Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079895

ARTICLE

An HRMCTS-Based Optimization Method for Efficient Multi-Objective Path Planning

Qianshu Yang, Shuangxi Liu*, Xianyu Wu and Wei Zhao

Advanced Propulsion Technology Laboratory, National University of Defense Technology, Changsha, China

*Corresponding Author: Shuangxi Liu. Email: lsxdouble@163.com

Received: 30 January 2026; Accepted: 16 April 2026; Published: 08 May 2026

ABSTRACT: Path planning for unmanned systems in complex environments must simultaneously satisfy safety, kinematic feasibility, and real-time performance requirements. Monte Carlo Tree Search (MCTS) offers advantages such as model-free operation, strong interpretability, and anytime planning capability, but it suffers from large branching factors, excessive search depths, and poor convergence under sparse reward conditions in high-dimensional state spaces. To address these challenges, this paper proposes a Heuristic Rolling Monte Carlo Tree Search (HRMCTS) framework. First, the path planning problem is formulated as a constrained Markov decision process, where the state consists of position and heading, and actions are discretized heading changes. Second, a heuristic selection strategy incorporating goal-directed guidance and obstacle safety margins is introduced to improve search directionality, while a limited-depth forward simulation with branch pruning is employed during the rollout phase. A multi-objective reward function is designed to integrate distance, goal progress, tangent-based obstacle avoidance, smoothness, and efficiency, thereby jointly optimizing path quality and computational performance. Experiments are conducted in three scenarios: static polygonal environments, dynamic circular obstacle environments, and dynamic polygonal obstacle environments. Simulation results demonstrate that the proposed method offers significant advantages in terms of planning efficiency, environmental adaptability, generalization capability, and interpretability.

KEYWORDS: Unmanned system; path planning; MCTS; heuristic algorithm; reward shaping

1 Introduction

Unmanned systems, such as unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs), autonomous underwater vehicles (AUVs), and unmanned surface vehicles (USVs), have found extensive applications across various domains. In recent years, the path planning issues concerning these unmanned systems have garnered significant research attention [1–5]. In this work, UAV is selected as the case study to examine path planning, which has emerged as a central and challenging research focus in the field of UAV autonomy.

The objective of path planning is to generate an optimal trajectory from starting point to goal point, ensuring both safety and dynamic feasibility. Recent advancements underscore a significant transition from traditional shortest path optimization to a multidimensional equilibrium that encompasses energy efficiency, risk mitigation, and temporal constraints. This shift is particularly vital in high-dynamic and stringently constrained environments, such as autonomous flight through dense obstacle fields or penetration missions in uncertain no-fly zones, where numerous complex and irregularly distributed obstacles render real-time planning particularly challenging. In such scenarios, a high-quality path planner must not only generate

collision-free geometric paths but also rigorously respect physical constraints, such as nonholonomic kinematics, maximum angular velocity, and speed limits, while processing vast state spaces within decision windows. This necessity has driven the development of advanced path planning algorithms that tightly integrate environmental perception, kinematic constraints, and online decision-making capabilities, establishing it as a central research focus in the UAV domain [6,7].

Existing path planning methodologies are generally categorized into grid-based, sampling-based, and optimization-based paradigms. Within these groups, heuristic optimization algorithms like particle swarm optimization (PSO) have gained significant traction due to their proficiency in global searching and effective constraint handling in complex, dynamic environments. Such capabilities provide a clear advantage over deterministic or exhaustive search techniques [8–10]. Deep reinforcement learning (DRL) has also proven highly effective by utilizing deep neural networks to optimize policies within high-dimensional, continuous state spaces [11,12]. However, the DRL decision-making process is often contained within learned network parameters. This reliance on automatic feature extraction lacks transparent reasoning steps, which makes it difficult to map state observations to final actions and potentially compromises the reliability of agent behavior [13]. Consequently, research into RL explainability seeks to clarify how state information influences decisions to foster more transparent and trustworthy policies. Conversely, Monte Carlo Tree Search (MCTS) provides an inherently interpretable framework for decision-making. In this search tree, nodes represent concrete states while branches correspond to feasible actions. Statistics such as visit counts and simulation results directly quantify the value of each action [14]. This explicit structure facilitates human-understandable reasoning because inspecting high-value paths allows one to reconstruct the rationale for a chosen action and clearly state the intent of every planning step.

Common approaches to path planning in dynamic environments include MCTS, A* and its hybrid variants, sampling methods, and DRL. MCTS enhances computational efficiency and reduces collision rates through its capacity for real-time decision-making. These benefits are particularly evident when the algorithm incorporates velocity obstacle techniques or adaptive action sets within dynamic environments [15]. Recent research has also emphasized modeling obstacle persistence to facilitate robust multi-agent replanning [16]. Furthermore, extending MCTS for long-term horizons assists in overcoming the complexities associated with high-dimensional action spaces [17]. In structured settings, A* and its hybrids are favored for their low latency and ability to generate smooth trajectories. These methods utilize pre-training to prune unnecessary nodes, which makes them highly effective for rapid response in constrained environments [18]. Sampling methods maintain probabilistic completeness and optimality while improving computational speed. These algorithms often employ temporal reasoning and pruning strategies to achieve further optimization [19]. While DRL displays significant adaptability after training, its inherent sample inefficiency often limits real-time implementation. Techniques such as prioritized experience replay and attention mechanisms are therefore required to accelerate policy convergence [17,20]. Ultimately, MCTS serves as a compelling foundation for efficient dynamic path planning due to its balance of adaptability, speed, and interpretability. It avoids the black-box nature of DRL, the potential lack of optimization guarantees in sampling methods, and the structural constraints inherent to A*.

Building upon the advancements in single-agent navigation, the focus of recent scholarship has increasingly pivoted toward the complexities of multi-agent collaborative planning. A primary concern in this domain is the deployment of heterogeneous Amphibious Unmanned Surface Vehicles (AUSV) under stringent energy budgets. To address this, He et al. [21] utilized swarm-based coordination to optimize coverage efficiency, while Li et al. [22] introduced dynamic path-adjustment models grounded in real-time energy consumption telemetry.

The challenge of heterogeneity extends to USV-UAV synergistic operations, where Huang et al. [23] demonstrated that incorporating hull dynamics into the planning phase significantly enhances target search efficacy. Such robustness is equally critical in chaotic underwater environments. For instance, in scenarios characterized by 3D terrain and volatile currents, multi-objective frameworks are now frequently employed to navigate the inherent trade-offs between path safety and mission timeliness [24]. Recent trends, as exemplified by Sun and Lv [25], favor a hybridized approach: integrating PSO for global strategic mapping with the Dynamic Window Algorithm (DWA) for tactical obstacle avoidance. This dual-layer architecture, often complemented by evolutionary algorithms to identify Pareto-optimal solutions, ensures that conflicting metrics, such as communication quality and energy overhead, are systematically balanced. Furthermore, in communication-sparse zones, distributed frameworks based on consensus or leader-follower theory continue to serve as the benchmark for maintaining formation stability [26].

MCTS frames path planning as a sequential decision-making problem within the context of a Markov Decision Process (MDP). Although traditional greedy heuristics often stall in local optima during long-horizon missions, MCTS maintains a robust balance between exploration and exploitation. It achieves this equilibrium by incrementally building a search tree through selective simulations. This mechanism offers a clear advantage in interpretability. Rather than functioning as an opaque black box, MCTS populates its nodes with explicit state statistics. Data such as visitation frequency and cumulative rewards provide a transparent record of the perceived utility of each action. From an operational perspective, the anytime property is particularly valuable in dynamic environments. This feature allows the immediate retrieval of a valid solution once the computational budget is exhausted. However, applying these methods to high-dimensional continuous spaces reveals significant bottlenecks. The combination of exponential branching and sparse reward signals can severely effect convergence rates. To mitigate these structural constraints, researchers have proposed various enhanced MCTS variants to improve sampling efficiency and search depth [14,27,28].

Efforts to augment MCTS generally converge on three critical fronts: narrowing the action space, optimizing search depth, and leveraging structural or hardware acceleration. When confronted with the "curse of dimensionality" and excessive branching factors, the literature has moved toward more selective growth strategies. These include progressive expansion [29], dominance-based pruning [30–32], and the use of macro-actions or "options" to abstract long-range decision sequences [33,34]. Interestingly, even Large Language Models have recently been tapped to evolve the heuristic functions that steer these searches [35]. To refine simulation efficiency and depth, contemporary research has pivoted toward more sophisticated selection policies—utilizing weighted UCB algorithm [36] or value gradients [37], while employing cost-approximations for early simulation termination [38–41]. This focus on resource allocation is further evident in the adoption of nested structures [42–44] and neural-network-based value proxies [45,46], both of which concentrate computational effort on the most promising sub-trees. Beyond algorithmic refinements, the integration of transposition tables [47] and information sets [48] has streamlined the underlying search structure. When these are hybridized with deep learning [45,49] or evolutionary meta-heuristics [50,51], and further backed by hardware-level parallelization [52–56], MCTS becomes significantly more viable for the intricate demands of complex, real-world environments.

Modern path planning for unmanned systems has undergone a fundamental shift, moving beyond simple shortest-path heuristics toward a complex multi-objective equilibrium that balances energy conservation, risk mitigation, and strict temporal deadlines. This transition is especially critical in highly dynamic or adversarial environments, such as autonomous navigation through dense obstacle fields or penetration missions within contested no-fly zones. In these settings, the primary challenge for any planning algorithm

is the need to reconcile high-fidelity kinematic constraints with the necessity of navigating vast state spaces within millisecond-level decision windows.

While MCTS offers a theoretically sound foundation for such problems, standard implementations and common enhancements (e.g., action pruning or parallelization) often fail when tasked with maintaining solution quality under extreme computational and kinematic pressure. There remains a pressing need for MCTS architectures that provide both the scalability and search intensity required for the next generation of intelligent systems. To this end, this work introduces a Heuristic Rolling Monte Carlo Tree Search (HRMCTS) framework. By synthesizing a receding-horizon logic with targeted heuristic guidance, HRMCTS significantly accelerates search convergence while preserving the algorithm's inherent transparency. Although the present study centers on the high-fidelity navigation of a single agent, the core pillars of HRMCTS specifically its localized planning strategy, multi-objective reward design, and heuristic-informed evaluation, like establish a robust computational template for future extensions into heterogeneous multi-agent coordination and energy-constrained missions. The primary contributions of this research are structured as follows:

(1) We introduce a heuristic upper confidence bound for trees (UCT) selection and rollout strategy that embeds goal-directed guidance and obstacle safety margins directly into the search process. By mitigating the raw stochastic exploration and high branching factors inherent in traditional MCTS, this approach significantly sharpens search directionality and minimizes redundant computational overhead.

(2) To overcome the challenges of excessive search depth and dynamic threats, this work develops a receding-horizon planning mechanism characterized by limited-depth forward simulation and strategic branch pruning. The resulting closed-loop "plan–execute–replan" architecture effectively reconciles the tension between immediate real-time responsiveness and global path feasibility within complex, dynamic environments.

(3) A multifaceted reward function is formulated to unify disparate metrics including goal proximity, progress, and trajectory smoothness with predictive dynamic obstacle avoidance. This integrated design facilitates the simultaneous optimization of path quality and efficiency, notably accelerating MCTS convergence in scenarios typically degenerated by sparse reward signals.

