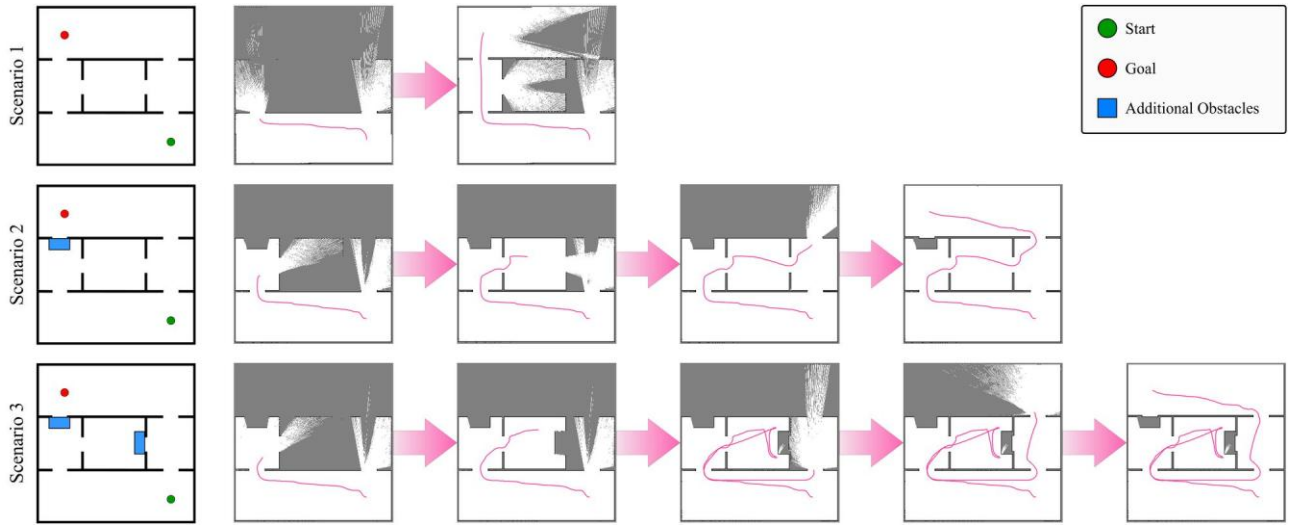


Fig. 7. Example of UDAQ navigating Map A with additional obstacles. The robot dynamically re-plans its path to reach the goal while avoiding newly introduced obstacles, illustrating persistent and adaptive navigation behavior.

4.2. Experiment 2: Environment Exploration

The second experiment compared UDAQ with three baseline exploration strategies: frontier-based exploration, wall-following, and random walk, across three virtual environments of increasing complexity. Results are reported per map and organized into three components: (1) exploration snapshots with trajectory overlays, (2) plots of travel distance and exploration index over time, and (3) summary tables of travel distance, duration, and exploration index.

4.2.1. Map A

Fig. 8 shows exploration snapshots across seven timestamps, with trajectories for all four methods. In Map A's open areas, Random Walk explored most of the map by row 6 but required manual termination, following straight-line paths that bounced off walls and stayed close to obstacles, raising safety concerns. Wall-Following covered the entire map by row 5 but continued along boundaries due to the lack of intelligent termination, keeping the robot near obstacles even after completion. Frontier-Based exploration achieved systematic, goal-directed coverage, completing by row 5 and terminating autonomously, with a jagged trajectory caused by frequent rotations. UDAQ completed exploration the fastest, terminating by row 4, and exhibited smoother trajectories indicative of more efficient navigation and reliable autonomous completion detection.

