SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2024, Vol. 67, Iss. 5, 152204:1–152204:18
https://doi.org/10.1007/s11432-022-3904-9

Sampling-efficient path planning and improved actor-critic-based obstacle avoidance for autonomous robots

Yefeng YANG[1,2], Tao HUANG[1,2], Tianqi WANG[1], Wenyu YANG[1], Han CHEN[1], Boyang LI[4*] & Chih-yung WEN[1,3]

[1]*Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong* 999077*, China;*
[2]*Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin* 150001*, China;*
[3]*Research Center for Unmanned Autonomous Systems, The Hong Kong Polytechnic University, Hong Kong* 999077*, China;*
[4]*School of Engineering, The University of Newcastle, Callaghan* NSW 2308*, Australia*

Received 20 December 2022/Revised 2 August 2023/Accepted 18 November 2023/Published online 26 April 2024

Abstract Autonomous robots have garnered extensive utilization in diverse fields. Among the critical concerns for autonomous systems, path planning holds paramount importance. Notwithstanding considerable efforts in its development over the years, path planning for autonomous systems continues to grapple with challenges related to low planning efficiency and inadequate obstacle avoidance response in a timely manner. This study proposes a novel and systematic solution to the path planning problem within intricate office buildings. The solution consists of a global planner and a local planner. To handle the global planning aspect, an adaptive clustering-based dynamic programming rapidly exploring random tree (ACDP-RRT) algorithm is proposed. ACDP-RRT effectively identifies obstacles on the map by leveraging geometric features. These obstacles are then represented as a collection of sequentially arranged convex polygons, optimizing the sampling region and significantly enhancing sampling efficiency. For local planning, a network decoupling actor-critic (ND-AC) algorithm is employed. The proposed ND-AC simplifies the local planner design process by integrating planning and control loops into a neural network (NN) trained via an end-to-end model-free deep reinforcement learning (DRL) framework. Moreover, the adoption of network decoupling (ND) techniques leads to an improved obstacle avoidance success rate when compared to conventional actor-critic (AC)-based methods. Extensive simulations and experiments are conducted to demonstrate the effectiveness and robustness of the proposed approach.

Keywords rapidly exploring random tree (RRT), adaptive clustering, network decoupling, actor critic (AC), path planning

1 Introduction

The prompt focuses on the pressing necessity for search and rescue operations in challenging urban settings, particularly within complex office buildings, with the aim of locating and rescuing individuals who may still be alive following natural calamities. The utilization of autonomous robots as a means of aiding human rescuers in search and rescue operations is advocated due to the inherent risks faced by human personnel in such perilous environments. The introduction of autonomous robots not only enhances the safety of the rescue process but also improves overall operational efficiency. Autonomous robotic systems can reduce the workload of rescuers and provide environmental safety information [1–3]. Nevertheless, performing autonomous missions is challenging for robotic systems since they cover several different domains, for example, simultaneous localization and mapping (SLAM), task allocation, planning, and control. Robotic path planning significantly affects the performance of robots in complex environments. A complete path planning framework is usually a combination of a global planner and a local planner. The global planner provides paths or reference waypoints based on a global map. The local

* Corresponding author (email: boyang.li@connect.polyu.hk)

planner is responsible for generating a smooth trajectory and avoiding the unmapped intruding obstacles while the robots follow the global path.

The global planner algorithm systematically determines a trajectory devoid of obstacles or collisions, originating from the initial location and extending to the desired destination. Consequently, it generates a series of reference waypoints that are spatially distributed in a sequential manner, thereby representing a comprehensive global path for navigation [4]. The rapidly-exploring random tree (RRT) [5] algorithms have gained significant popularity in the domain of robot path planning owing to their robustness against high-dimensional data, conceptual simplicity, and probabilistic completeness [6]. Probabilistic completeness implies that the probability that the global planner will return a global path approaches to 1 as the number of samples or iterations increases [6].

Nevertheless, the efficiency of sampling greatly diminishes in scenarios where the environment exhibits complexity. Traditional approaches tend to demand excessive time and memory resources for a random tree to navigate elongated corridors, evade obstacles, and traverse narrow passages. To alleviate this concern, Kuffner et al. [7] introduced the RRT-connect method to accelerate the pathfinding process. RRT-connect introduces a novel strategy wherein two random trees grow simultaneously from the initial and goal configurations. The algorithm terminates upon convergence of the two trees, signifying the discovery of a path between the start and goal configurations. In [8], an asymptotically optimal method named RRT* is presented, which is based on the RRT algorithm. RRT* rewires nodes in a random tree to shorten the path length. Moreover, the RRT*-Smart [9] approach is introduced to straighten paths by reducing redundant waypoints. Li et al. [10] proposed a near-optimal RRT (NoD-RRT) motion planning framework in a clustered environment. A potential guided bidirectional RRT* method is proposed in [11] to address fast optimal path planning problems. The elastic band-based RRT (EB-RRT) [12] approach is developed for path re-planning. Qi et al. [13] introduced MOD-RRT*, which considers both path length and path smoothness and provides a higher-quality initial solution. In [14], a risk-based dual-tree RRT (risk-DTRRT) approach is proposed for optimal motion planning. Risk-DTRRT uses a rewired tree mechanism to guarantee the optimality of all heuristic trajectories. However, the aforementioned methods mainly focus on improving the growth of random trees, resulting in limited enhancement in sampling efficiency. In addition, intelligently selecting sampling regions is a promising approach to significantly improve sampling efficiency. In light of these challenges, this paper introduces a novel method named adaptive clustering-based dynamic programming-based RRT (ACDP-RRT) to efficiently tackle the global path planning problem.

The local planner pays more attention to the surroundings of robots and enables the agent to address the challenge brought by the unknown part of changes in the environment [15]. Numerous well-established algorithms, namely dynamic window approach (DWA) [16], artificial potential field (APF) [17], and time-elastic band (TEB) [18], have been introduced as viable solutions for addressing the challenge of local path planning in robotics. The DWA utilizes a dynamic window of feasible velocities to evaluate and select the trajectory for a robot's local path planning, considering factors such as collision avoidance, smoothness, and goal proximity [16]. The APF guides robot motion by employing attractive forces towards the goal and repulsive forces around obstacles, enabling path planning and obstacle avoidance based on a virtual potential field [17]. The TEB facilitates obstacle avoidance by dynamically adjusting a time-varying elastic band around the robot's planned trajectory, ensuring safe passage through narrow spaces while avoiding collisions with obstacles [18].

Nevertheless, the aforementioned approaches exhibit a drawback in terms of their sensitivity to parameters. The task of identifying appropriate hyper-parameters that guarantee optimal performance becomes challenging, particularly when dealing with complex distributions of obstacles within the environment. To deal with this issue, several control barrier functions (CBFs)-based methods are proposed [19]. In [20], a sampling-based motion planning via CBFs is proposed. The CBFs are utilized to avoid obstacles in the environment. A learning-based CBFs method for motion planning is proposed in [21]. Manjunath et al. introduced a safe and robust motion planning algorithm using CBFs. While the CBFs method has the potential to ensure both stability and safety concurrently, it is essential to acknowledge that certain inherent limitations impede its overall performance and constrain its broader application [19]. Primarily, the design process of methods based on CBFs typically entails the resolution of intricate optimization problems. Secondly, the successful implementation of CBFs methods necessitates a meticulous delineation of safety boundaries to guarantee the robot's movement remains within secure regions. In environments characterized by complexity, the precise specification of these boundaries can prove to be a challenging task, and any imprecision in their definition may lead to excessively conservative control

strategies.

To address this issue, the advent of artificial intelligence has witnessed numerous researchers employing deep reinforcement learning (DRL) as a means to tackle the local planning problem faced by robots. DRL-based local planners have garnered attention as they do not necessitate the use of explicit mathematical models for robots. Instead, these planners adopt an iterative approach wherein the current policy is continually refined through interactions with the environment. Moreover, DRL-based local planners offer an end-to-end solution by seamlessly integrating local planners and controllers within neural networks (NNs). Since 2015, when a deep Q-network (DQN) is proposed to train agents to play computer games [22], many DRL-based local planning methods have appeared. The hierarchical DRL approach presented by Shi et al. [23] is developed for drone-cell path planning and resource allocation. Peng et al. [24] proposed a DRL-GAT-SA framework to achieve safe autonomous driving and a retraining mechanism to improve efficiency. In [25], a safe RL framework with stability guarantee for robot motion planning is presented. Despite the advancements in enhancing the adaptability of robots, there are still inherent limitations associated with DRL that can have a detrimental impact on the effectiveness of local planners. The obfuscation of data is a significant factor that impedes the performance of DRL-based local planners. Data obfuscation refers to a situation where different dimensions of structured data, which are input to the neural network, represent distinct physical meanings but possess similar values. Data obfuscation highlights the limited feature extraction capability of NNs when confronted with complex data compositions. Therefore, it is vital to optimize the structure of NNs in the DRL-based algorithms to mitigate the effect of data obfuscation.

Considering the aforementioned challenges in the field of robot trajectory planning, the main contributions of this paper are shown as follows.

(1) A novel ACDP-RRT algorithm for robot global path planning is proposed. Compared with several existing algorithms, the ACDP-RRT leverages the geometric characteristics of obstacles to precisely define the sampling region and guide the growth direction of the tree structure.

(2) A novel network decoupling actor-critic (ND-AC) algorithm is introduced for robot local planning. In a pioneering approach, the utilization of network decoupling (ND) technology effectively segregates sensor data and robot motion data within the NN, leading to improved data quality. The integration of ND technology enhances the rate of obstacle avoidance, thereby improving the overall performance of the ND-AC local planner.

(3) All proposed methods are integrated into a complete autonomous navigation system for ground vehicles. Comprehensive simulations and experiments are conducted to verify the robustness and effectiveness of the proposed methods.

The remainder of this paper is organized as follows. Sections 2 and 3 present the basic principles and methodology of the ACDP-RRT global planner and the ND-AC local planner, respectively. The comparative experiments, results, and discussions are performed in Section 4. Finally, Section 5 concludes the study.