2 Background

2.1 MDP

MDP is a probabilistic modeling framework for sequential decision-making, formally characterizing how an agent selects actions in a stochastic environment to maximize cumulative long-term reward. Rather than being a specific problem-solving algorithm, MDP constitutes a general theoretical framework that bridges dynamic system modeling with globally optimal decision-making. It abstracts complex decision problems into a cyclic structure of "*state–action–transition–reward*", quantifying environmental uncertainty through a probabilistic transition function while explicitly defining a long-term optimization objective. This formulation provides a unified mathematical representation and optimization logic for sequential decision problems such as path planning [13]. The core assumption of an MDP is the Markov property (memorylessness): the evolution of future states depends solely on the current state and action, independent of the historical trajectory. In path planning, this implies that the agent need not retain its full motion history; instead, only essential information—such as current position, environmental constraints (e.g., obstacle distribution), and task progress—is sufficient for effective future path decisions. This abstraction reduces state-space complexity while preserving decision efficacy, forming the foundation for MDP's applicability across diverse dynamic scenarios [14].

An MDP is formally defined by a tuple (S, A, P, R), often extended to a quintuple with a discount factor γ. In the context of path planning, these components are interpreted as follows:

1. State space S: The set of all possible system configurations, commonly represented as coordinate pairs (x, y) in 2D planning scenarios.
2. Action space A: The set of admissible actions $A_s \subseteq A$ available at state s. While low-branching cases may utilize atomic moves, higher-dimensional problems often rely on action sampling or macro-actions to manage computational complexity.
3. Transition function P: The probabilistic mapping $P: S \times A \times S \to [0,1]$ that defines environmental dynamics. Although P facilitates modeling stochasticity, many path planning implementations treat these transitions as deterministic.
4. Reward function R: The mapping $R: S \times A \times S \to \mathbb{R}$ that formalizes the optimization objective. In this paper, R must reconcile mission goals with state and path constraints to dictate trajectory quality as a requirement that aligns with RL paradigms where reward density and scale are paramount for convergence [4].

As a foundational framework for sequential decision-making, MDP provides a abstract modeling paradigm, serving as both the theoretical basis and problem description language for RL and MCTS.

2.2 MCTS

MCTS optimizes decision-making by merging stochastic simulation with structured tree exploration. The algorithm dynamically constructs a search tree to reconcile the tension between exploring unvisited states and exploiting known high-reward trajectories. While MCTS shares a formal problem structure with RL and their conceptual intersections are well-documented [14], this study characterizes MCTS as a heuristic search algorithm to ensure clarity. By employing stochastic sampling in lieu of exhaustive enumeration, MCTS effectively addresses the complexities of state spaces characterized by high branching factors and significant depths. As depicted in Fig. 1, the algorithm approximates the value of specific states through a cyclic four-phase process consisting of selection, expansion, evaluation, and backpropagation. During the selection phase, the agent traverses the tree from the root to a leaf node s_L based on accumulated statistics. The expansion phase then appends new child nodes to s_L to represent potential actions. In the evaluation phase, the utility of these new states is estimated through simulation or statistical aggregation. Finally, backpropagation updates the statistics of all ancestral nodes along the path to the root. This iterative refinement allows MCTS to identify optimal policies in environments where conventional value or policy iteration remains computationally prohibitive [30].

As discussed above, the core of MCTS lies in the selection and rollout phases. By refining the algorithms in these two phases, MCTS variants tailored to specific problems can be developed. The goal of the selection phase is to traverse from the root node down to an expandable leaf node along the branch with the highest exploration value. Its essence is to avoid blind exploration and prioritize actions that yield high information gain. A commonly used selection policy is the UCT formula [57]:

$$a^* = \arg\max_{a \in A(s)} \left\{ \frac{W(s,a)}{N(s,a)} + C \cdot \sqrt{\frac{\ln N(s)}{N(s,a)}} \right\} \tag{1}$$

where $Q(s,a)$ represents the reward for executing action a in state s, while $N(s,a)$ and $N(s)$ track the selection frequency of specific actions and the total visitation count for the state, respectively. The first component of the expression prioritizes the exploitation of historically favorable actions. Specifically, the ratio $\frac{W(s,a)}{N(s,a)}$ serves as an empirical approximation of the expected reward $Q(s,a)$. This standard approach

encounters two significant bottlenecks in long-horizon path planning characterized by sparse rewards and high-dimensional spaces. A primary issue is that meaningful reward signals typically only emerge upon goal attainment or collision. Because intermediate steps yield nearly uniform rewards, the exploitation term struggles to distinguish node quality during the early stages of the search. Furthermore, the exploration term relies exclusively on visitation statistics and lacks awareness of the goal's direction. In the absence of goal-oriented guidance, MCTS often requires an excessive number of iterations to locate the target. This leads to the inefficient expenditure of computational resources on irrelevant regions and necessitates a modified heuristic to improve search efficiency. The objective of the rollout phase is to approximate the value of newly expanded nodes by simulating trajectories toward terminal states using a default policy. By substituting exact evaluations with stochastic sampling, this process significantly accelerates the iteration cycle [58]. However, this approach faces two fundamental challenges. Increasing simulation fidelity introduces substantial computational overhead, which is a problem magnified by the vast search spaces inherent in path planning. Additionally, the extreme depth required to reach terminal states often renders purely random simulations computationally prohibitive. These constraints require a refined rollout policy that can effectively reconcile evaluation accuracy with limited computational budgets.

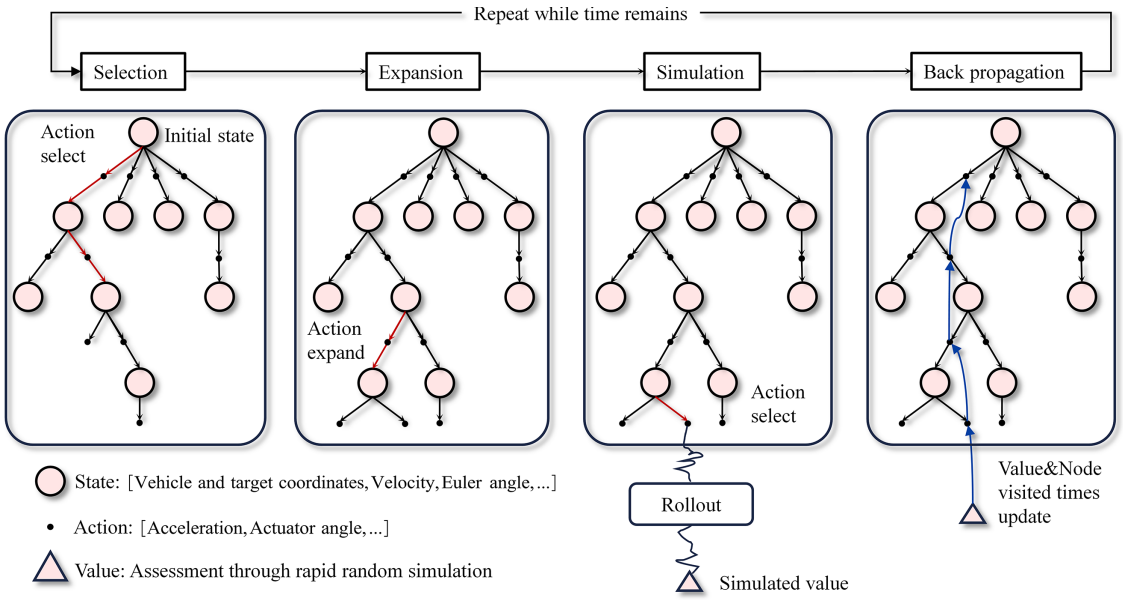


Figure 1: Description of MCTS process.

Remark 1: *The interpretability of MCTS is rooted in the structural clarity of its search tree and the transparency of its underlying node-level statistics. Unlike black-box models that rely on opaque computations, MCTS establishes decisions through the incremental expansion of a visible search space. The individual components of Eq. (1) clarify exactly why a specific action is prioritized. These terms provide a clear distinction between an action's historical performance and its exploration potential. Furthermore, each node preserves essential metrics such as visitation frequency and cumulative rewards. This organizational structure ensures that the entire decision chain remains traceable. By backtracking from the selected node to the root, the specific logic informing the final output can be fully audited and analyzed [59].*

3 Methods

Assumption 1: *The unmanned system in this paper is modeled as a point mass, denoted as an agent.*

In this scenario, the agent must autonomously plan an optimal path from a given initial state to a target region, subject to its own kinematic constraints, environmental boundary constraints, and obstacle avoidance requirements, while simultaneously optimizing multiple objectives: path smoothness, efficiency, and safety. The core challenge lies in balancing exploration and exploitation under dynamically distributed obstacles and limited computational resources, ensuring that the planned path remains feasible in complex environments populated by polygonal obstacles and maximizes cumulative reward. This work achieves multi-objective path optimization, encompassing safe obstacle avoidance, path efficiency, and motion smoothness, through the synergistic integration of a quantified reward mechanism and a heuristic UCT strategy.

3.1 Path Planning Problem in MDP Modeling

3.1.1 State Definition

By default, the MCTS algorithm assumes discrete states and actions with finite cardinality. The instantaneous state vector of the agent is defined as:

$$\boldsymbol{s}(t) = \left[\boldsymbol{p}(t), \theta(t)\right]^T \tag{2}$$

where $\boldsymbol{p}(t) = \left[x(t), y(t)\right]^T$ denotes the position vector of the agentcenter at time t, where $x(t)$ and $y(t)$ represent the coordinates along the horizontal and vertical axes, respectively; $\theta(t)$ is the heading angle of the agent at time t, measured counterclockwise from the positive x-axis with a range of $[0, 2\pi)$. This state vector comprises two core dimensions—position and heading.

3.1.2 Dynamic Model

In order to reduce unnecessary complexity, the agent adopts a kinematic model of constant speed v with discrete heading adjustments. The action is defined as the heading angle offset $\Delta\theta(t)$, which corresponds to the angular velocity in continuous control. The discrete-time update rule over a time step Δt is as follows:

$$\begin{cases} \boldsymbol{p}(t+\Delta t) &= \boldsymbol{p}(t) + v \cdot \Delta t \cdot \left[\cos\theta(t+\Delta t), \sin\theta(t+\Delta t)\right]^T \\ \theta(t+\Delta t) &= \theta(t) + \Delta\theta(t) \end{cases} \tag{3}$$

3.1.3 Constraints

Path planning for the agent is governed by four primary constraint categories. These include environmental boundaries, obstacle collisions, action space limitations, and the specific requirements of the MCTS search process. It is illustrated within a predefined two-dimensional rectangular map $\Omega = [0, W] \times [0, H]$, where W and H represent the map width and height. To account for the agent's physical dimensions, a safety radius r_{agent} is utilized. This radius incorporates both the agent's body size and sensor uncertainty. Consequently, the agent must remain entirely within the map boundaries throughout the mission. The positional constraint for all $t \in [0, T]$ is defined as:

$$\begin{cases} r_{\text{agent}} + \delta \le x_k \le W - (r_{\text{agent}} + \delta) \\ r_{\text{agent}} + \delta \le y_k \le H - (r_{\text{agent}} + \delta) \end{cases} \tag{4}$$

where T denotes the termination time of the trajectory, and δ represents an additional safety margin. If the position exceeds the boundary, it is constrained to the valid range via a clamp operation.