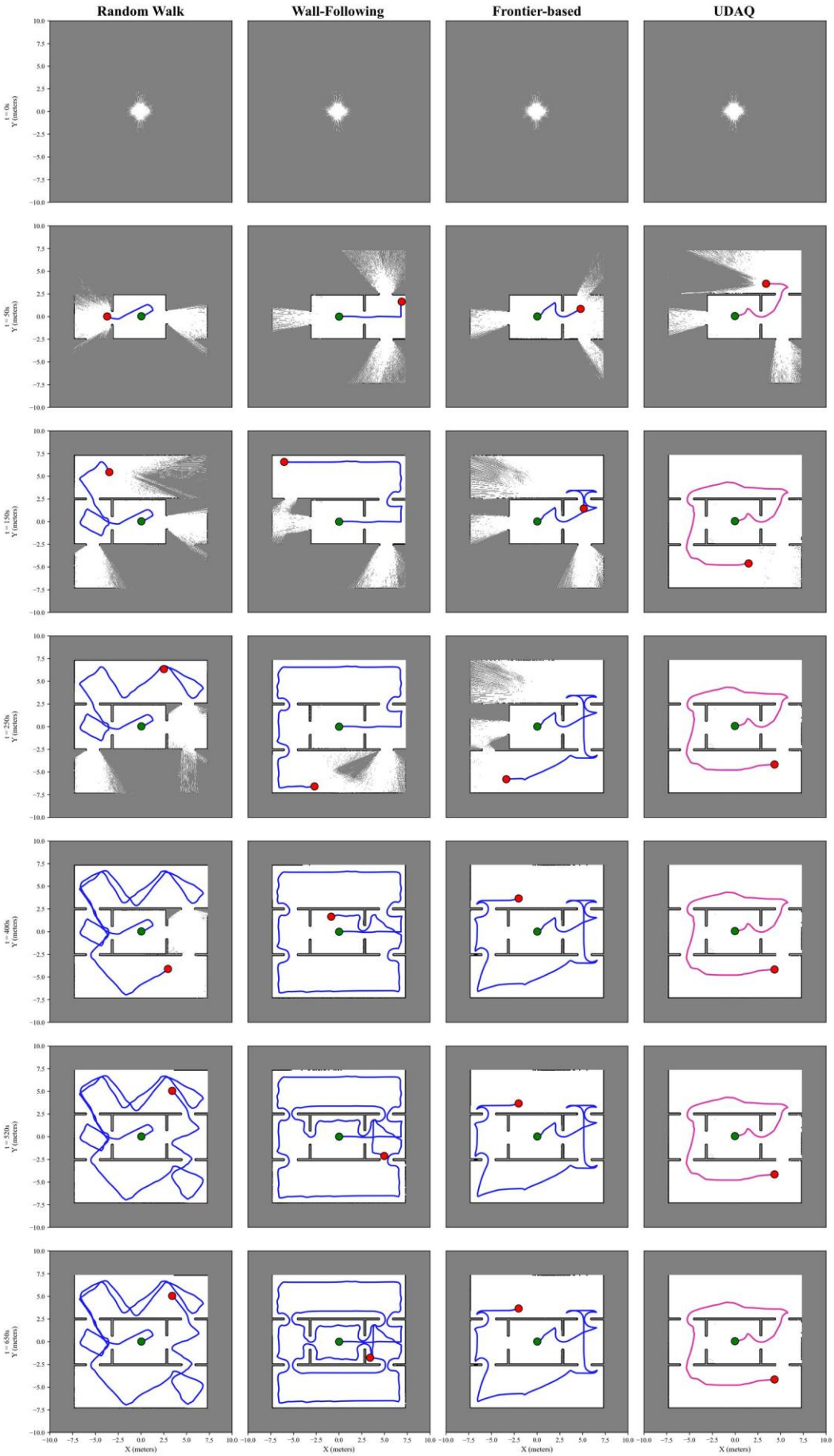


Fig. 8. Map A: exploration progress snapshots for all methods, showing trajectory overlays at multiple timestamps.

Fig. 9 presents the quantitative analysis of travel distance and exploration index (EI) over time for all four methods in Map A.

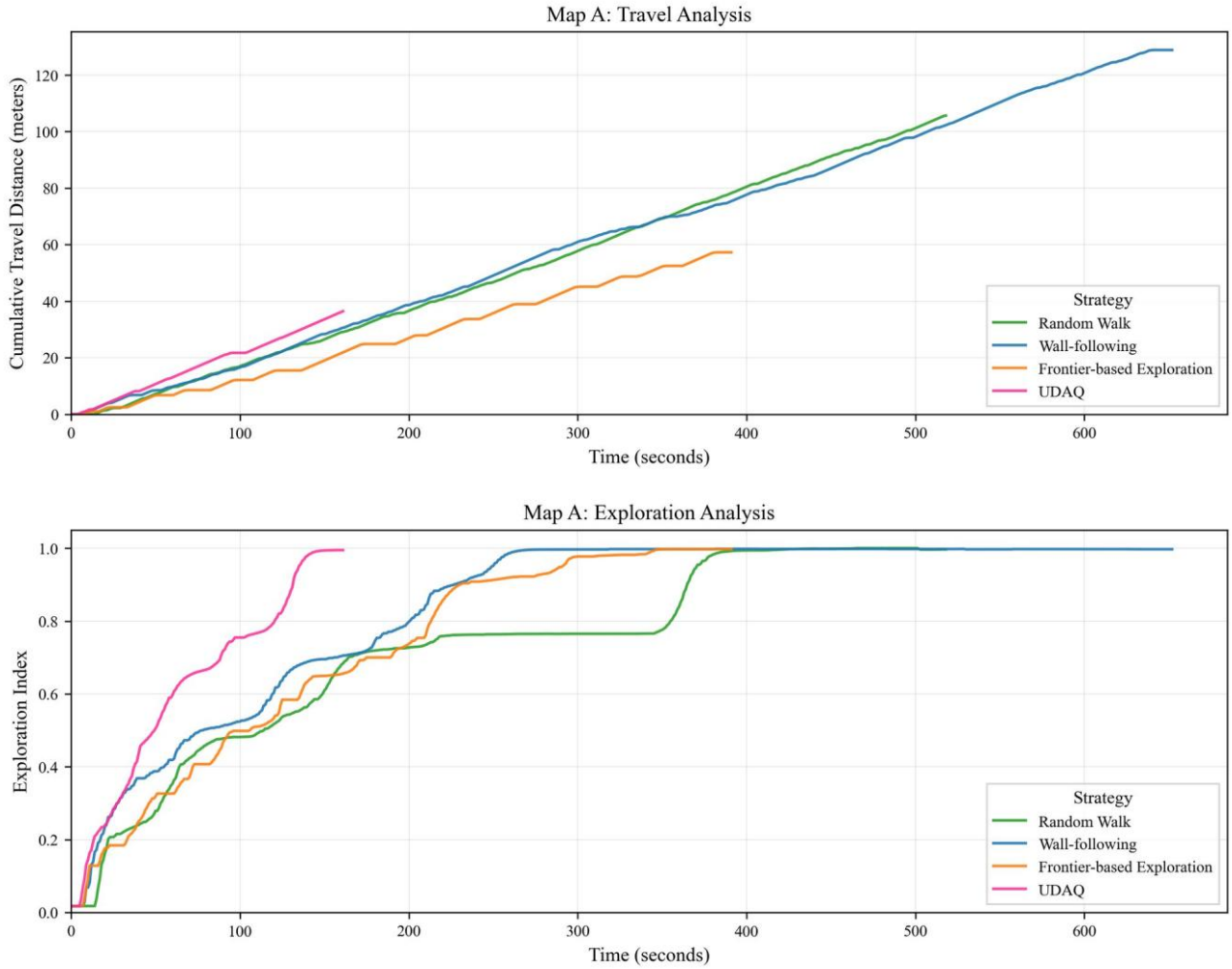


Fig. 9. Map A: travel and exploration analysis over time for all methods.

Travel Analysis: The travel distance plot highlights differences in navigation efficiency. Random Walk traveled approximately 100 m in 500 s, showing a semi-linear increase in distance, reflecting constant movement without strategic guidance. Wall-Following covered about 130 m in 650 s, following a similar semi-linear pattern, indicating systematic but inefficient exploration. Frontier-Based exploration completed the task in roughly 60 m over 400 s, with a staircase-like trajectory reflecting periodic stops for frontier selection and decision-making. UDAQ demonstrated the best performance, completing exploration in only 35 m and 150 s, following the most efficient path.

Exploration Analysis: All methods eventually achieved full coverage (EI ≈ 1.0), but efficiency varied. Random Walk exhibited extended flat regions, indicating periods of movement without meaningful

exploration. Wall-Following progressed steadily at first but plateaued after about 250 s, continuing along walls after full coverage and wasting time. Frontier-Based maintained a consistent pace, while UDAQ showed the steepest and fastest exploration curve with minimal flat regions, demonstrating continuous and efficient coverage.

These results confirm that UDAQ outperforms the other methods in Map A, completing exploration faster, with shorter travel distances, and avoiding the inefficiencies inherent to Random Walk, Wall-Following, and Frontier-Based approaches.