2 ACDP-RRT global planner

In this section, we propose the ACDP-RRT global planning algorithm, which includes six steps, obstacle adaptive clustering, convex hull and key point generation, distance matrix and topological map generation, path search in topological map, sampling region determination, and random tree growth, respectively. Then, the feasibility analysis of the ACDP-RRT is performed. Figure 1 illustrates the procedures of the ACDP-RRT algorithm.

The proposed ACDP-RRT represents an extension of the classical RRT algorithm. The original RRT algorithm employs a uniform sampling strategy across the entire map, which tends to be suitable for relatively simple maps without significant complexities. In such scenarios, the uniform sampling approach does not present significant challenges or obstacles to efficient path planning. However, the search efficiency of the RRT algorithm significantly diminishes in the presence of highly complex maps. This can be attributed to the propensity of the random tree's expansion trajectory to encounter obstacles and the inherent lack of foresight in the random tree's growth direction. In contrast to the RRT, ACDP-RRT incorporates the geometry information of obstacles in the map in its path planning strategy, provides a directional guidance for the growth of the random tree, and thereby improves the sampling efficiency compared to the RRT algorithm.

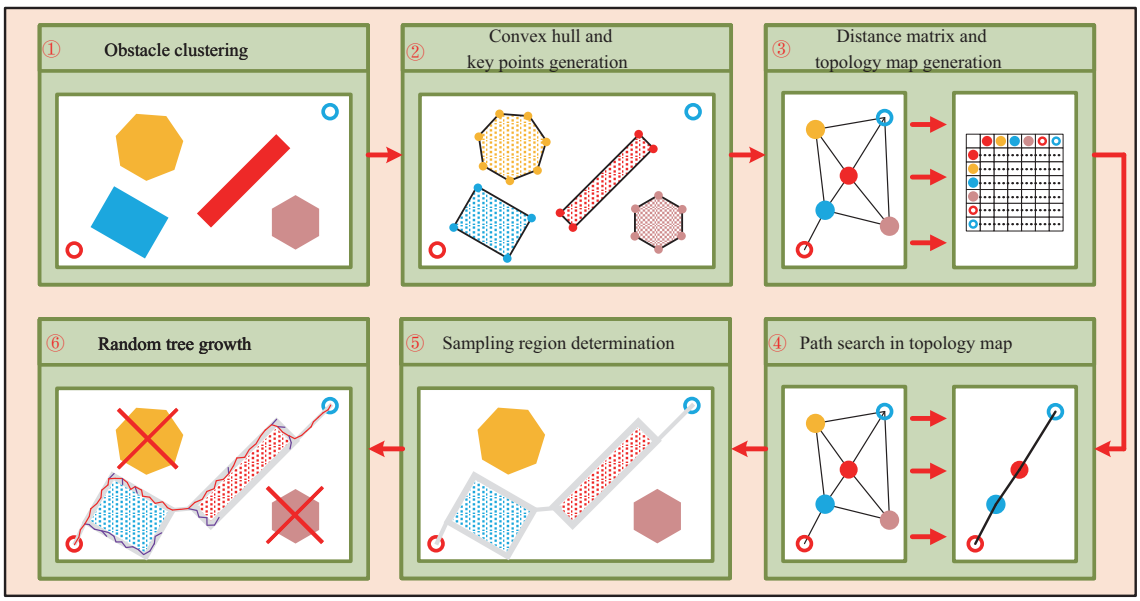

Figure 1 Schematic representation of ACDP-RRT.

Algorithm 1 Adaptive clustering [27]

Require: ϵ, N_0, obstacle space $\mathcal{X}_{\text{obs}} = \cup_{i=1}^{N} \mathcal{X}_{\text{obs}}^i$; //$\mathcal{X}_{\text{obs}}^i$ is the ith obstacle;
Ensure: Number of clusters: nc, all clusters: C;
Initialization: wa = {}, nc = 0, C = {}, C' = {};
while $\mathcal{X}_{\text{obs}}$ is not empty **do**
  Pop the first element from $\mathcal{X}_{\text{obs}}$ and push it to wa;
  while wa is not empty **do**
    Pop w, the first element of wa;
    Push w to C';
    Find all obstacles belong to the same cluster as w by using function find_$C(\mathcal{X}_{\text{obs}}, w)$;
    The result of find_$C(\mathcal{X}_{\text{obs}}, w)$ is stored in new;
    for _new $\in$ new **do**
      Push _new to wa;
    end for
  end while
  Push C' to C;
  Empty C';
  nc $+ = 1$;
end while
return nc, C.

The ACDP-RRT extracts the geometric information of the obstacles in the map. The obstacles in the map are grouped into clusters, and each cluster is bounded by a convex hull. Then, a topological map with a distance matrix is created to represent the map. The path from the initial position and the final position in the topological map with minimum cost is computed by dynamic programming (DP) [26]. Finally, the complete sampling region for the RRT is determined.

2.1 Algorithm design

(1) Obstacle adaptive clustering. Obstacles in the environment are abstracted as convex polygons. We use the DBSCAN [27] algorithm to classify obstacles on the map. Obstacle clustering is a vital module in the pre-processing of path planning. A plethora of clustering algorithms, including well-established ones such as k-means [28] and k-means++ [29], can be effectively employed to cluster obstacles. However, k-means-based methods require the number of the clusters and the initial clustering center for each cluster. In contrast, the DBSCAN is a density-based clustering algorithm, which does not require any prior information about obstacle clusters. Therefore, DBSCAN can be utilized to adaptively cluster obstacles. The DBSCAN classifies data belonging to different clusters and noise with two hyper-parameters: ϵ and N_0. ϵ is the search radius of each data point. N_0 is the minimum distance between the two clusters. Algorithm 1 illustrates the pseudo-code of the adaptive clustering process.

Algorithm 2 Distance matrix generation

Require: Key point sets of all clusters: $\mathcal{K} = \cup_{i=1}^{n} \mathcal{K}_i$;
Require: $\mathcal{K}_{n+1} = \{\mathcal{S}\}$, $\mathcal{K}_{n+2} = \{\mathcal{T}\}$; //$\mathcal{S}$ is starting position, and $\mathcal{T}$ is target position.
Ensure: Distance matrix $\boldsymbol{M}$
for $i := 1$ to $n+2$ **do**
  $\boldsymbol{M}_{i,i} = -1$; //Traversal of matrix rows.
  for $j := i+1$ to $n+2$ **do**
    for $\forall p \in \mathcal{K}_i$, $\forall q \in \mathcal{K}_j$ **do**
      $\boldsymbol{M}_{i,j} = d_{i,j}$, $\boldsymbol{M}_{j,i} = d_{j,i}$; //Traversal of matrix columns.
    end for
  end for
end for
return $\boldsymbol{M}$.

(2) Generation of convex hulls and key points. Each cluster is bounded by a circumscribed convex hull. Some vertices of the convex hull are defined as the key points of the cluster. Definition 1 defines the key points.

Definition 1 (Key points)**.** The free space of the map is defined as $\mathcal{X}_{\text{free}}$. For a convex hull $\mathcal{H}$ and the corresponding vertex set $\mathcal{V}_H$, the key point set $\mathcal{K}$ is defined as follows:

$$\mathcal{K} = \{v | v \in (\mathcal{V}_H \cap \mathcal{X}_{\text{free}})\}. \tag{1}$$

(3) Distance matrix and topological map generation. A topological map is a way of modeling a map. A topological map, which usually only contains nodes and edges, simplifies the original map, only vital information remains, and unnecessary detail has been removed. In this paper, each obstacle cluster is abstracted as a node, and the interconnection between two clusters is modeled as an edge. A distance matrix $\boldsymbol{M}$ is utilized to record the distance between different clusters together with the robot's origin and destination positions on the map. The topological map is a graphical representation of $\boldsymbol{M}$. $\boldsymbol{M}$ is a square matrix of dimension $(N+2) \times (N+2)$, where N denotes the number of clusters. Definition 2 defines the distance between cluster c_i and cluster c_j.

Definition 2 (Distance between cluster c_i and cluster c_j)**.** Let $\mathcal{K}_i$ denote the key point set of the ith obstacle cluster c_i, $\mathcal{K}_j$ denote the key point set of the jth obstacle cluster c_j, and $\mathcal{X}_{\text{obs}}$ denote the obstacle space. For all $p \in \mathcal{K}_i$, $q \in \mathcal{K}_j$, the distance between c_i and c_j is

$$d_{i,j} = d_{j,i} = \begin{cases} \min\limits_{p,q} \|p-q\|_2, & \text{line segment } pq \text{ does not pass through } \mathcal{X}_{\text{obs}}, \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

Algorithm 2 shows the pseudo-code of distance matrix generation.

(4) Topological path search in topological map. DP [26] is used to find the topological path with minimum cost in the topological map. Unlike Dijkstra [30], DP solves an optimization problem in a graph iteratively by finding optimal substructures and overlapping subproblems. Therefore, DP can save memory and computational resources for large maps [26]. The topological path consists of alternating nodes and edges. Specifically, in the topological path, a node (n) represents either an obstacle cluster, the starting position, or the target position and an edge (e) represents the connection between two clusters. The cost between the ith and jth clusters is $\boldsymbol{M}_{i,j}$ (or $\boldsymbol{M}_{j,i}$).

Noting that the path found in the topological map is not the final global path, but an abstract expression of the sampling region. Only the neighborhood of obstacle clusters and the connectors between two corresponding clusters recorded in the topological path can be determined as the sampling region. Therefore, in step (5), the topological path is correspondingly restored to the sampled regions in the map.

(5) Determination of sampling region. The complete sampling region is a combination of convex hulls and connectors after the path is found in step (4). A convex hull is a polygon used to bind an obstacle cluster. A connector (represented as an edge in the topological path) is a slender rectangle that connects two adjacent convex hulls. Furthermore, if a convex hull does not contain other obstacle cluster areas, then the convex hull is simplified into a convex hull ring.