The environment is modeled with a set of M static obstacles, denoted by $\mathcal{O} = \{O_1, O_2, \ldots, O_M\}$. Each obstacle O_j is defined as a closed polygon comprising N_j vertices, where the k-th vertex is represented as $\boldsymbol{p}_{\text{obs},j,k} = [x_{\text{obs},j,k}, y_{\text{obs},j,k}]^T$. These vertices are ordered to form a contiguous boundary. To ensure navigation safety, the agent must maintain a minimum separation distance equal to its safety radius r_{agent} from any obstacle edge. This collision avoidance constraint is expressed as $d_{\min}(\boldsymbol{p}(t), O_j) \geq r_{\text{agent}}$ for all $t \in [0, T]$ and $j \in \{1, 2, \ldots, M\}$. The term $d_{\min}(\boldsymbol{p}(t), O_j)$ represents the shortest Euclidean distance from the agent's position $\boldsymbol{p}(t)$ to the boundary of polygon O_j. Furthermore, no portion of the planned trajectory may lie within an obstacle. This condition is enforced by performing segment-wise sampling between consecutive path points to verify that all samples remain in free space. The agent begins its mission with an initial heading angle θ_{init}. Subsequent heading deviations $\Delta\theta(t)$ are restricted by the vehicle's physical turning capabilities. To facilitate efficient computation, the continuous action space is discretized into a finite set $\mathcal{A}$ by uniformly sampling $N_{\Delta\theta}$ values within the allowable range $[-\Delta\theta_{\max}, \Delta\theta_{\max}]$. Search efficiency and path quality are managed through three primary MCTS constraints. Computational expenditure is capped by a maximum iteration limit $I_{\text{mcts,max}}$. Additionally, the search depth d is restricted to $D_{\text{tree,max}}$ to prevent the generation of excessively long paths that could compromise real-time performance. Finally, the search terminates once the agent reaches the goal region. This region is defined as a circle of radius r_G centered at $\boldsymbol{p}_G = [x_G, y_G]^T$. The terminal condition $\|\boldsymbol{p}(t) - \boldsymbol{p}_G\| \leq r_G$ provides the necessary planning flexibility while ensuring the agent successfully reaches the target neighborhood.

To accommodate environmental complexity, the proposed framework integrates specific constraints for dynamic obstacles based on their instantaneous velocity and projected trajectories. Unlike the treatment of static boundaries, these dynamic constraints are evaluated across a finite prediction horizon h. The algorithm employs constant-velocity extrapolation to forecast the future positions of M moving obstacles. The predicted state for any dynamic obstacle j at a subsequent time step is modeled as $\hat{p}_j(t) = p_j^0 + v_j \cdot t \cdot \Delta t$. This predictive model also incorporates boundary reflection to ensure realistic environmental dynamics. A proactive safety buffer is maintained by requiring the agent to satisfy a predictive minimum clearance $c_{\text{safe}}^{\text{pred}}$. This metric accounts for both the agent's safety radius r_{agent} and the specific radius r_j of the obstacle.

3.2 Optimization Objectives and Reward Mechanisms

3.2.1 Heuristic UCT Strategy

The selection phase of MCTS requires choosing one child node from all children of the current node for expansion, with the UCT formula serving as the core decision criterion. As previously discussed, a common variant is the UCB1 formula, which relies solely on historical statistics of the nodes, whereas heuristic UCT incorporates domain knowledge to accelerate convergence [36]. To accommodate the domain-specific characteristics of agent path planning, a heuristic bias term $H(i)$ is introduced, extending the formula to:

$$\text{UCB1}_{\text{heuristic}}(i) = \bar{V}(i) + \omega \cdot H(i) + C \cdot \sqrt{\frac{\ln N(\text{parent}(i))}{N(i)}} \tag{5}$$

where $\omega \in [0, 1]$ is the weight coefficient for heuristic information, and the heuristic function $H(i)$ is defined as:

$$H(i) = \alpha \cdot \left(1 - \frac{\left\|\boldsymbol{p}_i - \boldsymbol{p}_g\right\|}{d_{\max}}\right) + \beta \cdot \min_{j=1..M} \left(\frac{d_{\min}(\boldsymbol{p}_i, O_j) - r_{\text{agent}}}{D_{\text{safe}}}\right) \tag{6}$$

where the first term is the goal-directed bias: the closer to the goal, the higher the value, guiding the search toward the goal; the second term is the safety bias: the greater the safety margin from obstacles, the higher the value. $\alpha, \beta > 0$ are heuristic coefficients ($\alpha + \beta = 1$ to ensure $H(i) \in [0,1]$), and $d_{\max} = \sqrt{W^2 + H^2}$ denotes the maximum diagonal distance of the map. $D_{\text{safe}} = 3.0$ represents the safety distance threshold, indicating the minimum safe distance that should be maintained from the center of the agent to the boundary of obstacles.

3.2.2 Reward Definition

The core objective of path planning is to maximize the cumulative discounted reward, where the reward function integrates flight safety, path efficiency, heading smoothness, and task completion through a weighted sum. Let T denote the termination time of the trajectory, with the action sequence given by $\{\Delta\theta(0), \Delta\theta(1), \ldots, \Delta\theta(T-1)\}$ and the corresponding state sequence by $\{\boldsymbol{s}(0), \boldsymbol{s}(1), \ldots, \boldsymbol{s}(T)\}$. The cumulative reward J obtained during the rollout is then defined as:

$$J = \sum_{t=0}^{T-1} \gamma_{\text{disc}}^{t} R_{\text{step}}(\boldsymbol{s}(t), \Delta\theta(t), \Delta\theta(t-1)) + \gamma_{\text{disc}}^{T} R_{\text{term}}(\boldsymbol{s}(T)) \tag{7}$$

where R_{step} denotes the step reward at time t, reflecting the performance of a single action; $\Delta\theta(-1) = 0$ (no heading adjustment occurs before the initial action, so the smoothness reward is zero); the discount factor $\gamma_{\text{disc}} \in (0,1)$; and R_{term} is the terminal reward, which is assigned if the goal is reached:

$$R_{\text{term}} = w_{\text{term}} - w_{\text{len}} \cdot \frac{L_{\text{path}} - L_{SG}}{L_{SG}} \tag{8}$$

where $w_{\text{term}} > 0$ is the terminal reward constant, $w_{\text{len}} > 0$ is the path-length penalty weight, and L_{path} denotes the total path length; otherwise, $R_{\text{term}} = -w_{\text{err}} \cdot (d_T - r_G)$, with $w_{\text{err}} > 0$ being the penalty weight for failing to reach the goal. The search process terminates when the agent successfully reaches the goal region, defined by $d_T \le r_G$, also when the maximum search depth $D_{\text{tree,max}}$ or the maximum number of iterations $I_{\max}$ are exhausted. At the termination state, the cumulative reward is formulated as:

$$R_{\text{step}} = R_{\text{dist}} + R_{\text{tangent}} + R_{\text{prog}} + R_{\text{smooth}} + R_{\text{eff}} + R_{\text{pred}} \tag{9}$$

Each item of the above formula jointly accounts for efficiency in approaching the goal, obstacle avoidance, and heading smoothness. Each component is detailed below. Specifically, R_{dist} is the distance-based reward, which increases as the agent gets closer to the goal, thereby encouraging progress toward the target. It is defined as:

$$R_{\text{dist}} = -w_d \cdot d_t \tag{10}$$

where $d_t = \|\boldsymbol{p}(t) - \boldsymbol{p}_G\|$ denotes the distance to the goal at time t, and $w_d > 0$ is the weight for the distance-based reward. R_{tangent} is the tangent reward, designed to encourage the agent to navigate along a tangent trajectory within a safe distance $d_{\text{min,clear}}$ from obstacle boundaries, thereby reducing detour length when circumnavigating obstacles. This is necessary because path-efficiency rewards alone are insufficient to guide the agent toward such tangent paths. Let $d_{\tan}$ be the tangent distance to the goal, and let $d_{\text{obs},k}$ denote the actual distance to obstacle k that requires circumnavigation. Defining the tolerance as $\epsilon_{\tan} = |d_{\text{obs},k} - d_{\tan}|$, the tangent reward is given by:

$$R_{\text{tangent}} = -w_{\tan} \cdot \frac{|d_{\text{obs},k} - d_{\tan}| - \epsilon_{\tan}}{\epsilon_{\tan}} \tag{11}$$

where $w_{\tan} > 0$ is the weight for the tangential reward. R_{prog} denotes the progress reward, which incentivizes the agent for moving closer to the goal and provides no reward when moving away, defined as:

$$R_{\text{prog}} = w_{pr} \cdot \max(0, d_{t-1} - d_t) \tag{12}$$

where $w_{pr} > 0$ is the weight for the progress reward. The smoothness reward R_{smooth} is based on the change in heading deviation between consecutive actions. It penalizes large and frequent heading changes that increase energy consumption and reduce stability, thereby encouraging actions with small heading variations to ensure trajectory smoothness.

$$R_{\text{smooth}} = w_s \cdot \left(1 - \left(\frac{|\Delta\theta_t - \Delta\theta_{t-1}|}{2\Delta\theta_{\max}}\right)^2\right) \tag{13}$$

where $w_s > 0$ and $w_p > 0$ denote the weights for smoothness reward and smoothness penalty, respectively, and $\Delta\theta_{\max}$ represents the maximum heading deviation in the action space; for the initial action, $R_{\text{smooth}} = 0$. The path efficiency reward R_{eff} encourages the agent to plan efficient paths by rewarding progress toward the goal per unit path length:

$$R_{\text{eff}} = w_{\text{eff}} \cdot \frac{\max(0, d_{t-1} - d_t)}{\Delta l_t} \tag{14}$$

where $\Delta l_t = \|\boldsymbol{p}(t) - \boldsymbol{p}(t-1)\|$ denotes the path length traversed from time $t-1$ to t, and w_{eff} is the efficiency reward weight.

The predictive dynamic avoidance reward R_{pred} evaluates the future collision risk posed by moving obstacles over a finite prediction horizon h. At each planning step, it computes the minimum clearance between the candidate position and all dynamic obstacles across future time steps $t+1$ through $t+1+h$, under constant-velocity motion prediction. This encourages the agent to preemptively deviate from trajectories that, while currently safe, will be occupied by approaching obstacles in the near future:

$$R_{\text{pred}} = \begin{cases} -w_{\text{pred}} \cdot \left(\dfrac{c_{\text{safe}}^{\text{pred}} - c_{\min}^{\text{pred}}}{c_{\text{safe}}^{\text{pred}}}\right)^{1.3}, & c_{\min}^{\text{pred}} < c_{\text{safe}}^{\text{pred}} \\ w_{\text{pred}} \cdot \epsilon, & c_{\min}^{\text{pred}} \geq c_{\text{safe}}^{\text{pred}} \end{cases} \tag{15}$$

where the predicted minimum clearance is defined as:

$$c_{\min}^{\text{pred}} = \min_{k=0,1,\ldots,H} \min_{j=1,\ldots,M} \left(\|\mathbf{p}_{\text{new}} - \hat{\mathbf{p}}_j(t+1+k)\| - r_j - r_{\text{agent}}\right) \tag{16}$$

where $\hat{\mathbf{p}}_j(t) = \mathbf{p}_j^0 + \mathbf{v}_j \cdot t \cdot \Delta t$ denotes the predicted position of the j-th dynamic obstacle under constant-velocity extrapolation with boundary reflection, r_j is its radius, M is the total number of dynamic obstacles, and $c_{\text{safe}}^{\text{pred}} = 3r_{\text{agent}}$ is the predictive safety margin. The exponent 1.3 provides a superlinear penalty gradient that intensifies rapidly as the predicted clearance diminishes, while remaining less aggressive than the instantaneous clearance penalty to avoid excessive conservatism at longer prediction horizons. When the predicted clearance exceeds the safety threshold, a small positive bonus $\epsilon = 0.05$ is awarded to mildly encourage trajectories that remain well-separated from future obstacle positions.