Table 2 summarizes exploration performance in Map A

Table 2

Map A: summary of travel distances, durations, and exploration efficiency for all methods.

Strategy	Travel Distance (m)	Duration (s)	Final Exploration Index (EI)	Exploration Rate (EI/s)
Random walk	105.63	517	1.0	0.19%
Wall-following	128.83	651	0.998	0.15%
Frontier-based	57.27	390	0.997	0.25%
UDAQ	36.45	160	0.995	0.61%

All methods achieved full coverage (EI ≈ 1.0), but efficiency varied widely. Random Walk traveled 100 m in 517 s (0.19 EI/s) and required manual termination. Wall-Following covered 130 m in 651 s (0.15 EI/s), continuing along walls after completion. Frontier-Based exploration completed the task in 57.27 m and 390 s (0.25 EI/s), with periodic pauses for frontier selection. UDAQ achieved the fastest and most efficient exploration, completing coverage in 36.45 m over 160 s (0.61 EI/s). Compared to Frontier-Based, UDAQ reduced travel distance by ~36% and completion time by ~59%; reductions relative to Wall-Following and Random Walk exceeded 70% for both distance and time. These results highlight UDAQ's substantial efficiency gains in open environments.

4.2.2. Map B

Fig. 10 presents exploration snapshots at eight timestamps for all four methods in Map B, a moderately complex environment with segmented rooms and connecting corridors. Random Walk

exhibits its typical straight-line, bouncing behavior but struggles in structured areas. By row 4, the robot becomes trapped in the upper-right room, producing dense, inefficient paths until it escapes and completes exploration by row 8. Wall-Following performs better than in Map A, taking advantage of concave walls and the room-corridor layout. Exploration completes by row 6, although the robot continues moving until row 8 due to the lack of autonomous termination. The trajectory remains close to walls, effectively utilizing structural features for navigation. Frontier-Based exploration completes by row 7 with reasonable efficiency. Its trajectory is less jagged than in Map A, as the structured environment provides clearer frontier boundaries, enabling more decisive navigation. UDAQ completes exploration fastest, finishing by row 5 with the smoothest trajectory. A minor area remains unscanned, but it is below the detection threshold and does not affect overall coverage. UDAQ demonstrates superior efficiency and smooth navigation compared to all other methods.

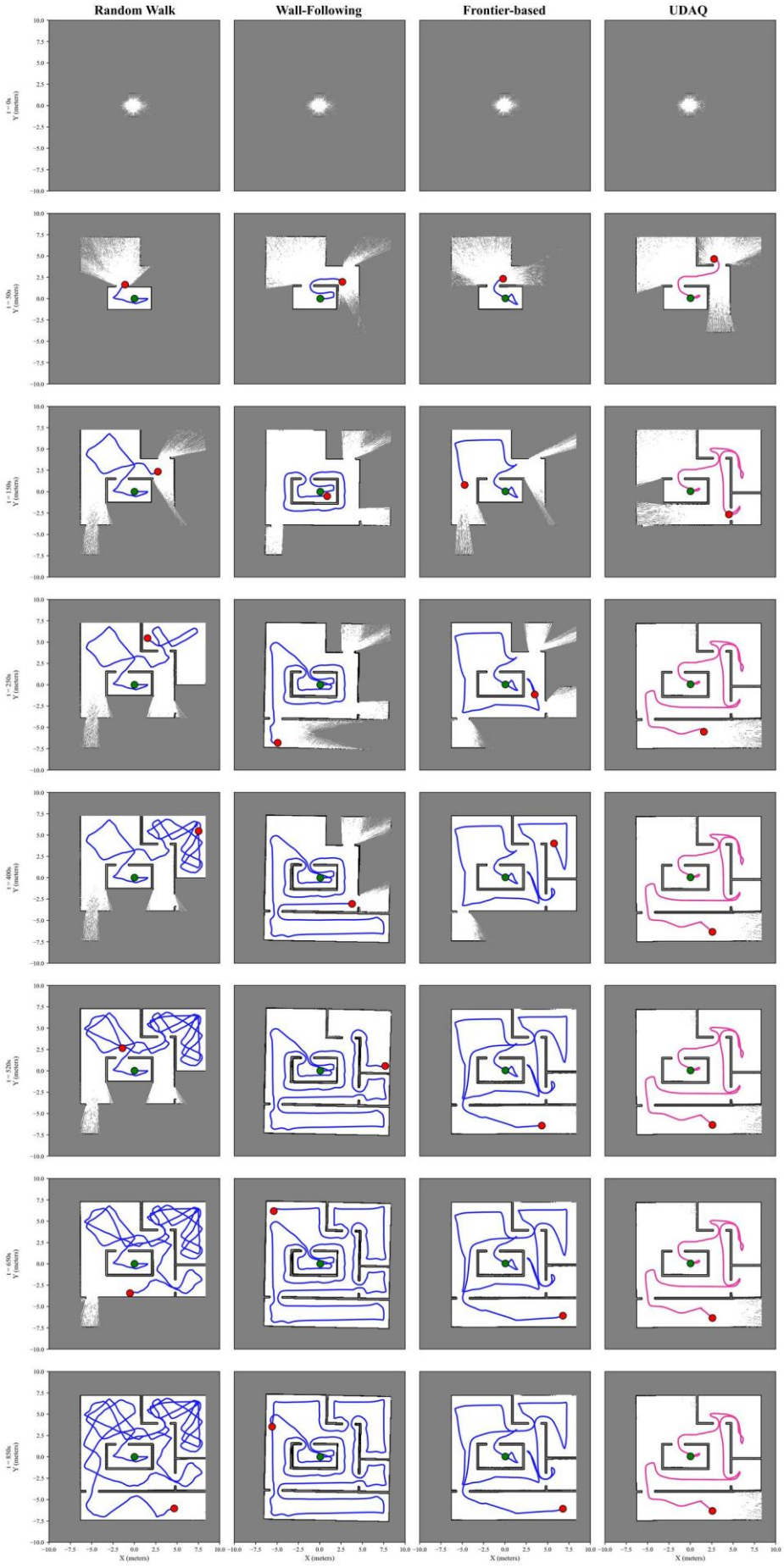


Fig. 10. Map B: exploration progress snapshots for all methods, showing trajectory overlays at multiple timestamps.

Fig. 11 presents the quantitative analysis of travel distance and exploration index (EI) over time for all four methods in Map B, a moderately structured environment with segmented rooms and corridors.