The sampling region can be further optimized by moving the endpoints of the connectors. Without changing the path computed in step (4), the purpose of connector movement is to minimize the distance

Algorithm 3 Connector movement

Require: (1) Two adjacent connectors (edges) e_i and e_{i+1};
Require: (2) The obstacle cluster (node) between e_i and e_{i+1}: n_i;
Ensure: Updated e_1 and e_2;
Let $v = \cup_{j=1}^{N} v_j$ denote the vertices set of n_i;
Keep the first endpoint of e_i and the second endpoint of e_{i+1} fixed;
Set dis $= \infty$;
for id1 := 1 to N **do**
  for id2 := 1 to N **do**
    Set v_{id1} as the second endpoint of e_i, v_{id2} as the first endpoint of e_{i+1};
    if Neither e_i nor e_{i+1} overlap the obstacle region **then**
      if $\|v_{\mathrm{id1}} - v_{\mathrm{id2}}\|_2 <$ dis **then**
        dis $= \|v_{\mathrm{id1}} - v_{\mathrm{id2}}\|_2$;
        Update the second endpoint of e_i to v_{id1};
        Update the first endpoint of e_{i+1} to v_{id2};
      end if
    end if
  end for
end for
return e_i, e_{i+1}.

Algorithm 4 RRT

Require: x_s, x_t, $\mathcal{X}_t$, d, K, τ;
Ensure: The random tree τ;
for $i = 1$ to K **do**
  $x_{\mathrm{rand}} \leftarrow$ RANDOM_SAM();
  $x_{\mathrm{near}} \leftarrow$ NEIGHBOR(x_{rand}, τ);
  $x_{\mathrm{new}} \leftarrow$ NEW_NODE($x_{\mathrm{rand}}, x_{\mathrm{near}}, d$);
  if COLLISION_FREE($x_{\mathrm{near}}, x_{\mathrm{new}}$) **then**
    τ.add_vertex(x_{new});
    τ.add_edge($x_{\mathrm{near}}, x_{\mathrm{new}}$).
  end if
end for

between two adjacent connectors. Algorithm 3 illustrates the pseudo-code for the principle of connector movement.

(6) Random tree growth. Unlike conventional RRT-based methods, the sampling region of ACDP-RRT is not the entire free space of the map but the region generated in step (5). The tree initially grows only in the first sub-region. When the nodes of the random tree appear in the second sub-region, it terminates the growth in the first sub-region and starts the growth process in the second sub-region. This recursion process terminates when the tree appears in the last sub-region. The global planning algorithm comes to the end when the target position is contained in the random tree. Specifically, the principle of random tree growth in a sub-region is identical to that of traditional RRT.

2.2 Probabilistic completeness proof

Probabilistic completeness refers to the property wherein the likelihood of discovering a feasible solution, if one exists, tends to converge to unity as the number of sampling episodes increases [8].

Let $\mathcal{X}$ be the space of the environment. The free space in $\mathcal{X}$ is defined as $\mathcal{F}$. The obstacles in $\mathcal{X}$ are defined as $\mathcal{X}_{\mathrm{obs}}$. The start state of the robot is defined as x_s. The target state of the robot is defined as x_t. $\mathcal{B}_r(x)$ denotes the ball of radius r and centered at x. For simplicity, we assume that there exist $d_0 > 0$, and a target region $\mathcal{X}_t \in \mathcal{F}$, such that $\mathcal{X}_t = \mathcal{B}_{d_0}(x_t)$. Then, the pseudo-code of traditional RRT (Algorithm 1 in [6]) can be given by Algorithm 4.

In Algorithm 4, τ is the random tree, K is the number of iterations, and d is the distance $\|x_{\mathrm{near}} - x_{\mathrm{new}}\|$. Noting that x_{rand} is a vertex that uniformly sampled in $\mathcal{X}$, $x_{\mathrm{near}} \in \tau$ is the vertex which is nearest to x_{rand}, and x_{new} is on the line segment between x_{near} and x_{new}. In addition, function COLLISION_FREE() is utilized to check if the path from x_{near} to x_{new} is collision-free. If so, x_{new} is added as a vertex of τ and the line segment between x_{near} and x_{new} is added as an edge of τ. Otherwise, the candidate vertex x_{new} is discarded.

We then introduce a preliminary Lemma and a Theorem to serve as an auxiliary result for the probabilistic completeness proof. We assume that a branch in the random tree corresponds to a collision-free path τ from the initial point x_s to the destination x_t, the length of the path is L, the number of vertexes on the path is $p > 5L/d$, where $d < \delta_0$ and δ_0 is the minimum distance from the random tree to the

obstacles. Then, a Lemma and a Theorem can be introduced.

Lemma 1 (Lemma 1 in [6]). By the above analysis, let $x_0 = x_s$, x_1, ..., $x_p = x_t$ denote the vertexes on the path. We assume that the length between two consecutive points is less than $d/5$, namely, $\|x_i - x_{i+1}\| \leqslant d/5$ for $0 \leqslant i < p$. Suppose that RRT has reached the ith vertex x_i. Then, there must be another vertex x_i' such that $x_i' \in \mathcal{B}_d(x_i)$ (otherwise, x_i cannot exist). If a new random node x_{rand} is in the near region of x_{i+1}, namely, $x_{\text{rand}} \in \mathcal{B}_d(x_{i+1})$. Then, the line segment between x_{rand} and x_{near} in τ lies in free space $\mathcal{F}$.

Further, note the fact that $\|x_{\text{rand}} - x_{\text{near}}\| \leqslant \|x_{\text{rand}} - x_i'\| \leqslant \|x_i' - x_i\| + \|x_i - x_{i+1}\| + \|x_{i+1} - r_{\text{rand}}\| \leqslant 3 \cdot \frac{d}{5} < d$, which means that $x_{\text{rand}} = x_{\text{new}}$. Then, a theorem can be given as follows.

Theorem 1 (Theorem 1 in [6]). Based on Lemma 1, the probability that RRT fails to reach $\mathcal{X}_t$ after k iterations is at most $\frac{p}{p-1}k^p \mathrm{e}^{-p_0 k}$, namely,

$$\Pr[X_k < p] \leqslant \frac{p}{p-1}k^p \mathrm{e}^{-p_0 k},$$

where Pr is the probability that RRT fails to reach the goal after k iterations, p is the number of vertexes on the path, and p_0 is the probability that x_{rand} falls into the ball $\mathcal{B}_d(x_i)$.

Theorem 1 indicates that the probability decays to zero exponentially with k. In the case of ACDP-RRT, the comprehensive sampling region is partitioned into multiple concatenated subregions, interspersed with connectors and convex hulls (or convex hull rings). The following observations can be made:

(1) The principle underlying the proposed ACDP-RRT implies that two adjacent sampling areas are bound to have overlapping segments.

(2) Based on the principle of ACDP-RRT, the presence of obstacles within certain sub-sampling areas does not compromise the connectivity between adjacent areas.

Therefore, based on the analysis presented, it can be inferred that all sampling sub-regions are interconnected. Considering that the random tree generation process of ACDP-RRT is identical to traditional RRT. Then, we can conclude that ACDP-RRT does not violate the probabilistic completeness of RRT; namely, ACDP-RRT is proven to be probabilistically complete.

3 ND-AC local planner

In this section, we present the ND-AC local planning algorithm. ND-AC represents an advancement over the classical AC-based method. Both AC and ND-AC share a common learning framework. However, the traditional AC method primarily focuses on the agent's learning process within the environment, neglecting the significance of data pre-processing. Consequently, when structured data contains an excessive amount of information, the quality of neural network learning tends to deteriorate in traditional AC. The introduction of a network decoupling pre-processing module enhances the training performance of ND-AC in comparison to traditional AC methods. By decoupling the network's input processing, ND-AC is able to effectively handle different aspects of the input data independently. This approach improves the network's ability to extract meaningful information from the input, resulting in enhanced training performance and ultimately better overall performance when compared to traditional AC algorithms.

3.1 AC framework

The basic local planning learning framework utilized in this study is AC [31], which integrates the policy gradient (PG) [32] and value iteration (VI) [33] methods. The proposed AC allows the advantages of PG and VI to coexist in DRL. The AC simultaneously updates the policy network (actor) and value network (critic). The Actor in the local planner is the robot's controller who can avoid obstacles. The critic in this context refers to an NN-based cost function, which serves the purpose of evaluating and quantifying the performance of the actor. Figure 2 illustrates the relationship between the AC framework and the local planner.

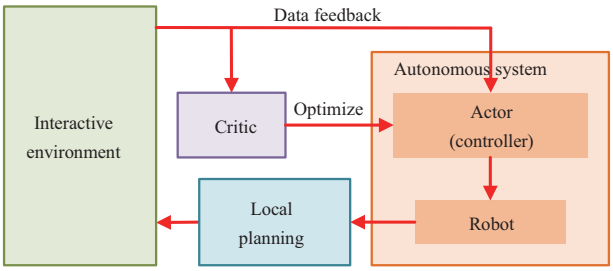

Figure 2 Relationship between AC framework and the local planner.

The objective of the actor is to maximize the expectation of cumulative reward:

$$\begin{aligned}\eta(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[G_t|S=s,A=a\right]\\ &= \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right]\\ &= \sum_{s\in S} d^{\pi_\theta}(s)\sum_{a\in A}\pi_\theta(a|s)Q_{\pi_\theta}(s,a;\theta),\end{aligned} \tag{3}$$

where $\eta(\theta)$ is the objective function, π_θ is the policy parameterized by θ, s is the state, a is the action, τ is a trajectory $(s_1,a_1,s_2,a_2,\ldots,s_n,a_n,\ldots)$ induced by π_θ, $R(\tau)$ is the reward function, $d^{\pi_\theta}(s)$ is the distribution of the state space with the current policy π_θ, $Q_{\pi_\theta}(s,a;\theta)$ is the state-action value function of the tuple $\langle s,a\rangle$, and θ is the parametric vector of the actor network in the AC framework.