3.3 Overview of Proposed Method

Combining the aforementioned constraints and rewards, the agent's path planning problem can be formulated as follows: at each time step, find an action sequence $\{\Delta\theta(0), \ldots, \Delta\theta(T-1)\}$ and the corresponding

state sequence $\{\boldsymbol{s}(0), \ldots, \boldsymbol{s}(T)\}$ within the intersection of all constraints $\mathcal{C}$ that maximizes the cumulative reward J, i.e.:

$$\begin{cases} \max_{\{\Delta\theta(t)\},\{\boldsymbol{x}(t)\}} J & \text{s.t. } \boldsymbol{s}(t+\Delta t) = f(\boldsymbol{s}(t), \Delta\theta(t)) \\ \boldsymbol{s}(t) \in \mathcal{C} & \forall t \in [0, T] \end{cases} \tag{17}$$

where $f(\cdot,\cdot)$ denotes the dynamics update function, and $\mathcal{C}$ represents the intersection of all constraints. As outlined in Algorithm 1, the algorithm initiates from the agent's current position and orientation, invoking the MCTS path planner to generate candidate trajectories. Only the first H_{exec} steps of the selected candidate trajectory are executed to mitigate error accumulation from long-horizon planning. After execution, the agent's state is updated, and the planning process repeats from this new state, establishing a closed-loop "plan–execute–replan" cycle. The procedure terminates when the agent reaches the goal region or the maximum number of outer-loop iterations $I_{\max}$ is reached, at which point the final executed trajectory is returned. This mechanism enables high-frequency replanning to adapt to complex environments while leveraging MCTS's global search capability to ensure trajectory feasibility.

The HRMCTS framework iteratively optimizes path planning through a structured four-phase cycle: selection, expansion, rollout, and backpropagation. During the selection phase, the algorithm traverses the search tree from the root using the UCT strategy. This approach maintains a principled balance between the exploitation of known high-value trajectories and the exploration of uncharted state spaces. Following the expansion of a leaf node, the rollout phase executes a rapid forward trajectory simulation up to a maximum depth of $D_{\text{rollout,max}}$. To enhance computational efficiency and avoid the sub-optimality of purely random exploration, this phase incorporates a heuristic action selection strategy. At each simulation step $k \in [1, D_{\text{rollout,max}}]$, the action space is constrained by the target's relative bearing, $\psi_{\text{target}} = \text{acrtan}(y_g - y_k, x_g - x_k) - \theta_k$. By prioritizing actions that minimize this angular deviation, the algorithm imposes a forward branching constraint B_{forward} that effectively prunes search paths deviating significantly from the goal. Each step within the rollout strictly adheres to kinematic constraints via state transition equations while simultaneously performing boundary and collision checks. If the simulation detects a collision or a breach of a no-fly zone, it terminates prematurely and applies a penalty. The terminal cumulative quality of the trajectory is then quantified by Eq. (7). Finally, in the backpropagation phase, the total reward is propagated upward through the traversed path to update the value estimates of all ancestral nodes. Specifically, the visit count for each node in the path is incremented, and the node value is updated using the incremental mean formula:

$$Q_{\text{new}} = Q_{\text{old}} + \frac{1}{N}(R_{\text{total}} - Q_{\text{old}}) \tag{18}$$

This iterative update ensures that the statistical value of each node converges toward the true expected reward, ultimately yielding the optimal, kinematically feasible decision for the current planning step.

Algorithm 1: HRMCTS planner pseudo-code

Input: algorithm parameters and initial state of agent and environment
Output: an optimized safe path between start and destination
while *not at destination* **and** *outer loop step* $< I_{\max}$ ***and*** $\neg fail$ **do**
    $start \leftarrow$ currernt position; $headingangle \leftarrow$ currernt head ▷ update state
    for $i = 1$ **to** $I_{\text{mcts,max}}$ **do**

(Continued)

Algorithm 1 (continued)

```
        leaf ← Improved UCB1Select(root)                                  ▷ selection
        if ¬root.terminal and root.depth < D_tree,max then
            leaf ← Expand(leaf)                                           ▷ expansion
        end
        r ← ForwardRollout(leaf, D_rollout,max)                           ▷ rollout
        Backpropagate(leaf, r)                                            ▷ backpropagation
    end
    candidate path ← ExtractBestPath(root)                ▷ get best path(highest average reward)
    return candidate path                                                 ▷ get candidate paths
    if step number(candidate path) < H_exec then
        break                                  ▷ unable to generate sufficient paths and exit loop
        return
    end
    else
        take the first H_exec step to form an execution segment
        add execution segment to execution path
        current position ← execution segment end point                    ▷ update new state
        current head ← end point heading
    end
    outer loop step ← outer loop step + 1
end
return
```

4 Simulation Results and Analysis

In this section, we first conduct comparative experiments to demonstrate the effectiveness of the proposed HRMCTS method. We then evaluate the performance of MCTS under various parameter settings, with a particular comparison against the PSO algorithm. Highlighting the real-time capability, robustness, and rapid convergence of HRMCTS. These results substantiate the ability of MCTS to address sequential decision-based path planning problems. Specifically, we first describe our experimental setup, followed by test results across different tasks and conditions, and conclude with a performance analysis of HRMCTS. The simulation environment runs on a system equipped with an AMD Ryzen 7 5800H CPU, an NVIDIA GeForce RTX 3060 laptop GPU with 6 GB of memory.

To ensure a fair comparison, all algorithms were assigned identical path planning tasks. The parameters and iteration limits of the conventional MCTS and HRMCTS were set identically, and the fitness function of PSO was aligned with the rollout value evaluation function of the improved MCTS, ensuring that both algorithms shared the same optimization objective. All necessary parameters mentioned above are listed in Table 1. Systematic experiments established the core parameters for the HRMCTS algorithm to balance real-time UAV constraints with safe navigation in complex environments. Following classical UCT theory, the exploration constant is defined as $C = 1.4$. To meet the real-time threshold of $T_{\max} = 1.8$ s per planning step, the maximum number of iterations is capped at $I_{\max} = 180$, as additional iterations yield diminishing marginal improvements. Weighting follows a magnitude hierarchy designed around a "safety-first, goal-oriented" framework. A dominant terminal reward ensures that reaching the goal takes precedence over cumulative step-level gains. With $w_{\text{terminal}} = 2000$ and $w_{\text{progress}} = 18.0$, the algorithm maintains a strong attraction toward the goal. This configuration ensures that a unit displacement of 1m results in a positive

increment of roughly 18.0, which effectively alleviates "dead zone" issues in cornering scenarios. Regarding dynamic safety, the predictive penalty weight $w_{\text{pred}} = 35.0$ is intentionally higher than the instantaneous clearance weight. This reflects the principle that anticipating future collisions is more critical than current proximity because early evasion is kinematically more efficient than last-moment maneuvers. The smoothness weight $w_{\text{smooth}} = 6.0$ penalizes steering magnitude and inter-step fluctuations. This filters out kinematically infeasible zigzag paths while allowing for essential evasive actions. This design ensures that safety-critical objectives override efficiency when required. Meanwhile, the global path reference provides firm but flexible guidance that permits local deviations when encountering dynamic threats. A discount factor of $\gamma = 0.97$ ensures that cumulative rewards over a 20-step rollout remain significant at $\gamma^{20} \approx 0.54$, effectively balancing long-term planning with immediate collision avoidance. These parameters were validated through ablation studies spanning diverse scenarios with varying obstacle densities and velocities.

Table 1: Algorithm parameters and details.

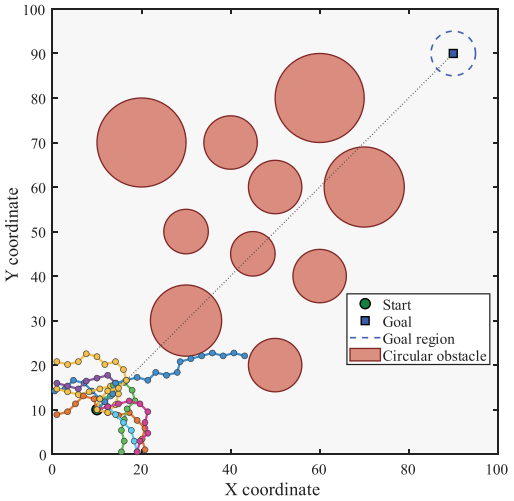
Parameter Symbol	Meaning	Value	Unit
	Map and Environment		
W, H	2D map width/height	$100, 100$	m
$\boldsymbol{p}_S = (x_S, y_S)$	Start coordinates (vehicle)	$(10, 10)$	m
$\boldsymbol{p}_G = (x_G, y_G)$	Target coordinates	$(90, 90)$	m
r_G	Target region radius (goal tolerance)	5	m
	Vehicle Kinematics		
r_{agent}	Agent collision radius	0.5	m
$\Delta\theta_{\max}$	Max steering angle offset	$\pi/4$	rad
$N_{\Delta\theta}$	Sampled steering angles count	9	–
$v_{\max}$	Vehicle max linear velocity	5	m/s
Δt	State update time step	1.0	s
θ_{init}	Initial heading angle of vehicle	$\pi/4$	rad
	Constraints		
δ_{bound}	Agent-map boundary margin	1.0	m
$d_{\text{min,clear}}$	Obstacle/boundary safe distance	3.0	m
	Reward and Penalty		
w_{d}	Distance-to-target reward weight	2.0	–
w_{pr}	Progress-to-target reward weight	18.0	–
w_{s}	Trajectory smoothness reward weight	6.0	–
w_{pred}	Predictive penalty weight	35.0	–
w_{tan}	Obstacle tangent movement reward weight	9.0	–
w_{eff}	Path efficiency reward weight	8.0	–
w_{term}	Terminal reward constant	2000	–
w_{len}	Overlong path penalty weight	10.0	–
w_{err}	Unreached target penalty weight	30.0	–
γ	Cumulative reward discount factor	0.95	–
ϵ_{tan}	Tangent reward clearance tolerance	0.55	m
	MCTS Algorithm		
C_{explore}	UCB1 exploration coefficient	1.4	–
$D_{\text{rollout,max}}$	Single rollout max steps	35	–
B_{forward}	Forward candidate branch limit	3	–
$I_{\text{mcts,max}}$	MCTS max iterations per plan	120	–
$D_{\text{tree,max}}$	MCTS tree max depth	200	–

(Continued)