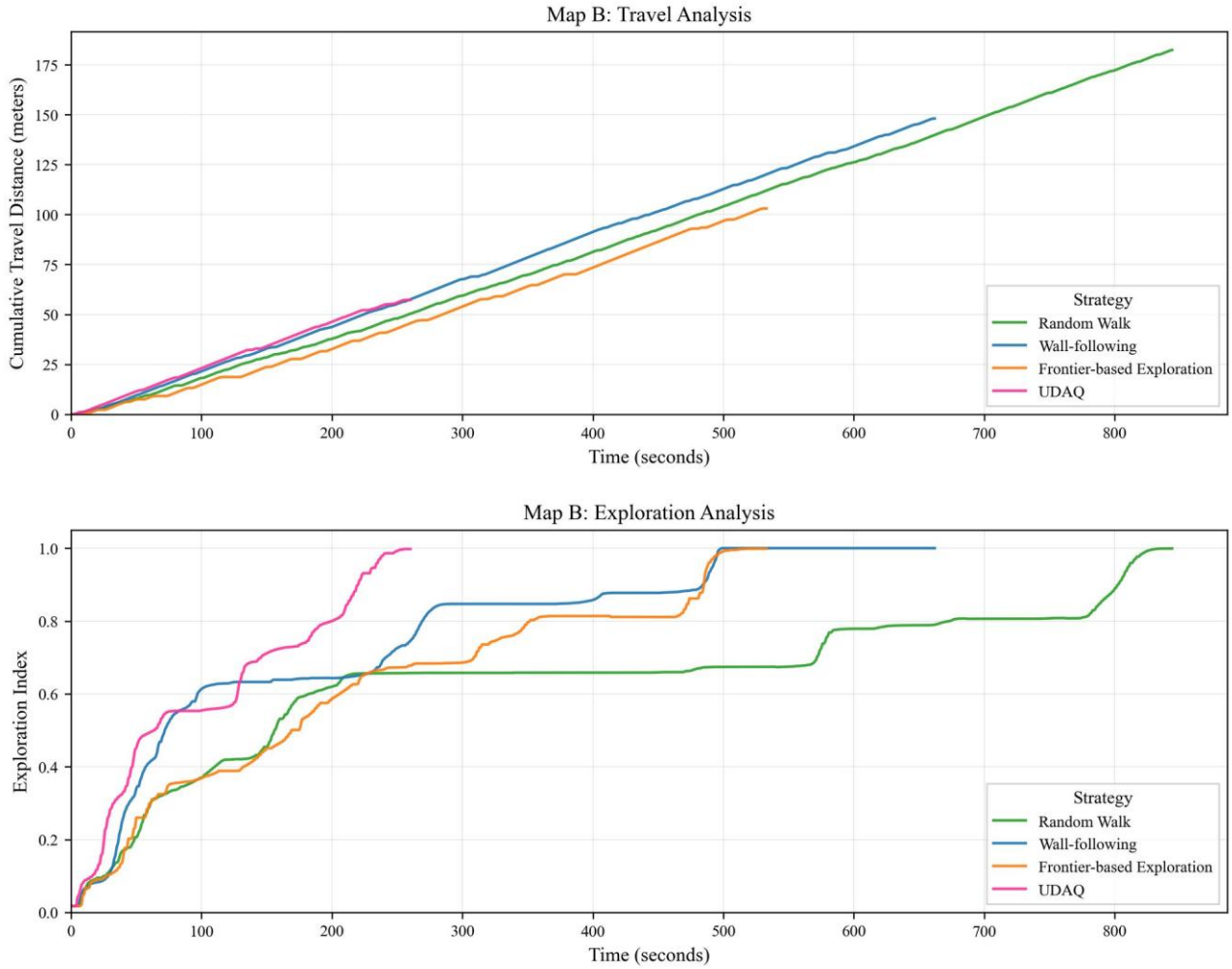


Fig. 11. Map B: travel and exploration analysis over time for all methods.

Travel Analysis: Random Walk traveled approximately 180 m in 840 s, maintaining a linear distance-time relationship despite periods of being trapped, which reduced spatial efficiency. Wall-Following covered about 150 m in 660 s, showing improved efficiency compared to Map A due to better alignment with walls and concave structures. Frontier-Based exploration completed the task in 100 m over 530 s, progressing mostly linearly with minor staircase effects reflecting brief frontier decision points. UDAQ completed exploration in 60 m over 260 s, demonstrating the most efficient path utilization.

Exploration Analysis: Random Walk exhibited flat regions between 200–580s and 600–790s, corresponding to trapping and inefficient movement, reaching full coverage (EI ≈ 1.0) only after 800s.

Wall-Following displayed flat regions between 270–480s and 500–660s, with the latter indicating continued motion after full exploration. Frontier-Based exploration maintained consistent progress, completing coverage by 530 s. UDAQ displayed the steepest and fastest exploration curve, with only a small flat region between 50–120s caused by scanning previously explored areas, demonstrating efficient adaptation to the room-corridor structure.

Table 3 summarizes exploration performance in Map B.

Table 3

Map B: summary of travel distances, durations, and exploration efficiency for all methods.

Strategy	Travel Distance (m)	Duration (s)	Final Exploration Index (EI)	Exploration Rate (EI/s)
Random walk	182.36	843	0.999	0.12%
Wall-following	148.07	661	1.0	0.16%
Frontier-based	103.05	532	0.999	0.18%
UDAQ	57.25	259	0.998	0.38%

Random Walk struggled, covering 182.36 m in 843 s (0.12 EI/s). Wall-Following improved slightly, achieving 148.07 m in 661 s (0.16 EI/s) but lacked intelligent termination. Frontier-Based completed exploration in 103.05 m and 532 s (0.18 EI/s), with smoother progression due to clearer frontier boundaries. UDAQ completed the task in 57.25 m over 259 s (0.38 EI/s), demonstrating the most efficient coverage. Compared to Frontier-Based, UDAQ reduced travel distance by ~44% and time by ~51%, while reductions relative to Wall-Following and Random Walk exceeded 60% in both metrics, highlighting UDAQ's effectiveness in structured indoor environments.