According to policy gradient theory (Theorem 1 in [32]), the gradient of $J(\theta)$ with respect to θ is given by

$$\begin{aligned}\nabla_\theta\eta(\theta) &= \nabla_\theta\sum_{s\in S} d^{\pi_\theta}(s)\sum_{a\in A}Q_{\pi_\theta}(s,a)\pi_\theta(a|s)\\ &= \sum_{s\in S} d^{\pi_\theta}(s)\sum_{a\in A}Q_{\pi_\theta}(s,a)\nabla_\theta\pi_\theta(a|s)\\ &= \sum_{s\in S} d^{\pi_\theta}(s)\sum_{a\in A}\pi_\theta(a|s)Q_{\pi_\theta}(s,a)\frac{\nabla_\theta\pi_\theta(a|s)}{\pi_\theta(a|s)}.\end{aligned} \tag{4}$$

It follows

$$\nabla_\theta\eta(\theta) = \mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s,a)\nabla_\theta\ln\pi_\theta(a|s)\right]. \tag{5}$$

The actor net is introduced to approximate policy $\pi_\theta(a|s)$. A critic net is unitized to approximate the state-action value function $Q_\pi(s,a)$. The objective of the critic net is to maximize the state-action value function $Q_\pi(s,a)$ by minimizing the temporal difference (TD) error. The formula for the TD error δ_t is given by

$$\delta_t = r_{t+1} + \gamma Q_{\pi_\theta}(s_{t+1},a_{t+1}) - Q_{\pi_\theta}(s,a), \tag{6}$$

where r_{t+1} is the immediate reward of step $t+1$ and δ_t is the loss function of the critic network whose gradient can be computed automatically by the deep learning toolbox (DLT). Algorithm 5 provides the pseudo-code of the one-step training of AC, where τ is the soft update rate, and the learn() is automatically realized by DLT.

3.2 Network decoupling technology

Theoretically, NNs consisting of more than two layers have the capacity to approximate functions with arbitrary complexity. However, the practical performance of such approximations is not always reliable due to limitations in data quality and the possibility of the optimizer getting trapped in local minima. To address these challenges, the utilization of ND techniques is employed to enhance the performance of NN learning. By decoupling the network's input processing, ND aids in mitigating the adverse effects of insufficient data quality and local minima, thereby improving the overall performance and reliability of NN-based function approximation.

Algorithm 5 One-step learning of AC

Require: (1) Target actor network: $\pi(s;\theta)$, evaluation actor network: $\pi'(s;\theta')$;
Require: (2) Target critic network: $q_\pi(s,a;\omega)$, evaluation critic network: $q'_\pi(s,a;\omega')$;
Ensure: Updated $\pi(s;\theta)$, $\pi'(s;\theta')$, $q_\pi(s,a;\omega)$, and $q'_\pi(s,a;\omega')$;
Get action a_{t+1} at time step $t+1$: $a_{t+1} \leftarrow \pi(s_{t+1};\theta_t)$;
Get values Q_{t+1} and Q_t at time step $t+1$ and t: $Q_{t+1} \leftarrow q_\pi(s_{t+1},a_{t+1};\omega_t)$, $Q_t \leftarrow q'_\pi(s_t,a_t;\omega'_t)$;
Get reward r_{t+1} at time step $t+1$ from the environment;
critic net training: //Computed automatically by DLT;
  loss $= \frac{1}{2}(r_{t+1}+\gamma Q_{t+1}-Q_t)^2$;
  Compute ω'_{t+1};
actor net training: //Computed automatically by DLT;
  loss $= -q'_\pi(s_t,\pi'(s_t;\theta'_t);\omega'_t)$;
  Compute θ'_{t+1};
parameter update
  $\omega_{t+1} = \tau\omega_t + (1-\tau)\omega'_{t+1}$;
  $\theta_{t+1} = \tau\theta_t + (1-\tau)\theta'_{t+1}$;
return $\pi(s;\theta)$, $\pi'(s;\theta')$, $q_\pi(s,a;\omega)$, and $q'_\pi(s,a;\omega')$.

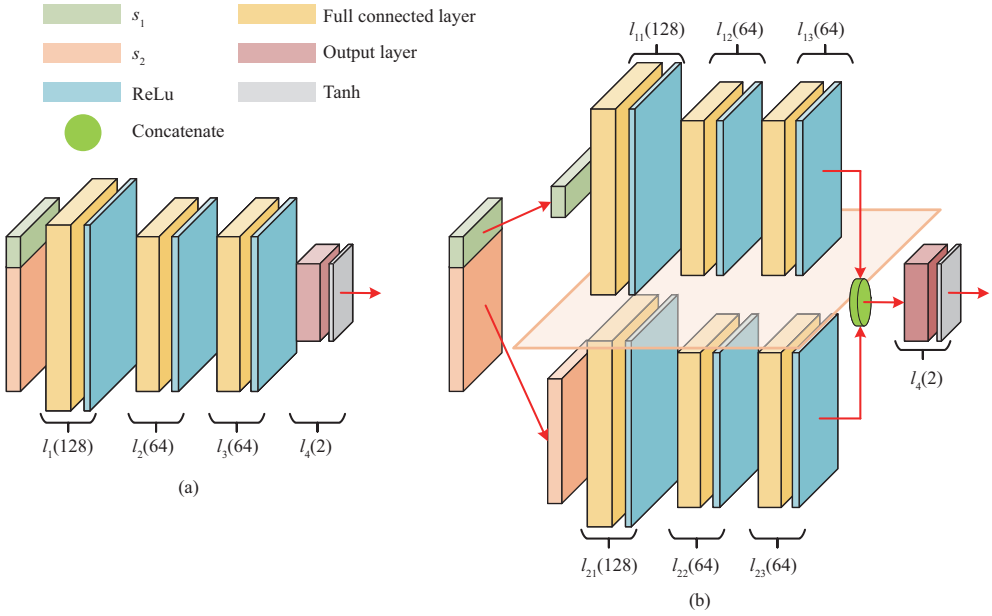

Figure 3 Comparison of the actor networks in (a) the original learning framework and (b) the decoupled learning framework.

Figure 3 is a comparative demonstration of the original actor network and the decoupled actor network. The input data of the NN, namely, the state space of the autonomous robot, covers two parts: motion part (s_1 in Figure 3) and sensor part (s_2 in Figure 3). l_1, l_2, and l_3 are fully connected hidden layers with the ReLu activation function. l_4 is the output layer with the tanh activation function. The number in each bracket behind l. represents the number of neurons in the corresponding layer. The green circular node means the concatenation of the outputs of l_{13} and l_{23}.

The physical meanings of these two parts are different. The sensor-related data represents the spatial distribution of obstacles around the robot. In contrast, the motion-related data covers information about the robot's position, orientation, and velocity. The transmission of motion-related data and sensor-related data is impeded through the implementation of ND technology. Consequently, the two distinct sub-networks are capable of independently extracting their own distinctive features without being obscured by other components. The concatenation of the networks takes place prior to the final hidden layer. Ultimately, the output of the actor network is equally influenced by both the motion and lidar components.

4 Simulations and experiments

This section contains comparative simulations and real-world experiments to demonstrate the robustness and effectiveness of the proposed global and local planner. Figure 4 shows the flowchart of the systemic path planning framework for a real autonomous robot system.

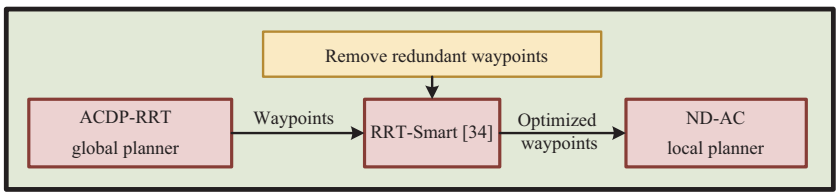

Figure 4 Data flow diagram of the systemic path planning framework.

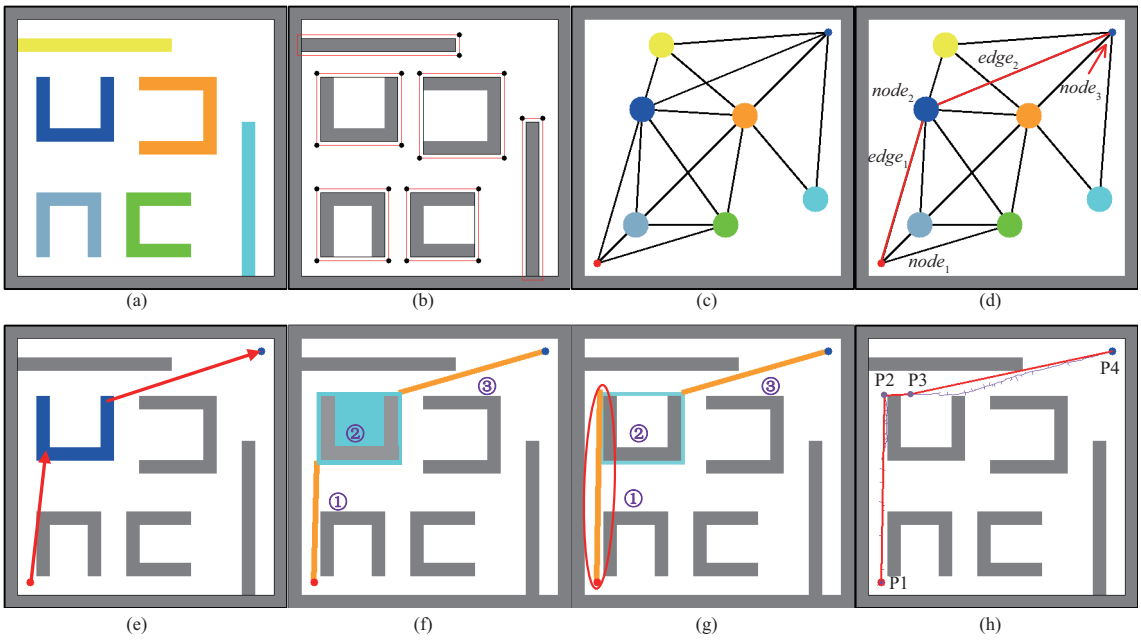

Figure 5 Global planning process of the ACDP-RRT. (a) Step 1; (b) Step 2; (c) Step 3; (d) Step 4; (e) Step 4; (f) Step 5; (g) Step 5; (h) Step 6.