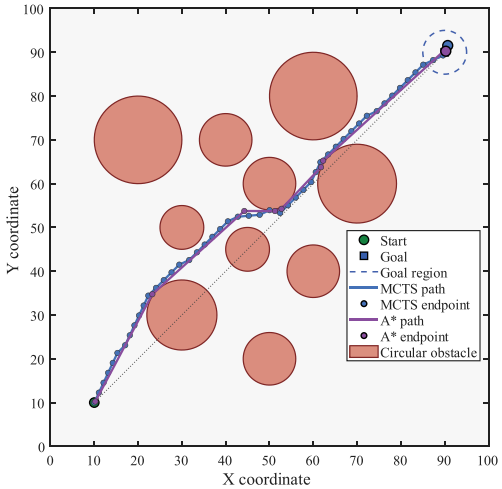
Table 1 (continued)

Parameter Symbol	Meaning	Value	Unit
$I_{\max}$	MCTS outer iterations	150	–
H_{exec}	MCTS execute horizon per step	3	–
	PSO Algorithm		
$I_{\text{pso,max}}$	PSO max iterations	100	–
N_{pso}	Number of particles (PSO)	20	–
w_{inert}	PSO inertia weight	0.7	–
c_1	PSO cognitive coefficient	1.5	–
c_2	PSO social coefficient	1.5	–
w_{decay}	PSO inertia weight decay rate	0.9	–
N_{wp}	Number of waypoints	18	–
$I_{\text{stall,max}}$	PSO max fitness stall iterations	8	–
$\epsilon_{\text{fitness}}$	PSO fitness stall tolerance	3×10^{-3}	–

This paper focuses on several representative path planning parameters. Notably, all effective path planning results presented below are averaged over multiple simulation runs and thus represent typical algorithmic behavior. The conventional MCTS algorithm serves as a baseline to demonstrate the effectiveness of the proposed approach. As illustrated in Fig. 2a, the baseline failed to complete the path planning task in all 10 trials. For this study, a trial is defined as a "failure" if the algorithm cannot identify a collision-free path to the goal within a limit of 50,000 iterations. This shortcoming stems from the reliance on purely random sampling during the rollout phase, which is particularly problematic when the action space is constrained by kinematic requirements. In such high-dimensional environments, it is statistically improbable for a random policy to select a sequence of actions that consistently nears the goal. These rollouts often terminate in collisions or reach the iteration cap without success. Consequently, the algorithm fails to provide accurate value evaluations for state-action pairs. Without effective simulation-based estimates, the search tree is unable to converge on a feasible solution within a practical computational budget.



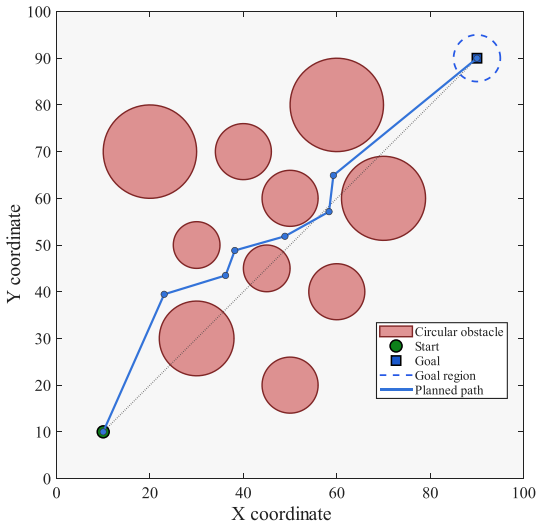
(a) Random MCTS path planning result diagram.



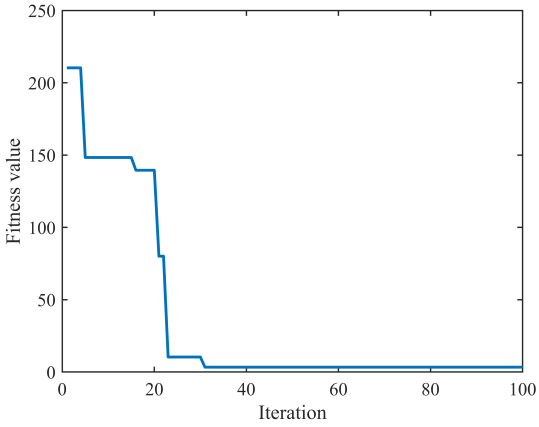
(b) HRMCTS and A* algorithm path planning result diagram.

Figure 2: Comparison of random MCTS and HRMCTS results.

The results indicate a significant improvement over conventional MCTS, featuring an average planning time of approximately 0.001 s per step. This duration is substantially shorter than the execution time step and ensures that planning concludes before execution commences. Fig. 2b compares the computation time and path efficiency across 100 repeated simulations. The findings for HRMCTS and PSO represent stable outcomes verified through these 100 runs to remove the impact of randomness. PSO serves as a representative baseline for traditional heuristic algorithms, employing a fitness function identical to the MCTS rollout reward function as detailed in Table 1. As shown in Fig. 3a,b, PSO achieves effective path planning with convergence occurring after roughly 30 iterations and a path efficiency of 94.48%. However, its computational cost is significant, averaging 211.8 s per run. Note that all reported computational costs, including this average, are derived from 10 independent repeated experiments to ensure statistical reliability.



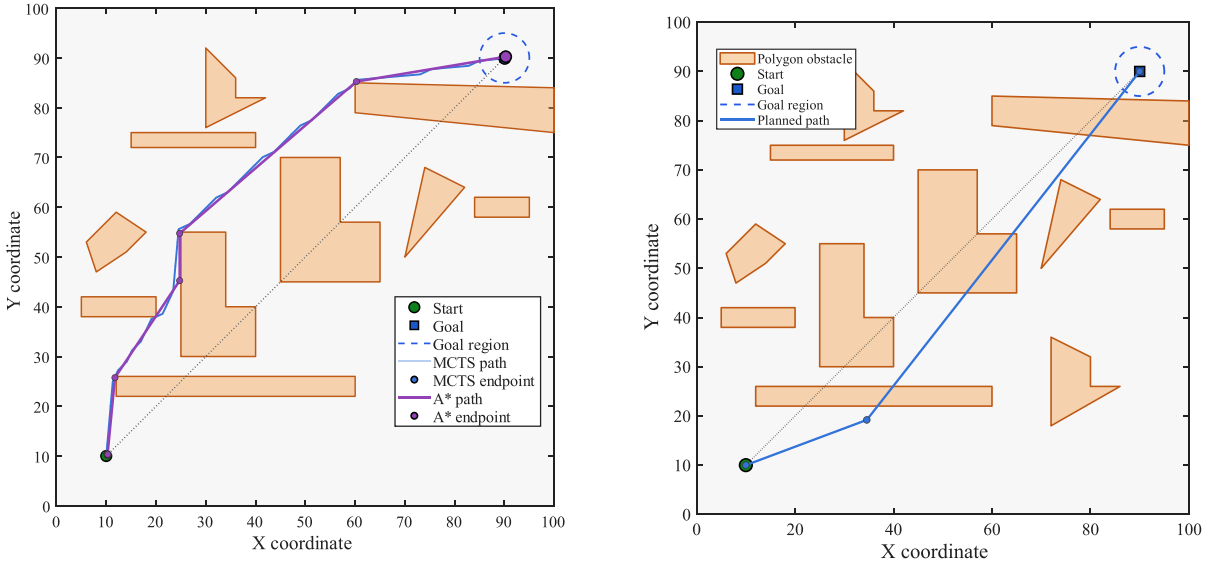
(a) PSO path planning result diagram.



(b) Diagram of PSO fitness value converge history.

Figure 3: PSO path planning results.

This study utilizes the A* algorithm to generate the global reference path within a 100 × 100 continuous workspace. The environment is discretized into an occupancy grid with a resolution of $\delta = 0.5$ units. A grid cell is classified as occupied if its Euclidean distance to the nearest polygonal obstacle boundary is less than the vehicle collision radius $r_{\text{agent}} = 0.5$. Starting from the grid cell corresponding to the start position $\boldsymbol{p}_s$, the algorithm expands nodes across an 8-connected neighborhood. This process employs Euclidean distance as the heuristic function $h(\boldsymbol{n}) = \|\boldsymbol{n} - \boldsymbol{p}_G\|_2$, where $\boldsymbol{p}_G$ denotes the goal. Finally, a goal tolerance of $\boldsymbol{r}_{\text{goal}}$ is applied to define the completion of the path. Subsequent experiments were conducted in more complex environments, and the results in Fig. 4a demonstrate that the proposed HRMCTS method can effectively perform path planning. In contrast, the PSO methods exhibite significant failure, yielding unsuccessful planning outcomes even after prolonged execution, as shown in Fig. 4b. This highlights the effectiveness and superiority of the proposed approach.



(a) HRMCTS path planning result diagram.

(b) PSO path planning result history.

Figure 4: Comparison of HRMCTS and PSO results in polygon obstacle environment.

To further validate HRMCTS in challenging scenarios, simulation tests were conducted in dynamic environments where obstacles moved along predefined trajectories. These tests demanded high-fidelity online replanning and real-time responsiveness. To evaluate obstacle avoidance, a circular dynamic obstacle with a 2.5 m radius was introduced, moving at 1.6 m/s. Its initial position and velocity were set to intercept the agent at a narrow passage along the shortest path. This point is a critical location that the A* algorithm inherently traverses. Since A* cannot adjust its path online, a collision is inevitable in this scenario. In contrast, HRMCTS uses its receding-horizon replanning and heuristic guidance to perceive motion and adjust the trajectory dynamically. This rolling-horizon mechanism is well-suited for time-varying environments because it allows for continuous updates to maintain safety. For a quantitative assessment, a reinforcement learning baseline using a Double Deep Q-Network (ddQn) was also implemented. The ddQn agent observes a 22-dimensional state vector $\mathbf{s}_t$ involving position, heading, goal-relative metrics, and 8-directional obstacle-distance rays. It outputs actions from a discrete space of $N_a = 9$ steering angles within $[-\pi/4, \pi/4]$. The Q-function is parameterized by a $22 \to 256 \to 128 \to 5$ fully connected network and was trained via the Adam optimizer ($\alpha = 3 \times 10^{-4}$) over 2000 episodes. To ensure a fair comparison, both planners share identical kinematics, obstacle representations, and reward functions within the same MPC rolling-horizon framework. Training performance for the RL method is illustrated in Fig. 5. The significant drop in success rate during training likely stems from hard target network updates and stale samples in the replay buffer. Hard updates occur every 20 episodes and cause sudden jumps in target Q-values. These jumps increase TD error and disrupt the learned policy. Furthermore, stale samples from early exploration exacerbate estimation bias. The success rate eventually recovers as the network adapts to new target values and higher-quality experiences replace older samples.