4.2.3. Map C

Fig. 12 presents exploration snapshots at eight timestamps for all four methods in Map C, a complex environment with multiple rooms and narrow corridors. Random Walk struggles significantly, becoming trapped in confined corridors by row 4. Dense, inefficient paths form in trapped areas, and the method fails to fully explore the environment by the final timestamp, leaving several rooms unvisited. Wall-Following performs effectively, completing exploration by row 6 and continuing until row 7 due to systematic constraints. The trajectory remains close to walls, allowing the robot to navigate the dense structure efficiently. Frontier-Based exploration completes by row 6. The trajectory exhibits increased jaggedness compared to Map B, reflecting the added complexity of frontier selection among multiple rooms and corridor intersections, which increases computational decision-making. UDAQ completes exploration fastest, finishing by row 5 with the smoothest path among all successful methods. It navigates the complex network efficiently, maintaining smooth and continuous progress while achieving full coverage of the environment.

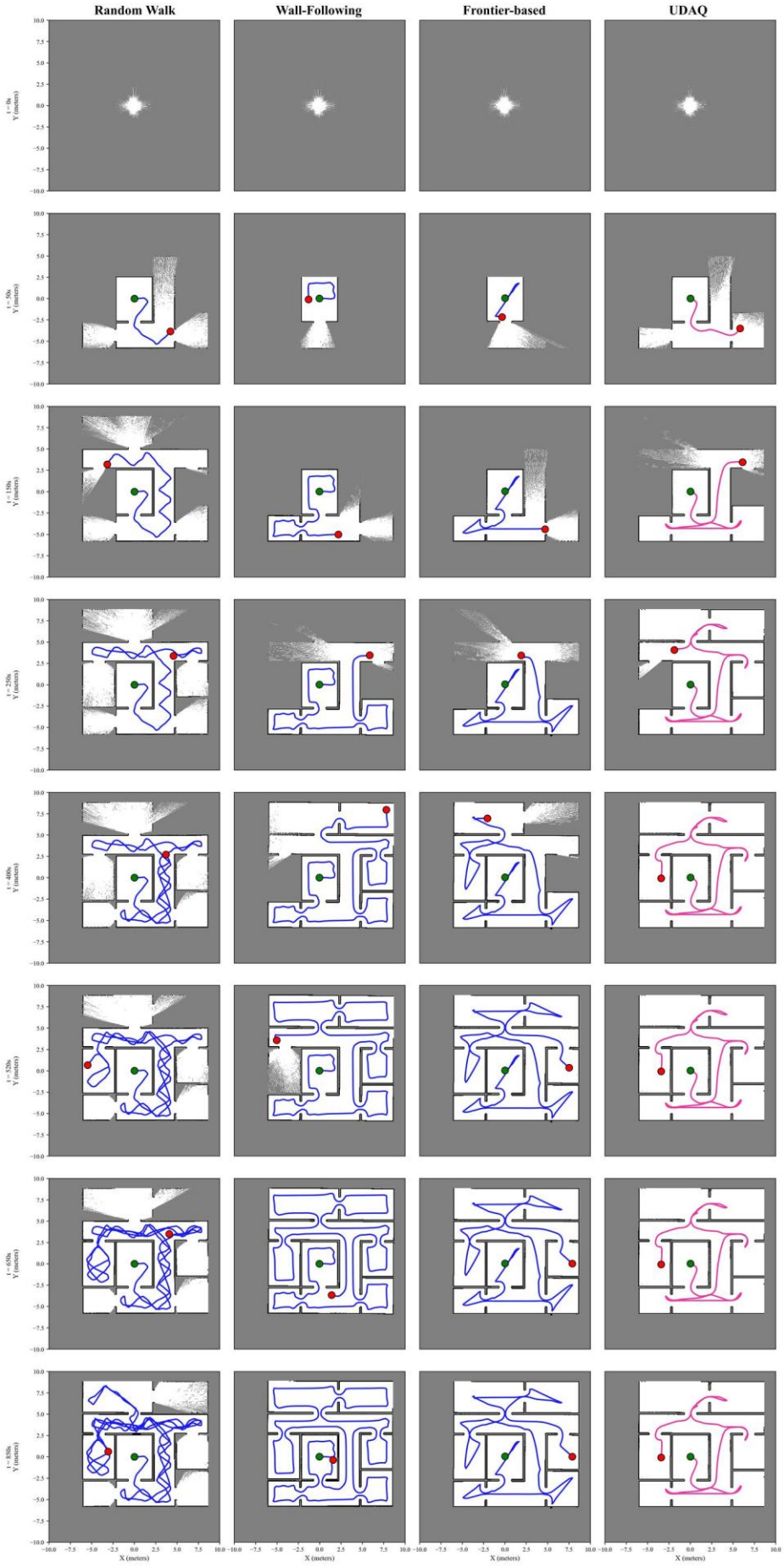


Fig. 12. Map C: exploration progress snapshots for all methods, showing trajectory overlays at multiple timestamps.

Fig. 13 presents the quantitative analysis of travel distance and exploration index (EI) over time for all four methods in Map C, a complex environment with multiple rooms and narrow corridors.