As shown in Figure 4, the complete path planning task is divided into two parts, global planning and local planning. In addition, the RRT-Smart technique is utilized as a bridge between the global planner and the local planner. To begin with, using the ACDP-RRT global planner, the autonomous robot generates a global path represented by sequentially distributed nodes. Nevertheless, the path exhibits an excessive number of redundant nodes, thereby imposing an augmented burden on the local planner. Consequently, the application of the RRT-Smart methodology becomes imperative in order to effectively eliminate such redundant nodes and optimize the path planning process. RRT-Smart interconnects the furthest node on the path if the node is directly visible by the current node. The detailed optimizing process of RRT-Smart can be found in Algorithm 2 in [34]. After optimization by using RRT-Smart, the number of nodes on the global path is greatly reduced. The nodes encompassed within the optimized path are considered the intended targets for the local planner. Subsequently, the robot progressively advances towards each node in a sequential manner until it enters the vicinity of the final node.

4.1 Simulation results of the global planner

Figure 5 demonstrates the detailed process of the global planning. In Figure 5(a), six clusters are generated and obstacles belonging to different clusters are marked with different colors. After that, six convex halls (the red rectangles in Figure 5(b)) and the corresponding key points (the black solid points in Figure 5(b)) are generated. Additionally, all convex hulls are extended outward by half the size of the robot to ensure that the robot can pass through the gaps between obstacle clusters. Then, as shown in Figure 5(c), the topological map abstracted from Figure 5(b) is developed. In Figures 5(d) and (e), the topological path (the red lines) is generated in the topological map. Based on the result in Step 4, the sampling region is determined in Step 5. The sampling region shown in Figure 5(f) is the combination of one convex hull (the cyan region in Figure 5(f)) and two connectors (the orange rectangles in Figure 5(f)). The simplified sampling region is shown in Figure 5(g), which is, the convex hull is simplified as a convex

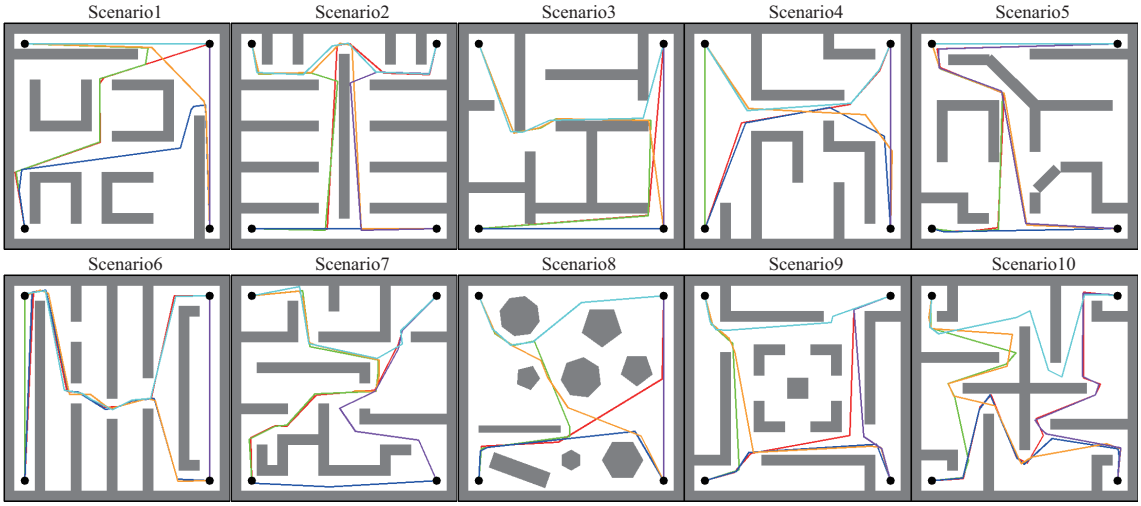

Figure 6 Planning results of ACDP-RRT. Different paths are indicated by different colors.

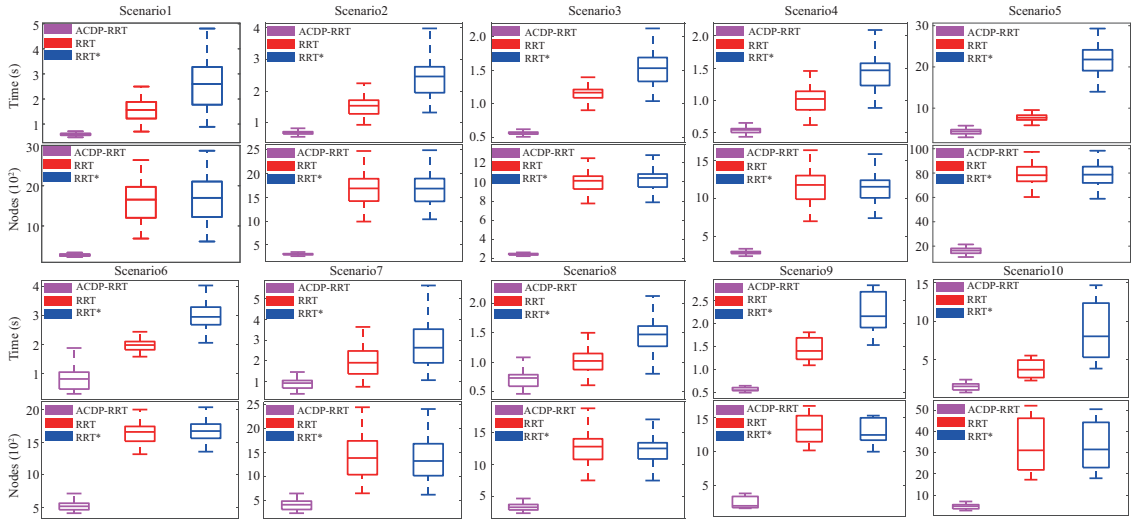

Figure 7 Evaluation of the time and memory efficiency for each algorithm in scenarios (1)–(10).

hull ring. Finally, based on Step 5, the global path containing four waypoints (P1–P4 in Figure 5(h)) is generated.

Figure 6 shows the planning results in different scenarios using ACDP-RRT. We also compare our proposed ACDP-RRT algorithm with two RRT-based planning algorithms, RRT and RRT*. For each office scenario shown in Figure 6, we choose any two of the four corners of the room as a (start, target) tuple. The dimension of the office scenarios is 11 m × 11 m. The locations of the four corners are (1 m, 1 m), (1 m, 10 m), (10 m, 1 m), and (10 m, 10 m). The number of (start, target) tuples is $C_6^2 = 12$. Each tuple in each scenario is executed 50 times. We use the time consumption and the number of nodes in the random tree when the paths are generated to evaluate the time and memory efficiency of each algorithm.

Figure 7 shows the efficiency of each algorithm for all ten scenarios. The box chart denotes the standard variation and the line in the box represents the mean value of the 600 simulation results. The top subfigure for each scenario is the time efficiency, and the bottom shows the memory efficiency of each algorithm. The ten figures clearly show that the performance of the proposed ACDP-RRT algorithm is much better than that of the conventional RRT and RRT*.

The time efficiency can be determined by comparing the top ten sub-figures in Figure 7. Although RRT* is an improved version of RRT, its main contributions focus on the path length and the proof of asymptotic optimality. However, path length is not an evaluation criterion in this study because RRT-Smart [34] technology is used after path generation, which means the length of the global path of a non-sensitive element. In general, the time efficiency of RRT* is lower than that of RRT because

the latter requires tree re-connecting within the neighborhood of new nodes. The time efficiency of the ACDP-RRT is significantly higher than that of the other two algorithms in all scenarios. The ACDP-RRT searches the map more directionally, and no collision detection is required when the tree is growing in connectors, saving further computational resources.

The memory efficiency is shown in the bottom ten sub-figures of each scenario in Figure 7. The figures reveal that the number of nodes is similar when RRT and RRT* generate a global path. ACDP-RRT requires only $\frac{1}{5}$ to $\frac{1}{2}$ nodes to generate a global path. The reason why the time efficiency improvement is smaller than the memory efficiency improvement is that ACDP-RRT requires some pre-processing before it starts searching the map.

4.2 Simulation results of the local planner

In this subsection, we conduct the performance evaluation of the proposed ND-AC local planner utilizing a two-wheel differential mobile robot as the chosen platform. This deliberate selection of the robot allows for the comprehensive validation and assessment of the ND-AC local planner's efficacy and applicability in real-world scenarios. Furthermore, the utilization of a two-wheel differential ground vehicle offers the advantage of possessing nonholonomic characteristics. Specifically, the vehicle's steering capabilities are limited to adjusting the speed difference between its two wheels, precluding the ability to move freely in any direction akin to an omnidirectional robot like a four-wheeled Mecanum wheel robot. The inherent nonholonomic nature of the chosen platform renders the local planner's design notably challenging, thereby underscoring the superior capabilities of the ND-AC-based local planner.

(1) Dynamic model for two-wheeled differential robot. A dynamic model of the two-wheeled robot is required for local planning. As in [35], the dynamic model is given by

$$\begin{cases} \dot{x} = \dfrac{r}{2}\left(\omega_{\mathrm{L}} + \omega_{\mathrm{R}}\right)\cos\varphi, \\ \dot{y} = \dfrac{r}{2}\left(\omega_{\mathrm{L}} + \omega_{\mathrm{R}}\right)\sin\varphi, \\ \dot{\varphi} = \dfrac{r}{r_b}\left(\omega_{\mathrm{L}} - \omega_{\mathrm{R}}\right), \end{cases} \tag{7}$$

where $[x, y]$ is the location, φ is the yaw angle, r is the radius of the wheel, r_b is the radius of the robot base, and $[\omega_{\mathrm{L}}, \omega_{\mathrm{R}}]$ represents the rotation speeds of the left and right wheels, respectively. In the training environment, we limit the maximum linear velocity to $V_{\max}$ and the maximum wheel speed to $\omega_{\max}$ to match the motion capability of the robot. The action space is given by

$$a = [\omega_{\mathrm{L}}, \omega_{\mathrm{R}}]. \tag{8}$$

The state variable is a 45-dimensional vector given by

$$s = [\tilde{e}_x, \tilde{e}_y, \tilde{x}, \tilde{y}, \varphi, \dot{x}, \dot{y}, \dot{\varphi}] + \mathrm{Lidar}(), \tag{9}$$

where $\tilde{e}_x = k(t_x - x)/X$, $\tilde{e}_y = k(t_y - y)/Y$, $\tilde{x} = kx/X$, and $\tilde{y} = ky/Y$ are normalized to improve the generalization ability of the learned policy, $[t_x, t_y]$ is the location of the target, X and Y are the dimensions of the map, $[x, y]$ is the location of the robot, k is a static gain, and Lidar() is a 37-dimensional vector down-sampled from the raw data of the lidar in intervals $[0°, 90°]$ and $[270°, 360°]$ with a resolution of $5°$.