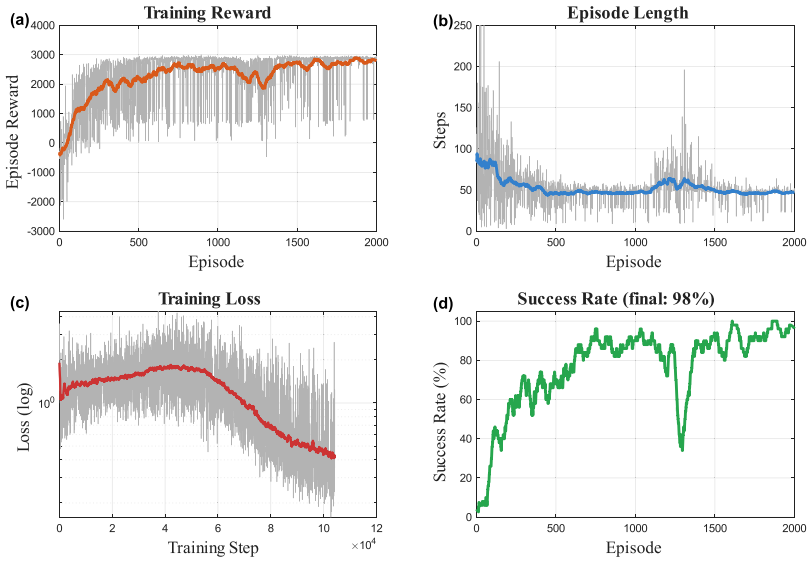
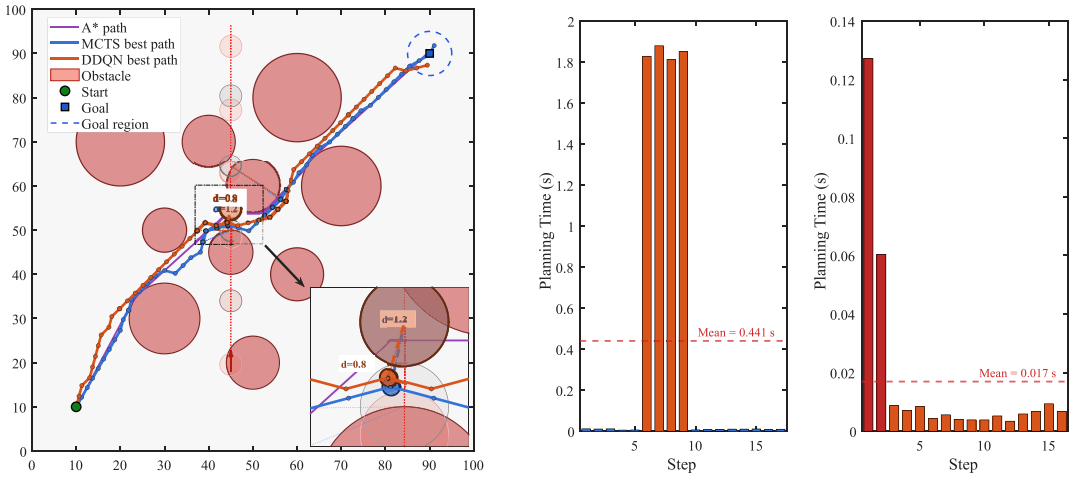


Figure 5: The gray vertical line represents the corresponding fluctuation range, and the solid line represents the average value. (**a**) The trend of training reward changes. (**b**) The trend of single round task steps. (**c**) The trend of training loss changes. (**d**) The trend of task success rate changes.

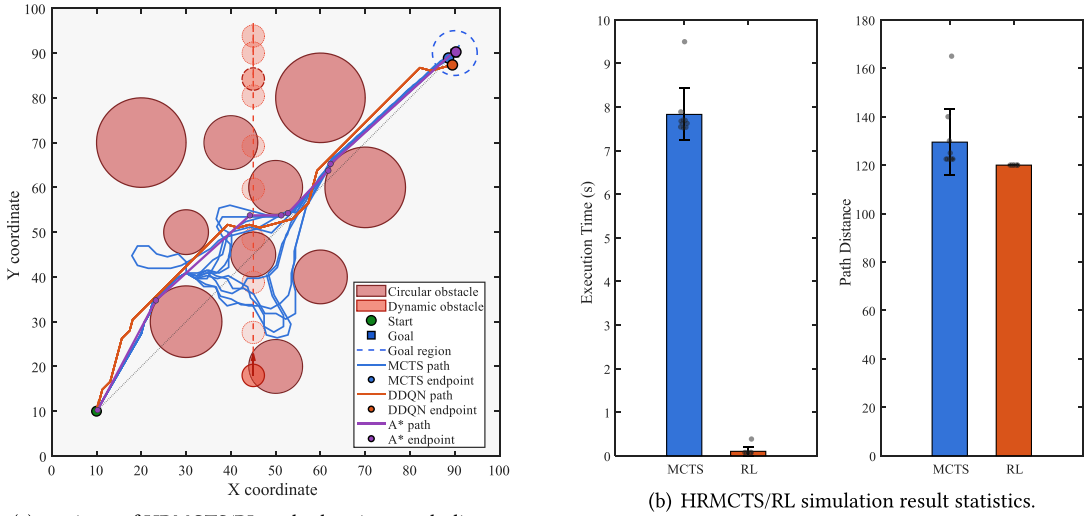
Fig. 6a illustrates the optimal dynamic avoidance performance for the circular obstacle scenario, confirming that both HRMCTS and RL algorithms successfully achieve collision avoidance. Analysis of the execution times in Fig. 6b shows that the computational cost of HRMCTS is highly targeted. The algorithm performs intensive calculations only when a dynamic obstacle is in close proximity and poses an immediate threat. This selective need for resources demonstrates the efficiency of the hierarchical framework. In comparison, the RL method requires planning times more than an order of magnitude shorter than those of HRMCTS. Data from ten independent trials in Fig. 7a,b further establish the RL approach as highly efficient and deterministic. This consistency is a primary advantage stemming from its extensive offline training phase.



(a) HRMCTS/RL path planning result diagram.

(b) HRMCTS(left)/RL(right) simulation time cost result.

Figure 6: Dynamic obstacle avoidance results and per-step running time of HRMCTS and RL under circular obstacle conditions.



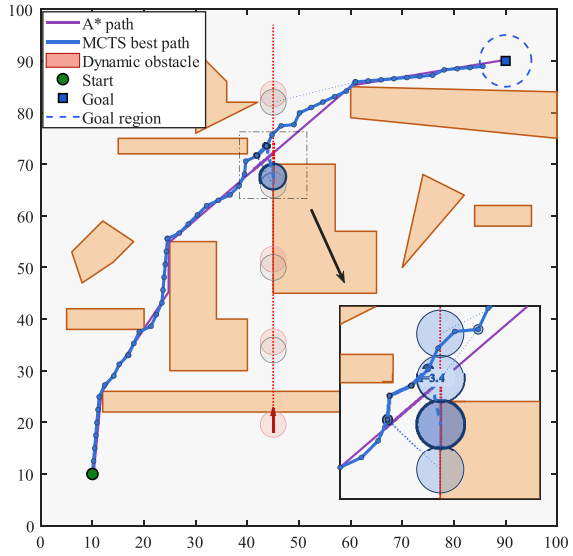
(a) 10 times of HRMCTS/RL path planning result diagram.

(b) HRMCTS/RL simulation result statistics.

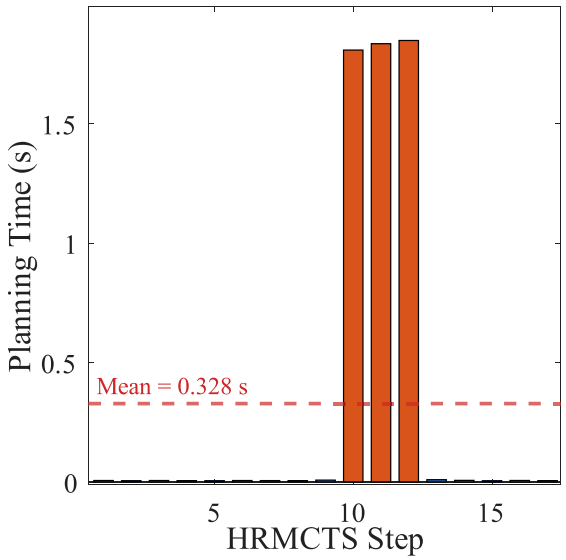
Figure 7: 10-path statistics and operation result statistics: dynamic obstacle avoidance results of HRMCTS and RL under circular obstacle conditions.

Performance results diverge significantly in complex environments featuring polygonal obstacles. As illustrated in Fig. 8a,b, HRMCTS consistently finished planning tasks. This stability underscores the algorithm's inherent robustness. Conversely, the RL algorithm fails across all trials in this setting, as shown in Fig. 9a. While HRMCTS's performance characteristics, including mean and standard deviation, remain highly consistent with the circular obstacle results shown in Fig. 9b. This failure is primarily due to the limited generalization of the RL policy. Since the model was trained only in static circular environments, it cannot adapt to the combined demands of dynamic avoidance and complex polygonal constraints. These findings suggest that while HRMCTS requires a slightly higher computational overhead, it provides greater utility for engineering applications. Its superior generalization and "anytime" characteristics allow it to handle complex, real-world dynamic scenarios effectively without environment-specific training.

To evaluate the computational scalability of the MCTS planner, a series of experiments examined various action-space granularities by adjusting the number of actions $N_{\Delta\theta}$. As illustrated in Fig. 10, these tests explored the trade-off between the path quality benefits of increased action flexibility and the computational overhead caused by a larger branching factor. Each configuration underwent 100 independent trials within a complex polygonal obstacle environment. Results show that as $N_{\Delta\theta}$ scales from 5 to 15, both per-step and total planning times exhibit a clear upward trend. Computational cost and its variance reach a peak at $N_{\Delta\theta} = 15$ due to the drastic expansion of the search space. Reliability also varied significantly. Success rates remained above 98% for $N_{\Delta\theta} \geq 7$ but dropped to approximately 80% at $N_{\Delta\theta} = 5$. This performance gap is primarily caused by "control blind spots" resulting from sparse action discretization. Limited steering options provide insufficient degrees of freedom for navigating sharp vertices or narrow passages. This constraint often prevents the search tree from finding feasible solutions within a finite depth. Although a higher action count can reduce total path distance, the improvement at $N_{\Delta\theta} = 15$ comes at an excessive temporal cost. A value of $N_{\Delta\theta} = 9$ maintains high success rates and path efficiency while offering much better cost-effectiveness. Because its planning times are significantly lower than those of higher-dimensional spaces, $N_{\Delta\theta} = 9$ is selected as the optimal granularity for satisfying both robustness and real-time requirements.

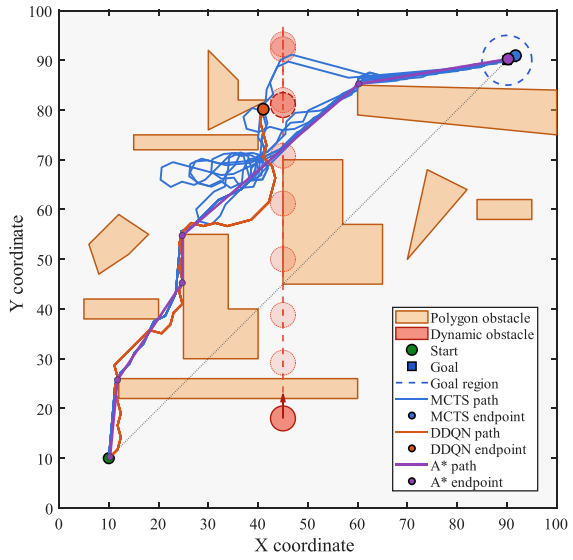


(a) HRMCTS/RL path planning result diagram.