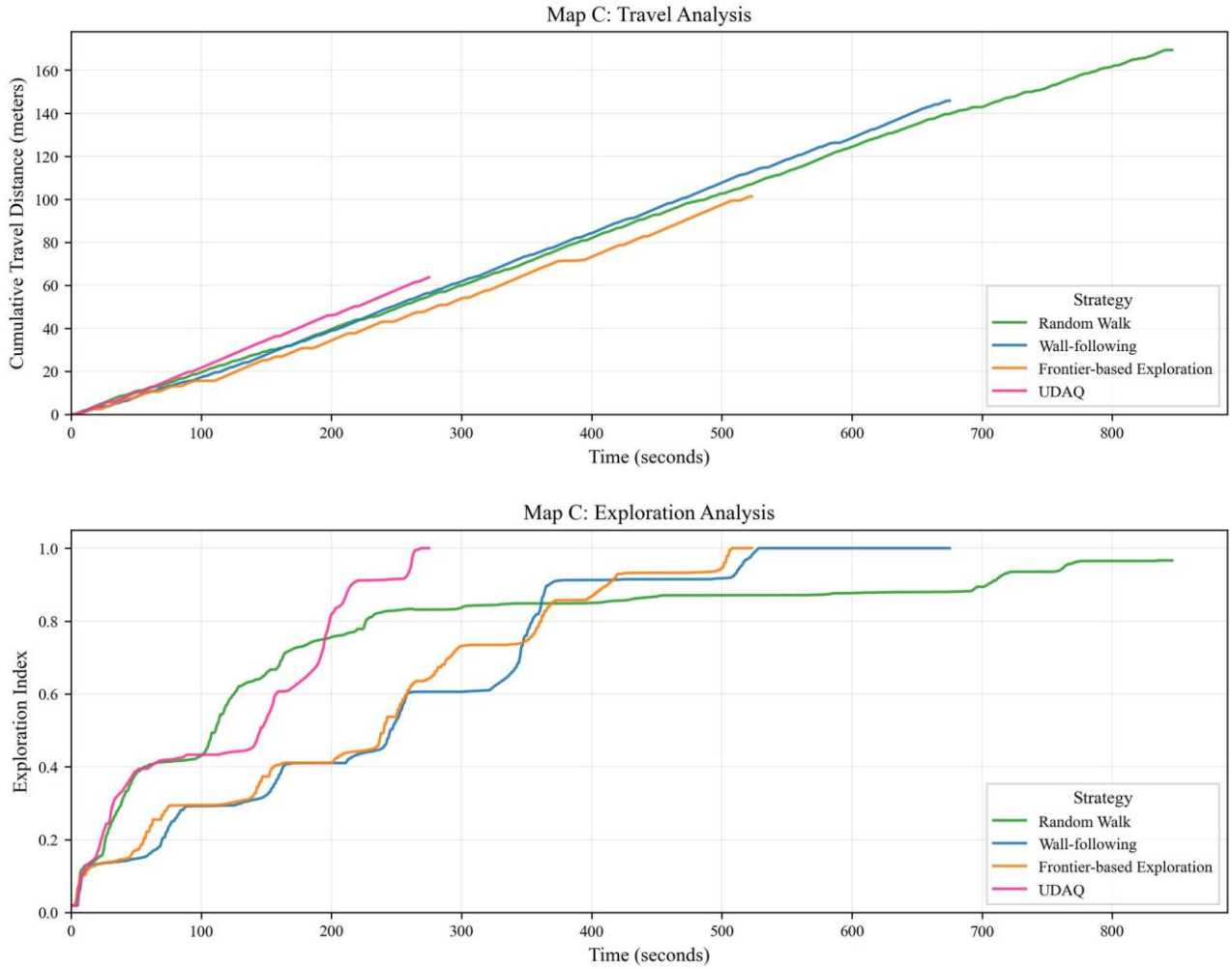


Fig. 13. Map C: travel and exploration analysis over time for all methods.

Travel Analysis: Random Walk traveled approximately 170 m in 845 s but failed to achieve full exploration. Wall-Following covered about 145 m in 670 s, efficiently using the dense wall structure. Frontier-Based exploration completed the task in 100 m over 520 s, progressing linearly with pronounced staircase effects due to increased frontier decision-making complexity. UDAQ completed exploration in 60 m over 270 s, demonstrating the most efficient path utilization despite the complex environment.

Exploration Analysis: Random Walk exhibited a large flat region from 220–700 s, failing to reach complete coverage (EI < 1.0) due to confinement in narrow corridors. Wall-Following showed multiple flat regions, producing a staircase-like progression as the robot navigated confined spaces and transitions between rooms, ultimately reaching full coverage by 520 s but continuing until 670 s. Frontier-Based exploration maintained steady progress, completing full coverage by 520 s. UDAQ displayed the fastest and steepest exploration curve, with only a small flat region between 50–150 s caused by revisiting previously scanned areas, indicating effective adaptation to the complex nested layout.

Table 4 summarizes exploration performance in Map C.

Table 4

Map C: summary of travel distances, durations, and exploration efficiency for all methods.

Strategy	Travel Distance (m)	Duration (s)	Final Exploration Index (EI)	Exploration Rate (EI/s)
Random walk	169.34	845	0.94	0.11%
Wall-following	145.80	674	1.0	0.14%
Frontier-based	101.36	522	1.0	0.19%
UDAQ	63.69	274	1.0	0.36%

Random Walk failed to achieve full coverage (EI = 0.94), traveling 169.34 m in 845 s and becoming trapped in confined areas. Wall-Following completed exploration in 145.80 m over 674 s (0.14 EI/s), benefiting from wall guidance but remaining inefficient. Frontier-Based exploration achieved full coverage in 101.36 m and 522 s (0.19 EI/s), with increased decision-making overhead. UDAQ completed the task in 63.69 m over 274 s (0.36 EI/s), demonstrating the fastest and most efficient exploration. Relative to Frontier-Based, UDAQ reduced travel distance by ~37% and completion time by ~48%; reductions relative to Wall-Following exceeded 55% and 58%, respectively. These results highlight UDAQ's robustness and suitability for complex structured environments.

4.3. Experiment 3: Oriented Exploration

The third experiment evaluated the oriented exploration capability of UDAQ by testing eight directional preference configurations in Map C, which consists of seven distinct rooms labeled “a” through “g” Fig. 14. The robot starts in room “d”. The objective was to examine how directional bias influences the order of room discovery, travel efficiency, and overall exploration performance in a complex structured environment.

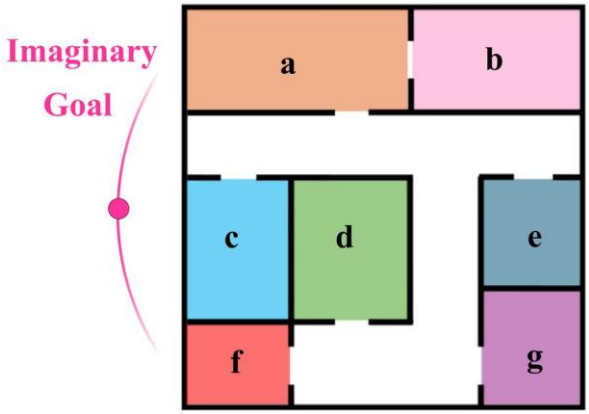


Fig. 14. Map C layout with labeled rooms and example of imaginary goal configured towards west.

The results reveal that directional preference has a significant impact on exploration efficiency. The worst-performing configuration was West, which produced the room visitation order d→f→c→a→b→e→g. This sequence caused repeated traversal of previously explored corridors Fig. 15, leading to increased path redundancy, longer travel distance, and extended exploration time. The inefficient ordering reflects poor alignment between the imposed directional bias and the underlying room-corridor topology. In contrast, the South configuration achieved the most efficient exploration, following the visitation order d→f→g→e→c→a→b. This ordering minimized backtracking and promoted a largely monotonic expansion of explored regions, resulting in the shortest travel distance (57.98 m) and lowest completion time (254 s). Directional preferences such as Southwest and Southeast also performed well, indicating that orientations aligned with major corridor directions yield more efficient exploration behavior. Intermediate configurations (East, North, Northeast, and Northwest) exhibited comparable performance, with moderate travel distances and durations. These results suggest that while oriented exploration does not guarantee optimality, it provides a controllable

mechanism to bias exploration toward structurally favorable directions, thereby influencing exploration order and efficiency.