(2) Learning framework setup. The training environment for the proposed DRL method is implemented on a computer equipped with an i7-11700 CPU and NVIDIA-RTX 2060 GPU. The software platform used is Ubuntu 20.04, ROS Noetic, and PyTorch 1.8.1. Figure 8 shows an illustration of the learning environment.

The reward function consists of position reward r_1, orientation reward r_2, and sparse reward r_3. r_1 is a positive value ($= 5$) if the position error decreases; otherwise, it is negative ($= -5$). r_2 is a positive value ($= 2$) if the angular error decreases; otherwise, r_2 is a negative value ($= -2$). r_3 is set to be positive ($= 10$) if the vehicle reaches the destination successfully; otherwise, r_3 is set to zero. The complete formation of the reward function is the sum of the following three parts:

$$r = r_1 + r_2 + r_3. \tag{10}$$

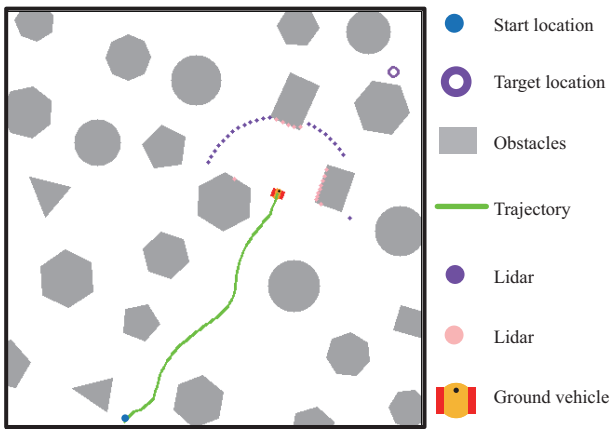

Figure 8 Illustration of the learning environment. The purple lidar detection points mean that no obstacles are detected at the corresponding angle. The pink lidar detection points indicate the obstacles are detected at the corresponding angle.

Table 1 Hyper-parameters of the learning framework

Parameter	Value	Parameter	Value
γ	0.99	N	60000
up_C	0.01	up_A	0.01
noise	0.25	noise_clip	0.5
lr_c	1E−3	lr_a	1E−4
batch	512	n_d	5

The three parts of the immediate reward function r are given by

$$\begin{cases} r_1 = \begin{cases} 5, & \text{positional error is smaller,} \\ -5, & \text{otherwise,} \end{cases} \\ r_2 = \begin{cases} 2, & \text{angular error is smaller,} \\ -2, & \text{otherwise,} \end{cases} \\ r_3 = \begin{cases} 0, & \text{collision,} \\ 10, & \text{otherwise.} \end{cases} \end{cases} \tag{11}$$

In each episode, the agent explores different maps generated by a stochastic map generator to improve the generalization of the learned policy. The obstacles in the map are convex polygons or circles. Table 1 lists some hyperparameters of the proposed learning framework. γ is the discount factor, up_C is the soft update rate of the critic net, up_A is the soft update rate of the actor net, noise and noise_clip are the variance and maximum value of the Gaussian distributed noise, lr_c and lr_a are the learning rates of the critic net and actor net, N is the capacity of the replay buffer, batch is the number of samples that the NN takes each time, and n_d is the number of steps by which the TD3 actor network delays the update.

Figure 9 shows the complete learning process of the proposed DRL method. The learning framework consists of three modules: an environment module, a learning module, and a data set module. The environment module receives the action of the agent from the learning module and updates the states of the mobile robot. The environment module sends a tuple (s, a, r, s') to the dataset. The learning module simultaneously updates the actor and critic networks by sampling a tuple batch from the data set at each time step. A recursion process is constructed for the three modules.

(3) Policy learning and simulation. We evaluate the performance of the proposed ND technology by comparing the success rate of the mobile robot in reaching the destination location on the maps. To evaluate the performance of the proposed ND technology, we implement two state-of-the-art AC-based DRL methods: TD3 [36] and DDPG [37]. Figure 10 illustrates the success rate and immediate reward during the training process.

Figure 10 shows that the success rate is higher when the net decoupling mechanism is integrated into the training framework. It is plausible that the success rate of DDPG is lower than that of TD3 because TD3 uses a double network, actor-network delayed update, and target-policy smoothing regularization

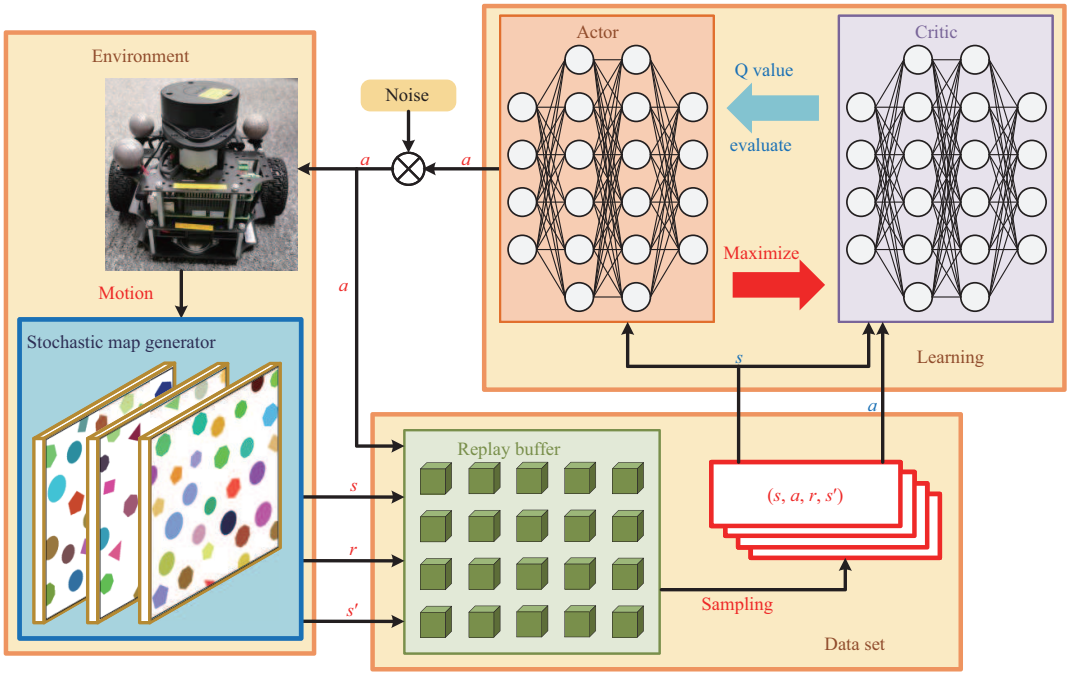

Figure 9 Complete DRL-based local planner learning framework.

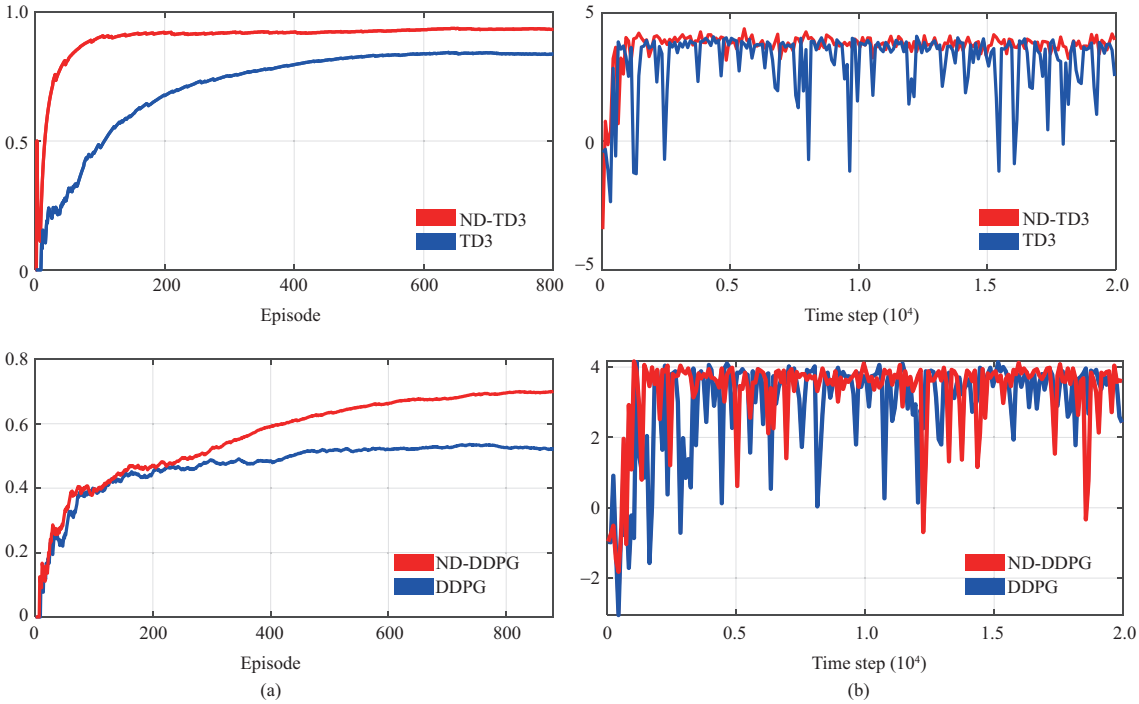

Figure 10 Training processes of two AC-based DRL algorithms: TD3 and DDPG. (a) Success rates of the robot in obstacle avoidance; (b) immediate rewards.

mechanism to mitigate the overestimation of the critic network as well as improve the smoothness of the learned policy. The corresponding figure for the immediate reward also shows the effectiveness of the ND mechanism. The immediate rewards of the planning algorithms with the ND mechanism are higher than those of the original algorithms. The immediate reward figure fluctuates because Gaussian distributed noise is added to both the hidden layers and the output of the actor network to realize a complete exploration of the state space. Therefore, we select the algorithm with a higher success rate, TD3, as the basic method for the proposed ND-AC local planner for real-world experiments.