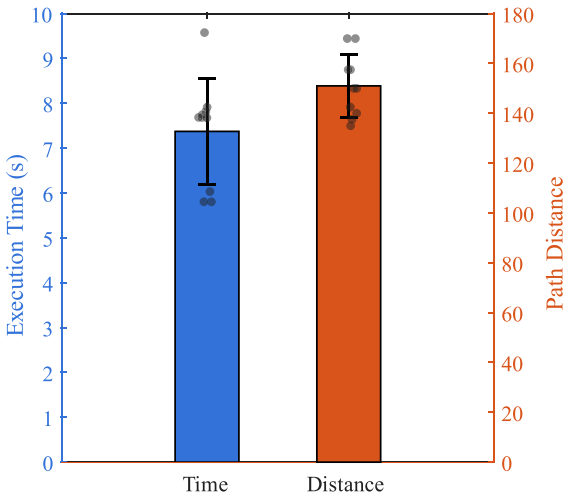


(b) HRMCTS simulation time cost result.

Figure 8: Dynamic obstacle avoidance results and per-step running time of HRMCTS and RL under polygongal obstacle conditions.



(a) 10 times of HRMCTS/RL path planning result diagram.



(b) HRMCTS simulation result statistics.

Figure 9: 10-path statistics and operation result statistics: dynamic obstacle avoidance results of HRMCTS and RL under polygonal obstacle conditions.

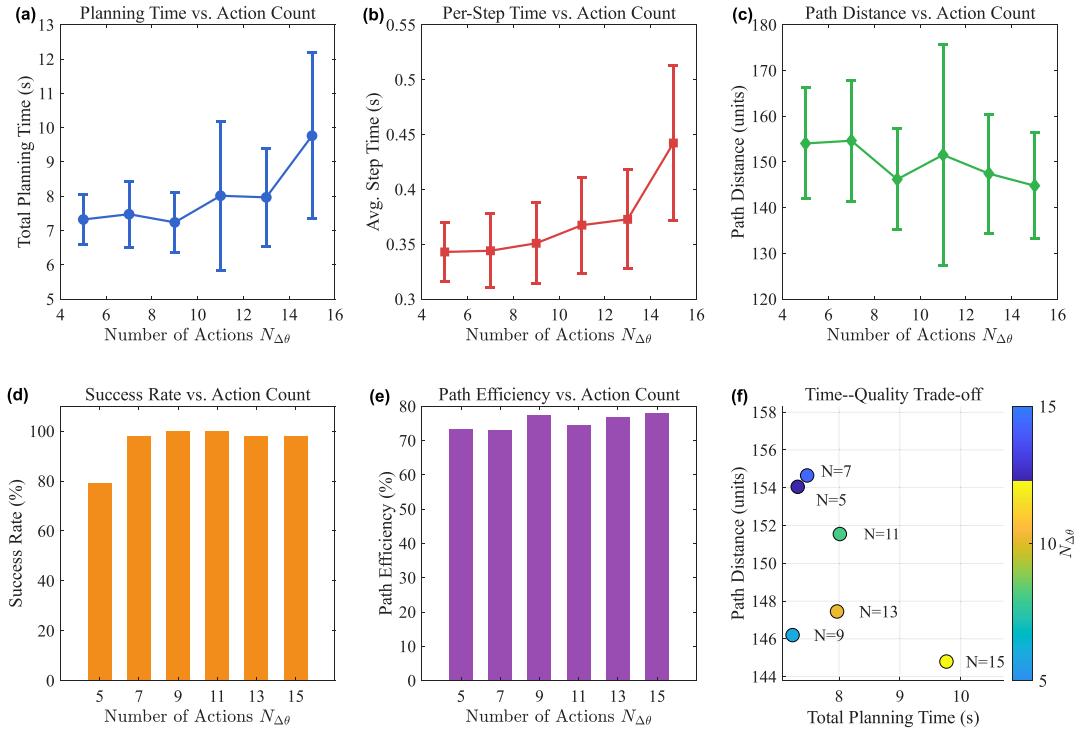


Figure 10: Scalability analysis of numbers of action space. (**a**) Planning time vs. action count. (**b**) Per-step time vs. action count. (**c**) Path distance vs. action count. (**d**) Success rate vs. action count. (**e**) Path efficiency vs. action count. (**f**) Time–quality trade-off.

Finally, each method was independently executed 100 times to generate the comparative results summarized in Table 2. These findings reveal significant differences in path planning performance among the three approaches. The proposed HRMCTS method demonstrates superior adaptability compared to raw MCTS and ddQn, particularly within complex dynamic environments. It also achieves better computational efficiency than the traditional PSO heuristic, reaching a performance level comparable to the A* algorithm. Ultimately, the experimental results indicate that HRMCTS delivers high-quality path planning while balancing efficiency and interpretability through its inherent tree-based structure.

Table 2: Comprehensive comparison of path planning methods across different environments.

Method	Environment	Dynamic Obs.	Success Rate	Path Efficiency	Total Execution Time	Path Length Std.
Raw MCTS	Any	No	0% (Failed)	N/A	Fail	N/A
PSO	Circular	No	High	94.18%	211.8s	5.15
PSO	Complex Polygonal	No	0% (Failed)	N/A	N/A	N/A
A*	Circular	No	100%	97.0%	0.004s	0
A*	Complex Polygonal	No	100%	88.8%	0.001s	0
RL (ddQn)	Circular	Yes	100%	90.6%	0.099s	0
RL (ddQn)	Complex Polygonal	Yes	0% (Failed)	N/A	N/A	N/A
HRMCTS	Any	No	100%	96.7%	0.9s	0.0043
HRMCTS	Circular	Yes	100%	83.96%	7.60s	0.596
HRMCTS	Complex Polygonal	Yes	100%	78.0%	7.37s	1.17

5 Conclusion

This paper introduces the HRMCTS framework for unmanned system path planning in complex environments. The approach enhances MCTS performance through several critical innovations. First, a heuristic UCT selection strategy incorporates goal distance and obstacle clearance to improve search directionality and convergence. Second, a multi-objective reward function applied during the rollout phase optimizes path length, goal progress, obstacle avoidance, trajectory smoothness, and time efficiency. This ensures a balanced trade-off between safety and performance. Third, a receding-horizon mechanism integrates local path generation with online replanning to maintain global feasibility while ensuring real-time responsiveness. To systematically evaluate the method, simulation experiments were conducted in four scenarios: static and dynamic environments with both circular and polygonal obstacles. The proposed framework is compared against standard MCTS, PSO, and ddQn baselines. The experimental results demonstrate:

- HRMCTS achieves a 100% success rate and 96.7% path efficiency in static environments, notably outperforming raw MCTS, which fails across all trials. The proposed method also provides a substantial speed advantage over PSO, which requires 211.8 s to complete a single run. While the A* algorithm establishes the performance ceiling with 97.0% efficiency, HRMCTS produces nearly identical results and maintains a minimal path length std of 0.0043.
- Both HRMCTS and the ddQn achieve 100% success in dynamic circular environments. Although ddQn yields a higher path efficiency of 90.6% in circular environment due to its offline training, HRMCTS demonstrates robust adaptability by achieving success in both circular and polygonal environments. This outcome is particularly significant because HRMCTS operates effectively without any prior knowledge of the environment or the obstacle field.
- When navigating complex polygonal dynamic environments, the performance of alternative methods diverges significantly as both ddQn and PSO fail to resolve the planning tasks. Conversely, HRMCTS maintains a 100% success rate with a stable path efficiency of 78.0% and a relatively low standard deviation. These findings confirm the superior generalization and reliability of the HRMCTS framework when addressing the coupled constraints of intricate geometries and moving obstacles.

HRMCTS effectively improves the efficiency constraints found in traditional MCTS when navigating sparse-reward and high-dimensional environments. The framework consistently demonstrates superior planning performance in some scenarios relative to traditional methods such as PSO, A* and ddQn. By utilizing a transparent tree structure alongside node-level statistical metrics, the algorithm ensures that the decision-making process is both interpretable and traceable. This approach successfully combines the explainability typically absent in RL with the dynamic adaptability that conventional search methods frequently lack. Moreover, the "anytime" planning capabilities of the algorithm facilitate efficient real-time responses even under limited computational resources. This functionality maintains a necessary balance between engineering practicality and operational reliability. This makes HRMCTS particularly well-suited for dynamic scenarios characterized by dense obstacles and rapidly changing states, such as urban low-altitude drone navigation and multi-agent coordination in warehouse logistics, where it enables rapid obstacle avoidance and local re-planning based on limited environmental perception. In contrast to black-box models that rely on extensive training, sampling strategies that lag in abruptly changing environments, or graph search methods dependent on structured settings, HRMCTS preserves decision transparency while delivering enhanced robustness and real-time performance, making it a good choice for safety-critical autonomous systems operating in unpredictable environments that demand human-machine collaborative trust.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Hunan Provincial Natural Science Foundation of China (No. 2025JJ60072), the Open Research Subject of State Key Laboratory of Intelligent Game (No. ZBKF-24-01), the Postdoctoral Fellowship Program of CPSF (No. GZB20240989), the China Postdoctoral Science Foundation (No. 2024M754304) and the Military High-Level Talent Program (No. 202401-RCGC-ZZ-004).

Author Contributions: Qianshu Yang: conceptualization, methodology design, data collection and manuscript drafting; Shuangxi Liu: supervision, project administration, funding acquisition and manuscript revision; Xianyu Wu: formal analysis, simulation implementation and result verification; Wei Zhao: resource support, technical guidance and review of the final manuscript. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Some or all data, models, or code generated or used during the study are proprietary or confidential in nature and may only be provided with restrictions.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Su C, Zhao L, Xiang D. Dynamic integration of Q-learning and A-APF for efficient path planning in complex underground mining environments. Comput Mater Contin. 2026;86(2):1–24. doi:10.32604/cmc.2025.071319.
2. Cao H, Li S, Li X, Liu Y. A UAV path-planning approach for urban environmental event monitoring. Comput Mater Contin. 2025;83(3):5575–93. doi:10.32604/cmc.2025.061954.
3. Xing B, Wang X, Liu Z. An algorithm of complete coverage path planning for deep-sea mining vehicle clusters based on reinforcement learning. Adv Theory Simul. 2024;7(4):2300970. doi:10.1002/adts.202300970.
4. Rivière B, Lathrop J, Chung SJ. Monte Carlo tree search with spectral expansion for planning with dynamical systems. Sci Robot. 2024;9(97):eado1010. doi:10.1126/scirobotics.ado1010.
5. Liu S, Lin Z, Huang W, Yan B. Technical development and future prospects of cooperative terminal guidance based on knowledge graph analysis: a review. J Zhejiang Univ-SCI A. 2025;26(7):605–34. doi:10.1631/jzus.a2400317.
6. Gan C, Li Z, Ge Y, Lu M. COM trajectory planning and disturbance-resistant control of a bipedal robot based on CP-ZMP-COM dynamics. J Zhejiang Univ-SCI A. 2025;26(5):492–8. doi:10.1631/jzus.a2400062.
7. Qian Y, Sheng K, Ma C, Li J, Ding M, Hassan M. Path planning for the dynamic UAV-aided wireless systems using monte carlo tree search. IEEE Trans Veh Technol. 2022;71(6):6716–21. doi:10.1109/tvt.2022.3160746.
8. Fu W, Wang B, Li X, Liu L, Wang Y. Ascent trajectory optimization for hypersonic vehicle based on improved chicken swarm optimization. IEEE Access. 2019;7:151836–50. doi:10.1109/access.2019.2947297.
9. Dong T, Zhang L, Wang Z, Zhang J. Obstacle avoidance path planning method for intelligent vehicle based on obstacle projection. Meas Sci Technol. 2025;36(9):096215. doi:10.1088/1361-6501/ae0508.
10. Li H, Wang Y, Xu S, Zhou Y, Su D. Trajectory optimization of hypersonic periodic cruise using an improved PSO algorithm. Int J Aerosp Eng. 2021;2021(1):2526916. doi:10.1155/2021/2526916.
11. Yuan W, Chen J, Chen S, Feng D, Hu Z, Li P, et al. Transformer in reinforcement learning for decision-making: a survey. Front Inf Technol Electron Eng. 2024;25:763–90. doi:10.1631/FITEE.2300548.
12. Csippán G, Péter I, Kővári B, Bécsi T. MCTS-based policy improvement for reinforcement learning. Mach Learn Know Extr. 2025;7(3):98. doi:10.3390/make7030098.
13. Gao Y, Wang Y, Tian L, Hong X, Xue C, Li D. Evolving adaptive and interpretable decision trees for cooperative submarine search. Def Technol. 2025;48:83–94. doi:10.1016/j.dt.2025.02.007.
14. Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, et al. A survey of monte carlo tree search methods. IEEE Trans Comput Intell AI Games. 2012;4(1):1–43. doi:10.1109/tciaig.2012.2186810.