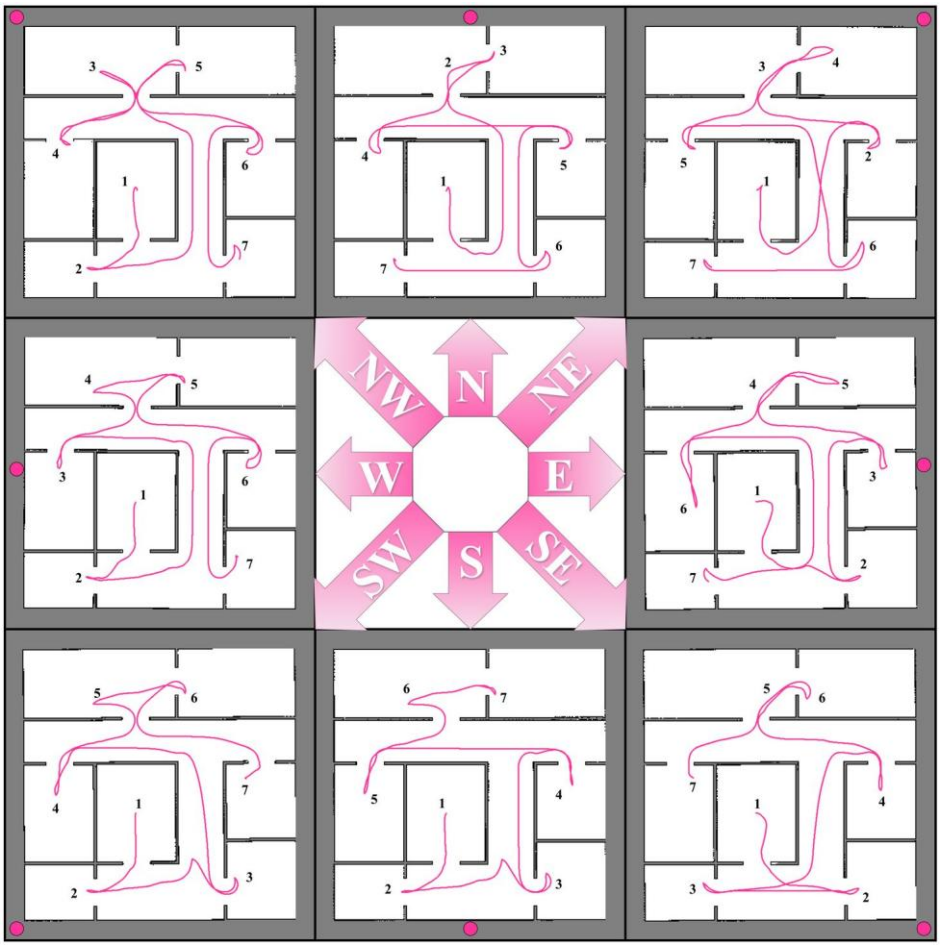


Fig. 15. Oriented exploration with eight direction configurations.

Table 5 list several configurations with their travel distance and travel duration results. Overall, the experiment demonstrates that oriented exploration can significantly affect exploration outcomes in structured environments. When the directional bias aligns with the dominant spatial layout, UDAQ can reduce redundant traversal and accelerate exploration. Conversely, poorly aligned orientations may increase path redundancy, highlighting the importance of informed or adaptive direction selection in practical deployments.

Table 5

Oriented exploration configuration tests.

Configuration	Travel Distance (m)	Duration (s)
East	73.46	314
Northeast	65.69	294
North	69.41	304
Northwest	69.53	309
West	66.81	297
Southwest	62.59	272
South	57.98	254
Southeast	60.71	273

5. Conclusion

This paper introduced UDAQ, a Unified Dynamic and Adaptive Q-iteration framework for autonomous navigation and exploration in unknown environments. The primary contribution of this work is the demonstration that multiple autonomous behaviors, including point-to-point path planning in unknown environments, full environment exploration, and oriented exploration, can be achieved within a unified reinforcement learning formulation in which behavior emerges from reward configuration rather than from planner switching or task-specific heuristics. A key aspect of UDAQ is its ability to operate under dynamically evolving state spaces, as encountered in SLAM-based mapping, where the environment expands and changes over time. To ensure stable convergence under these conditions, an adaptive reward scaling mechanism is introduced, enabling effective value propagation as the state space grows and preventing degradation of policy quality. UDAQ operates as a fully autonomous closed-loop system, continuously integrating SLAM updates, updating its state representation and reward parameters, performing Q-iteration to convergence, and generating navigation commands without human intervention.

Through extensive evaluation, UDAQ demonstrated superior performance in both navigation and exploration tasks. In point-to-point navigation, it consistently generated shorter and more direct paths than the ROS 2 NavFn planner, reducing path length by approximately 8–10% and travel duration by

20–30% across all tested start-goal pairs, approaching global optimal solutions despite operating without prior map knowledge. In exploration tasks, UDAQ outperformed frontier-based exploration, the classical baseline, across open, structured, and complex environments. Compared to frontier-based exploration, UDAQ reduced travel distance by approximately 36–44% and exploration time by 48–59%, while also substantially exceeding the performance of simpler reactive methods such as wall-following and random walk. The oriented exploration experiment further demonstrated UDAQ's adaptability, showing that directional bias can meaningfully influence exploration efficiency and room visitation order. Optimal directional configurations reduced redundant traversal and minimized exploration time, highlighting the potential of orientation-aware strategies in structured environments.

Despite these promising results, several limitations should be acknowledged. All experiments were conducted within ROS 2 simulation environments, which provide a controlled and highly reproducible setting for algorithm evaluation but do not fully capture the uncertainties encountered in real-world robotic systems. The simulations assume idealized sensing and state estimation conditions and do not explicitly model factors such as LiDAR measurement noise, wheel slip, actuator inaccuracies, communication delays, or localization drift. Consequently, while the results demonstrate the effectiveness of the proposed framework under controlled conditions, additional validation on physical robotic platforms is required to assess robustness under real-world operating conditions. Future work will focus on hardware deployment, incorporation of realistic sensor and actuator uncertainty models, and evaluation in larger and more dynamic environments.