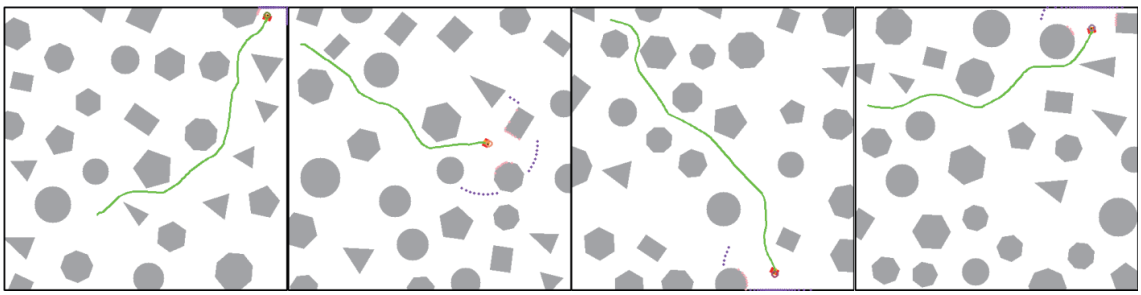

Figure 11 Graphical demonstration of local planning.

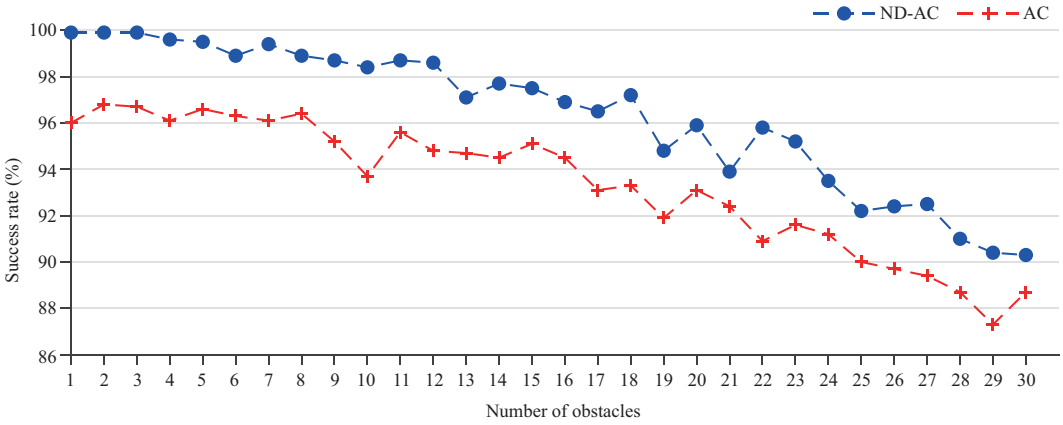

Figure 12 Comparative experiments on the success rates of AC local planner and ND-AC local planner in maps with a different number of obstacles.

In Figure 11, we list four examples of successful obstacle avoidance to give a clear graphical representation of local planning.

In addition, sufficient Monte Carlo experiments are conducted to support the conclusion of the proposed ND technology and to demonstrate the capabilities of the well-trained ND-AC local planner. We conduct 1000 Monte Carlo experiments on maps containing 1–30 obstacles to evaluate the performance of the AC and ND-AC local planners. The starting positions, target positions, shapes, sizes, and locations of the obstacles for each Monte Carlo experiment are stochastically initialized to guarantee the sufficiency of the experiment. Figure 12 illustrates the success rates of AC and ND-AC local planners in maps with varying numbers of obstacles. It is noteworthy that the success rates of the ND-AC and original AC local planners in the empty world are 0.999 and 0.987, respectively.

Figure 12 clearly shows that the performance of the ND-AC local planner is better than that of the original AC local planner. The success rate remains above 90% when 30 obstacles are placed on the map. The effectiveness and robustness of the proposed ND-AC local planner are corroborated by 30000 episodes of Monte Carlo simulation experiments.

4.3 Simulation and experiments with the integrated planner

A complete path planning framework is a combination of global and local planning. In this subsection, simulation and experimental results of combining the proposed global and local planning algorithms are presented. A NanoRobot is used (Figure 13) to test the trained local planner. The processor of the NanoRobot is a Raspberry Pi 4B (CPU frequency: 1.5 GHz) integrated with Ubuntu 20.04, ROS Noetic, and PyTorch 1.8.1. The sensors of the NanoRobot comprise an rplidar-SUPER single-wire lidar installed on the top of the body and two encoders connected to the motors. The markers on the body are used to position the robot in the VICON motion capture environment. As mentioned earlier, the maximum linear and angular velocities are $V_{\max} = 0.7$ m/s and $\omega_{\max} = 10$ rad/s, respectively. The detection range of the lidar is 8 m. We reduce the distance value of the lidar in the learning environment to [0.15 m, 2 m]. After testing, 2 m is enough for obstacle avoidance in both simulation and experiments. Furthermore, the minimum detection range is set to be 0.15 m to match the hardware properties of the lidar.

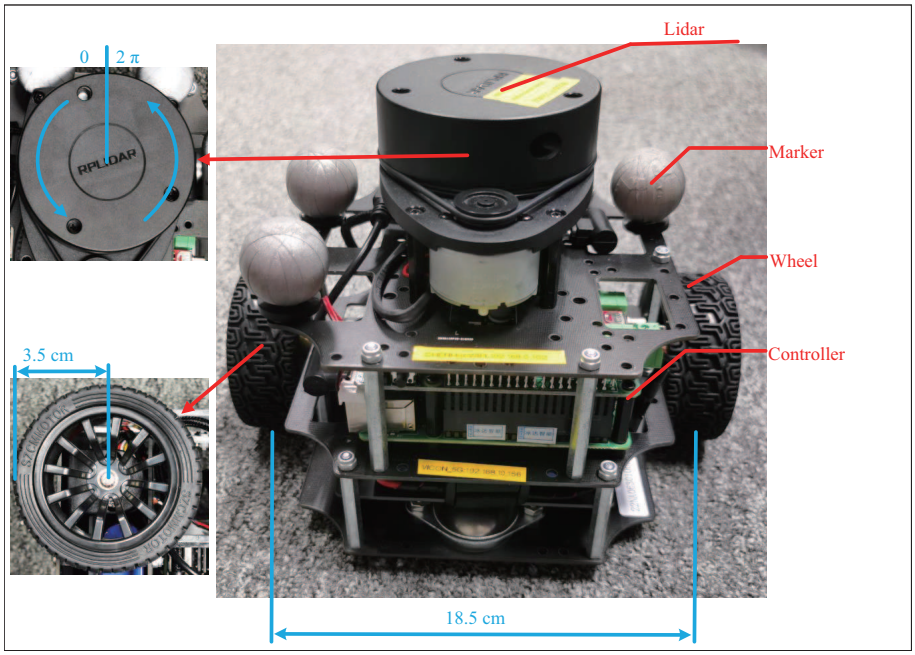

Figure 13 NanoRobot platform equipped with a single wire lidar.

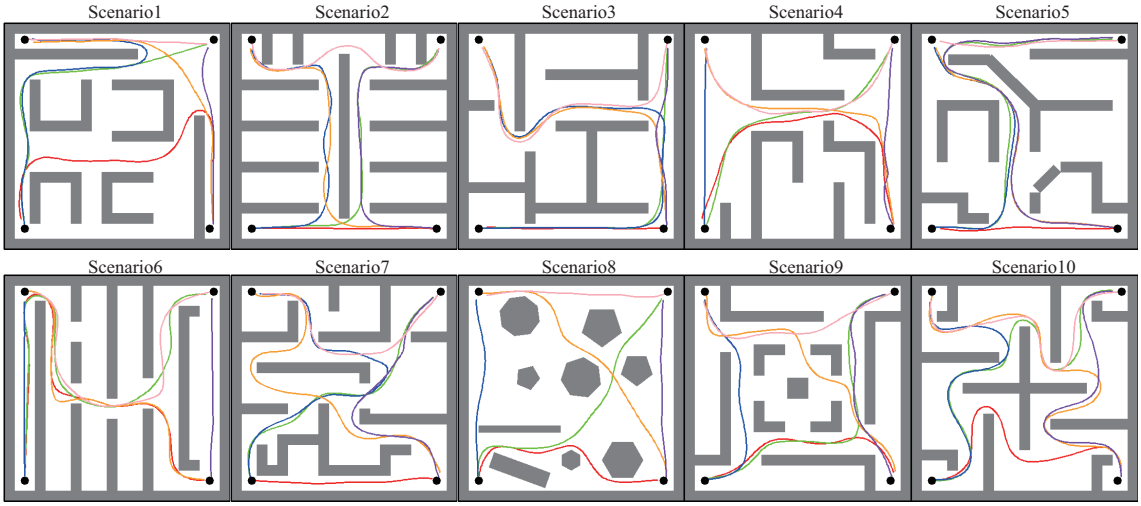

Figure 14 Demonstrations of complete path planning simulation experiments. The trajectories between different starting and ending points are highlighted in different colors.

(1) Simulation results. Four corners (1 m, 1 m), (1 m, 10 m), (10 m, 10 m), and (10 m, 1 m) are set as the start and target positions, respectively. Ten different groups of simulations are performed for each map. The complete path planning simulation results for different scenarios are shown in Figure 14.