15. Trotti F, Farinelli A, Muradore R. Path re-planning with stochastic obstacle modeling: a monte carlo tree search approach. In: Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2024 Oct 14–18; Abu Dhabi, United Arab Emirates. p. 8017–22.
16. Sukwadi R, Airlangga G, Basuki WW, Kristian Y, Rahmananta R, Sugianto LF, et al. Comparative analysis of path planning algorithms for multi-UAV systems in dynamic and cluttered environments: a focus on efficiency, smoothness, and collision avoidance. Int J Robot Cont Syst. 2024;4(4):1602–16. doi:10.31763/ijrcs.v4i4.1555.
17. Bonanni L, Meli D, Castellini A, Farinelli A. Monte carlo tree search with velocity obstacles for safe and efficient motion planning in dynamic environments. arXiv:2501.09649. 2025.
18. Yang X, Jin T, Xu W, Qi C, Ma H. Dynamic environment path planning based on hybrid pre-training algorithm. Eng Res Exp. 2025;7(3):035267. doi:10.1088/2631-8695/adf93f.
19. Tang J, Liang Y, Li K. Dynamic scene path planning of UAVs based on deep reinforcement learning. Drones. 2024;8(2):60. doi:10.3390/drones8020060.
20. Dong CK. Adaptive long-range UAV flight planning using Monte Carlo search trees. Vancouver, BC, Canada: University of British Columbia; 2026.
21. He Y, Liao B, Dang Q. Swarm-based cooperative coverage strategy for amphibious unmanned systems in complex wetland environments. IFAC-PapersOnLine. 2025;59(20):2388–93. doi:10.1016/j.ifacol.2025.11.516.
22. Li B, Page BR, Moridian B, Mahmoudian N. Collaborative mission planning for long-term operation considering energy limitations. IEEE Robot Autom Lett. 2020;5(3):4751–8. doi:10.1109/lra.2020.3003881.
23. Huang T, Chen Z, Gao W, Xue Z, Liu Y. A USV-UAV cooperative trajectory planning algorithm with hull dynamic constraints. Sensors. 2023;23(4):1845. doi:10.3390/s23041845.
24. Xue M, Huang H, Zhuang Y, Si J, Lü T, Sharma S, et al. Collaborative marine targets search algorithm for USVs and AAVs under energy constraint. IEEE Internet Things J. 2024;12(9):12137–52. doi:10.1109/jiot.2024.3520176.
25. Sun B, Lv Z. Multi-AUV dynamic cooperative path planning with hybrid particle swarm and dynamic window algorithm in Three-Dimensional terrain and ocean current environment. Biomimetics. 2025;10(8):536. doi:10.3390/biomimetics10080536.
26. Häusler AJ, Saccon A, Aguiar AP, Hauser J, Pascoal AM. Cooperative motion planning for multiple autonomous marine vehicles. IFAC Proc Vol. 2012;45(27):244–9.
27. Świechowski M, Godlewski K, Sawicki B, Mańdziuk J. Monte Carlo tree search: a review of recent modifications and applications. Artif Intell Rev. 2023;56(3):2497–562. doi:10.1007/s10462-022-10228-y.
28. Kemmerling M, Lütticke D, Schmitt RH. Beyond games: a systematic review of neural Monte Carlo tree search applications. Appl Intell. 2024;54(1):1020–46. doi:10.1007/s10489-023-05240-w.
29. Coulom R. Computing "elo ratings" of move patterns in the game of go. ICGA J. 2007;30(4):198–208. doi:10.3233/icg-2007-30403.
30. Keller T, Eyerich P. PROST: probabilistic planning based on UCT. Proc Int Conf Autom Plan Sched. 2012;22:119–27.
31. Huang J, Liu Z, Lu B, Xiao F. Pruning in UCT algorithm. In: Proceedings of the 2010 International Conference on Technologies and Applications of Artificial Intelligence; 2010 Nov 18–20; Hsinchu, Taiwan. p. 177–81.
32. Baier H, Winands MH. Beam monte-carlo tree search. In: Proceedings of the 2012 IEEE Conference on Computational Intelligence and Games (CIG); 2012 Sep 11–14; Granada, Spain. p. 227–33.
33. Vien NA, Toussaint M. Hierarchical monte-carlo planning. Proc AAAI Conf Artif Intell. 2015;29(1):3613–9. doi:10.1609/aaai.v29i1.9687.
34. De Waard M, Roijers DM, Bakkes SC. Monte carlo tree search with options for general video game playing. In: Proceedings of the 2016 IEEE Conference on Computational Intelligence and Games (CIG); 2016 Sep 20–23; Santorini, Greece. p. 1–8.
35. Wang H, Zhang X, Mu C. Planning of heuristics: strategic planning on large language models with monte carlo tree search for automating heuristic optimization. arXiv:2502.11422. 2025.
36. Manome N, Shinohara S, Ui C. Simple modification of the upper confidence bound algorithm by generalized weighted averages. PLoS One. 2025;20(5):e0322757. doi:10.1371/journal.pone.0322757.
37. Lee J, Jeon W, Kim GH, Kim KE. Monte-carlo tree search in continuous action spaces with value gradients. Proc AAAI Conf Artif Intell. 2020;34:4561–8. doi:10.1609/aaai.v34i04.5885.

38. Finnsson H, Björnsson Y. Learning simulation control in general game-playing agents. Proc AAAI Conf Artif Intell. 2010;24(1):954–9. doi:10.1609/aaai.v24i1.7651.
39. Lorentz R. Using evaluation functions in Monte-Carlo tree search. Theor Comput Sci. 2016;644(4):106–13. doi:10.1016/j.tcs.2016.06.026.
40. Walędzik K, Mańdziuk J. An automatically generated evaluation function in general game playing. IEEE Trans Comput Intell AI Games. 2013;6(3):258–70. doi:10.1109/tciaig.2013.2286825.
41. Painter M, Lacerda B, Hawes N. Convex hull monte-carlo tree-search. Proc Int Conf Autom Plan Sched. 2020;30:217–25.
42. Xie F, Nakhost H, Müller M. A local monte carlo tree search approach in deterministic planning. Proc AAAI Conf Artif Intell. 2011;25:1832–3.
43. Rosin CD. Nested rollout policy adaptation for Monte Carlo tree search. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence; 2011 Jul 16–22; Barcelona Catalonia, Spain. p. 649–54.
44. Nagórko A. Parallel nested rollout policy adaptation. In: Proceedings of the 2019 IEEE Conference on Games (CoG); 2019 Aug 20–23; London, UK. p. 1–7.
45. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529(7587):484–9. doi:10.1038/nature16961.
46. Goodman J. Re-determinizing MCTS in Hanabi. In: Proceedings of the 2019 IEEE Conference on Games (CoG); 2019 Aug 20–23; London, UK. p. 1–8.
47. Childs BE, Brodeur JH, Kocsis L. Transpositions and move groups in monte carlo tree search. In: Proceedings of the 2008 IEEE Symposium On Computational Intelligence and Games; 2008 Dec 15–18; Perth, WA, Australia. p. 389–95.
48. Cowling PI, Powley EJ, Whitehouse D. Information set monte carlo tree search. IEEE Trans Comput Intell AI Games. 2012;4(2):120–43.
49. Gao C, Müller M, Hayward R. Three-head neural network architecture for monte carlo tree search. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence; 2018 Jul 13–19; Stockholm, Sweden. p. 3762–8.
50. Lucas SM, Samothrakis S, Perez D. Fast evolutionary adaptation for monte carlo tree search. In: European Conference on the Applications of Evolutionary Computation. Berlin/Heidelberg, Germany: Springer; 2014. p. 349–60.
51. Baier H, Cowling PI. Evolutionary MCTS for multi-action adversarial games. In: Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG); 2018 Aug 14–17; Maastricht, the Netherlands. p. 1–8.
52. Chaslot GMB, Winands MH, van Den Herik HJ. Parallel monte-carlo tree search. In: Proceedings of the International Conference on Computers and Games. Berlin/Heidelberg, Germany: Springer; 2008. p. 60–71.
53. Enzenberger M, Müller M. A lock-free multithreaded Monte-Carlo tree search algorithm. In: Advances in computer games. Berlin/Heidelberg, Germany: Springer; 2009. p. 14–20.
54. Barriga NA, Stanescu M, Buro M. Parallel UCT search on GPUs. In: Proceedings of the 2014 IEEE Conference on Computational Intelligence and Games; 2014 Aug 26–29; Dortmund, Germany. p. 1–7.
55. Mirsoleimani SA, Plaat A, Van Den Herik J, Vermaseren J. Scaling monte carlo tree search on intel xeon phi. In: Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS); 2015 Dec 14–17; Melbourne, VIC, Australia. p. 666–73.
56. Nayak A, Nissimagoudar P, Revankar R, Giraddi P, Hosamani S, Iyer NC, et al. Accelerating decision-making in AI: parallelizing monte carlo tree search for connect 4 using CPU and GPU. Proc Comput Sci. 2025;263:82–9. doi:10.1016/j.procs.2025.07.011.
57. Kocsis L, Szepesvári C. Bandit based monte-carlo planning. In: Proceedings of the European Conference on Machine Learning. Berlin/Heidelberg, Germany: Springer; 2006. p. 282–93.
58. Chaslot GMJ, Winands MH, Herik HJVD, Uiterwijk JW, Bouzy B. Progressive strategies for Monte-Carlo tree search. New Mathem Nat Comput. 2008;4(3):343–57. doi:10.1142/s1793005708001094.
59. Weichert D, Kister A, Volbach P, Houben S, Trost M, Wrobel S. Explainable production planning under partial observability in high-precision manufacturing. J Manufact Syst. 2023;70:514–24. doi:10.1016/j.jmsy.2023.08.009.