Overall, these results confirm that UDAQ provides a robust, efficient, and flexible solution for autonomous navigation in unknown environments, unifying multiple navigation behaviors within a single reward-driven framework and operating entirely as a fully autonomous closed-loop system.

Acknowledgments

This work was supported in part by the Fundamental Research Grant Scheme (FRGS) through the Ministry of Higher Education of Malaysia (MOHE) under Grant FRGS/1/2022/TK08/UNIMAP/03/13, and in part by the Malaysian Technical University Network (MTUN) Research Grant by MOHE under Grant 9002-00094/9028-00002.

References

[1] S. Nahavandi *et al.*, "A Comprehensive Review on Autonomous Navigation," *ACM Comput. Surv.*, vol. 57, no. 9, pp. 1–67, Sep. 2025, doi: 10.1145/3727642.

[2] J. A. Placed *et al.*, "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1686–1705, Jun. 2023, doi: 10.1109/TRO.2023.3248510.

[3] D. Brugali, L. Muratore, and A. De Luca, "Mobile robots exploration strategies and requirements: A systematic mapping study," *Int. J. Rob. Res.*, vol. 44, no. 9, pp. 1461–1506, Aug. 2025, doi: 10.1177/02783649241313471.

[4] M. Tellaroli, M. Luperto, M. Antonazzi, and N. Basilico, "Frontier-Based Exploration for Multi-Robot Rendezvous in Communication-Restricted Unknown Environments," Jul. 2024, doi: https://doi.org/10.48550/arXiv.2403.11617.

[5] K. Cai, C. Wang, J. Cheng, C. W. De Silva, and M. Q.-H. Meng, "Mobile Robot Path Planning in Dynamic Environments: A Survey," Mar. 2021, doi: 10.15878/j.cnki.instrumentation.2019.02.010.

[6] K. Katona, H. A. Neamah, and P. Korondi, "Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot," *Sensors*, vol. 24, no. 11, p. 3573, Jun. 2024, doi: 10.3390/s24113573.

[7] C. Ntakolia, S. Moustakidis, and A. Siouras, "Autonomous path planning with obstacle avoidance for smart assistive systems," *Expert Syst. Appl.*, vol. 213, p. 119049, Mar. 2023, doi: 10.1016/j.eswa.2022.119049.

[8] H. Zhang, Y. Zhang, C. Liu, and Z. Zhang, "Energy efficient path planning for autonomous ground vehicles with ackermann steering," *Rob. Auton. Syst.*, vol. 162, p. 104366, Apr. 2023, doi: 10.1016/j.robot.2023.104366.

[9] X. Yue, Y. Zhang, and M. He, "LiDAR-based SLAM for robotic mapping: state of the art and new frontiers," Nov. 2023, [Online]. Available: http://arxiv.org/abs/2311.00276.

[10] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. "Towards New Computational Principles for Robotics and Automation,"* pp. 146–151, doi: 10.1109/CIRA.1997.613851.

[11] C. Liu *et al.*, "Enhancing autonomous exploration for robotics via real time map optimization and improved frontier costs," *Sci. Rep.*, vol. 15, no. 1, p. 12261, Apr. 2025, doi: 10.1038/s41598-025-97231-9.

[12] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968, doi: 10.1109/TSSC.1968.300136.

[13] A. Khanal, H.-D. Bui, G. J. Stein, and E. Plaku, "Guided Sampling-Based Motion Planning with Dynamics in Unknown Environments," Jun. 2023, doi: 10.48550/arXiv.2306.09229.

[14] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1396–1402, doi: 10.1109/IROS.2017.8202319.

[15] A. Feng, Y. Xie, Y. Sun, X. Wang, B. Jiang, and J. Xiao, "Efficient Autonomous Exploration and Mapping in Unknown Environments," *Sensors*, vol. 23, no. 10, p. 4766, May 2023, doi: 10.3390/s23104766.

[16] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1–4, pp. 403–430, Nov. 1987, doi: 10.1007/BF01840369.

[17] S. Al Mahmud, A. Kamarulariffin, A. M. Ibrahim, and A. J. H. Mohideen, "Advancements and Challenges in Mobile Robot Navigation: A Comprehensive Review of Algorithms and Potential for Self-Learning Approaches," *J. Intell. Robot. Syst.*, vol. 110, no. 3, p. 120, Aug. 2024, doi: 10.1007/s10846-024-02149-5.

[18] A. Jamshidpey and H. H.-T. Liu, “Multi-UAV Uniform Sweep Coverage in Unknown Environments: A Self-organizing Nervous System (SoNS)-Based Random Exploration,” Mar. 2025, [Online]. Available: http://arxiv.org/abs/2409.11116.

[19] M. Prágr, J. Bayer, and J. Faigl, “Autonomous robotic exploration with simultaneous environment and traversability models learning,” *Front. Robot. AI*, vol. 9, Oct. 2022, doi: 10.3389/frobt.2022.910113.

[20] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild,” *Sci. Robot.*, vol. 7, no. 66, May 2022, doi: 10.1126/scirobotics.abm6074.

[21] S. Macenski, F. Martin, R. White, and J. G. Clavero, “The Marathon 2: A Navigation System,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 2718–2725, doi: 10.1109/IROS45743.2020.9341207.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.

[23] Y.-J. Chen, B.-G. Jhong, and M.-Y. Chen, “A Real-Time Path Planning Algorithm Based on the Markov Decision Process in a Dynamic Environment for Wheeled Mobile Robots,” *Actuators*, vol. 12, no. 4, p. 166, Apr. 2023, doi: 10.3390/act12040166.

[24] B. Nie, Y. Gao, Y. Mei, and F. Gao, “Capability Iteration Network for Robot Path Planning,” Apr. 2021, doi: 10.2316/J.2021.206-0598.

[25] B. Hao, H. Du, J. Zhao, J. Zhang, and Q. Wang, “A Path-Planning Approach Based on Potential and Dynamic Q-Learning for Mobile Robots in Unknown Environment,” *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–12, Jun. 2022, doi: 10.1155/2022/2540546.

[26] N. Botteghi, B. Sirmacek, K. A. A. Mustafa, M. Poel, and S. Stramigioli, “On Reward Shaping for Mobile Robot Navigation: A Reinforcement Learning and SLAM Based Approach,” Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.04109.