(2) Real-world experiment results. Finally, experiments are conducted in real-world scenarios to further validate the performance of the proposed methods. The scale of the physical and simulation scenarios is 1 : 2.5. We use six office scenarios similar to the scenarios created in the simulation. Figure 15 shows snapshots of the complete path planning results. The first and second rows are the official maps of the simulation and natural world. The last row shows the actual trajectories of the mobile robot. The results indicate that the mobile robot follows the waypoints smoothly and avoids obstacles successfully. The demonstrations reveal the effectiveness and robustness of the ACDP-RRT global planner and the ND-AC local planner. Besides, the runtime of the local planner on the onboard Raspberry Pi is less than 2 ms. ND-AC local planner, as a compact end-to-end solution, only involves very light calculations when running onboard, which is a significant superiority over other classical motion planning methods. Therefore, thanks to the DRL-based local planner, the complete navigation framework effectively tackles

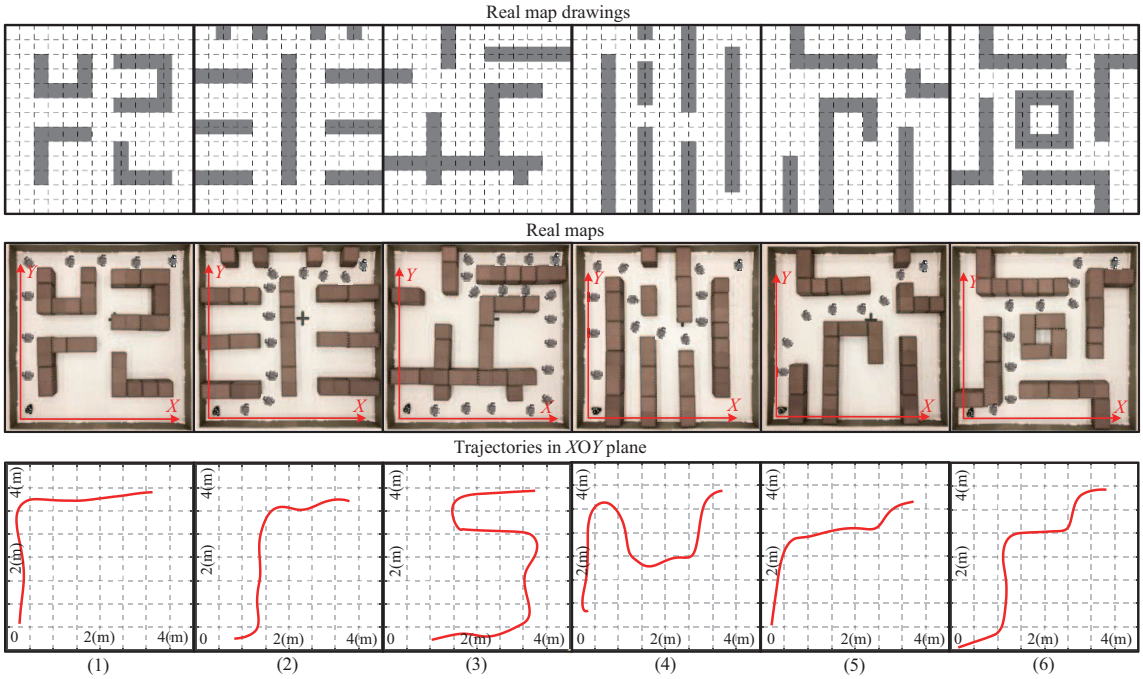

Figure 15 Physical experiments of the corresponding six office building scenarios. The dimension of the grids shown in the maps is 0.3 m × 0.3 m. The perimeter is made of plastic plates with a thickness of 1 cm. Opaque rough tapes are attached to the surfaces of the plastic boards to ensure the quality of the lidar data. The initial position is the bottom left corner and the target position is the top right corner.

environmental changes to ensure safety.

5 Conclusion

This paper presents an innovative trajectory planning framework designed for autonomous robots. To address global planning, an ACDP-RRT global planner is introduced, which efficiently generates global waypoints for local planning. In comparison to conventional global planning algorithms, the ACDP-RRT approach enhances sampling efficiency and reduces memory costs by intelligently determining the sampling region within the environment. For local planning, an ND-AC-based local planner is employed, which can iteratively learn an approximately optimal policy that integrates both the robot's controller and planner. The integration of ND technology not only expedites the training process of the AC framework but also enhances the robot's success rate in obstacle avoidance. Real-world experiments validate the robustness and effectiveness of the entire planning framework. In our future endeavors, we intend to pursue two main research directions. Firstly, we will focus on investigating path planning methodologies tailored to scenarios with unknown or partially known maps. Secondly, we will explore the implementation and adaptation of the proposed path planning algorithms within a multi-agent system context.

Acknowledgements This work was supported by Research Center of Unmanned Autonomous Systems (RCUAS), The Hong Kong Polytechnic University (Grant No. P0046487).

References

1 Niroui F, Zhang K C, Kashino Z, et al. Deep reinforcement learning robot for search and rescue applications: exploration in unknown cluttered environments. IEEE Robot Autom Lett, 2019, 4: 610–617
2 Hou X L, Li Z Y, Pan Q. Autonomous navigation of a multirotor robot in GNSS-denied environments for search and rescue. Sci China Inf Sci, 2023, 66: 139203
3 Kamegawa T, Akiyama T, Sakai S, et al. Development of a separable search-and-rescue robot composed of a mobile robot and a snake robot. Adv Robotics, 2020, 34: 132–139
4 Ma T, Zhou H B, Qian B, et al. A large-scale clustering and 3D trajectory optimization approach for UAV swarms. Sci China Inf Sci, 2021, 64: 140306
5 Wang J K, Chi W Z, Li C M, et al. Neural RRT*: learning-based optimal path planning. IEEE Trans Automat Sci Eng, 2020, 17: 1748–1758
6 Kleinbort M, Solovey K, Littlefield Z, et al. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. IEEE Robot Autom Lett, 2019, 4: 1–7

7 Kuffner J J, LaValle S M. RRT-connect: an efficient approach to single-query path planning. In: Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 2000. 995–1001
8 Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning. Int J Robotics Res, 2011, 30: 846–894
9 Nasir J, Islam F, Malik U, et al. RRT*-Smart: a rapid convergence implementation of RRT*. Int J Adv Robotic Syst, 2013, 10: 299
10 Li Y, Cui R X, Li Z J, et al. Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT. IEEE Trans Ind Electron, 2018, 65: 8718–8729
11 Tahir Z, Qureshi A H, Ayaz Y, et al. Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. Robotics Autonomous Syst, 2018, 108: 13–27
12 Wang J K, Meng M Q H, Khatib O. EB-RRT: optimal motion planning for mobile robots. IEEE Trans Automat Sci Eng, 2020, 17: 2063–2073
13 Qi J, Yang H, Sun H X. MOD-RRT*: a sampling-based algorithm for robot path planning in dynamic environment. IEEE Trans Ind Electron, 2021, 68: 7244–7251
14 Chi W Z, Wang C Q, Wang J K, et al. Risk-DTRRT-based optimal motion planning algorithm for mobile robots. IEEE Trans Automat Sci Eng, 2019, 16: 1271–1288
15 Xi L L, Peng Z H, Jiao L, et al. Smooth quadrotor trajectory generation for tracking a moving target in cluttered environments. Sci China Inf Sci, 2021, 64: 172209
16 Chang L, Shan L, Jiang C, et al. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Auton Robot, 2021, 45: 51–76
17 Huang Y J, Ding H T, Zhang Y B, et al. A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach. IEEE Trans Ind Electron, 2019, 67: 1376–1386
18 Zhang Y Z, Ma B, Wai C K. A practical study of time-elastic-band planning method for driverless vehicle for auto-parking. In: Proceedings of the International Conference on Intelligent Autonomous Systems, Singapore, 2018. 196–200
19 Ames A, Coogan S, Egerstedt M, et al. Control barrier functions: theory and applications. In: Proceedings of the 18th European Control Conference (ECC), Naples, 2019. 3420–3431
20 Yang G, Vang B, Serlin Z, et al. Sampling-based motion planning via control barrier functions. In: Proceedings of the 3rd International Conference on Automation, Control and Robots, Beijing, 2019. 22–29
21 Saveriano M, Lee D. Learning barrier functions for constrained motion planning with dynamical systems. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macao, 2019. 112–119
22 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
23 Shi W S, Li J L, Wu H Q, et al. Drone-cell trajectory planning and resource allocation for highly mobile networks: a hierarchical DRL approach. IEEE Internet Things J, 2021, 8: 9800–9813
24 Peng Y F, Tan G Z, Si H W, et al. DRL-GAT-SA: deep reinforcement learning for autonomous driving planning based on graph attention networks and simplex architecture. J Syst Architecture, 2022, 126: 102505
25 Zhang L X, Zhang R X, Wu T, et al. Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles. IEEE Trans Neural Netw Learn Syst, 2021, 32: 5435–5444
26 Wang J, An J, Chen M S, et al. From model to implementation: a network algorithm programming language. Sci China Inf Sci, 2020, 63: 172102
27 You H L, Hu Y Y, Pan Z W, et al. Density-based user clustering in downlink NOMA systems. Sci China Inf Sci, 2022, 65: 152303
28 Fahim A. K and starting means for k-means algorithm. J Comput Sci, 2021, 55: 101445
29 Li H Z, Wang J. CAPKM++2.0: an upgraded version of the collaborative annealing power k-means++ clustering algorithm. Knowledge-Based Syst, 2023, 262: 110241
30 Dijkstra E W. A note on two problems in connexion with graphs. Numer Math, 1959, 1: 269–271
31 Dong L, Yuan X, Sun C Y. Event-triggered receding horizon control via actor-critic design. Sci China Inf Sci, 2020, 63: 150210
32 Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the Advances in Neural Information Processing Systems, 1999. 1057–1063
33 Pflueger M, Agha A, Sukhatme G S. Rover-IRL: inverse reinforcement learning with soft value iteration networks for planetary rover path planning. IEEE Robot Autom Lett, 2019, 4: 1387–1394
34 Islam F, Nasir J, Malik U, et al. RRT*-Smart: rapid convergence implementation of RRT* towards optimal solution. In: Proceedings of the International Conference on Mechatronics and Automation, Chengdu, 2012. 1651–1656
35 Tang Z, Xu X, Wang F, et al. Coordinated control for path following of two-wheel independently actuated autonomous ground vehicle. IET Intelligent Transp Syst, 2019, 13: 628–635
36 Dankwa S, Zheng W F. Twin-delayed DDPG: a deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In: Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, 2019. 1–5
37 Qiu C R, Hu Y, Chen Y, et al. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. IEEE Internet Things J, 2019, 6: 8577–8588