Expert Systems With Applications 252 (2024) 124281

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An adaptive multi-RRT approach for robot motion planning

Bohan Feng, Xinting Jiang, Boyan Li, Qi Zhou, Youyi Bi [*]

University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, 800 Dongchuan Road, Minhang Dist., Shanghai 200240, China

A R T I C L E I N F O

Keywords:
Robot motion planning
Multi-Rapidly-exploring Random Trees
Information gain
Upper-confidence bound
Adaptive planning

A B S T R A C T

Efficient motion planning is essential for robots to operate properly and safely. Traditional sampling-based planning algorithms have been extensively investigated, but they still often struggle in highly constrained environments. To address the issue, this paper introduces an Adaptive Multi-Rapidly-exploring Random Trees (AMRRT*) approach. The core idea is to adaptively improve the subtree generation, selection, and planning within the multi-tree sampling structure so that the configuration space can be explored more efficiently. During subtree generation, an information gain-based method is proposed to select appropriate locations for generating subtrees, minimizing sampling in areas that have been relatively fully explored. For subtree selection, the upper-confidence bound (δ) method is leveraged to balance global exploration and local exploitation. Additionally, a dynamic subtree planning method is developed to update the sampling direction and step size to increase the possibility of forming feasible paths. The theoretical proofs of the probabilistic completeness and asymptotic optimality for the AMRRT* approach are provided. To validate its effectiveness, we conduct a series of experiments against the state-of-the-art motion planning algorithms within 2D to 6D robotic configuration space. The results demonstrate AMRRT*'s superior performance on key metrics including computational efficiency and success rate, suggesting its promising potential in addressing complex motion planning challenges.

1. Introduction

Robots are redefining various aspects of production processes ranging from smart assembly, material transportation, to patrolling and inspection (Evjemo et al., 2020; Jan et al., 2023). As the working environment becomes more complex, the potential for robots to encounter different obstacles, such as raw materials, other robots and equipment, or even human workers, also increases. Collisions between robots and these obstacles not only bring equipment damage and pose threats to human operators, they can also create a cascade effect that disrupts production pace, derails production schedule leading to higher manufacturing cost (Inkulu et al., 2022). Therefore, effective motion planning methods are crucial for robots to work safely and efficiently in complex environments.

Motion planning is fundamentally a computational task of determining paths that allow robots to move from a starting position to a target destination under physical constraints while avoiding collisions with obstacles. Robots usually operate in a distinct space compared to humans, known as the configuration space (C-space) (Lozano-Pérez & Wesley, 1979). The C-space represents the set of all potential configurations, characterized by a given degree of freedom for the robot. While additional joints provide robots increased flexibility, they also introduce greater computational complexity due to the exponential relationship between the complexity of the C-space and the robot's degrees of freedom, which is often referred to as the curse of dimensionality.

Sampling-based motion planning algorithms (SBPs) tackle this challenge by employing a stochastic sampling strategy. Instead of explicitly constructing the intractable high-dimensional C-space, SBPs progressively construct either a roadmap or a tree by connecting valid C-space samples (Elbanhawi & Simic, 2014; Liang & Zhao, 2023). Roadmap represents the graph of feasible paths in the C-space, whereas tree is the branching structure grown from an initial position. Theoretically, SBPs can guarantee probabilistic completeness (Karaman & Frazzoli, 2011), signifying that given an infinite number of sample points, a solution can be discovered with a certain probability, if it exists. Further theoretical advancements have demonstrated the asymptotic optimality of SBPs (Gammell & Strub, 2021), ensuring that these algorithms will find a solution that is close to the optimal one as the number of samples increases.

* Corresponding author.
E-mail addresses: bohan.feng@sjtu.edu.cn (B. Feng), evans_jiang@sjtu.edu.cn (X. Jiang), liboyan@sjtu.edu.cn (B. Li), zhouqi1998@sjtu.edu.cn (Q. Zhou), youyi.bi@sjtu.edu.cn (Y. Bi).

https://doi.org/10.1016/j.eswa.2024.124281
Received 23 August 2023; Received in revised form 16 May 2024; Accepted 17 May 2024
Available online 23 May 2024

Although SBPs own the advantages mentioned above, their running time is significantly affected by the C-space complexity. Higher spatial complexity results in longer running time, making it challenging for SBPs to handle highly constrained environments, where generating valid samples and establishing effective connections in the search space is extremely difficult. Fig. 1 illustrates an example of the motion planning for a four-degree-of-freedom robot arm. Although the workspace seems to have ample free space, the C-space reveals a significant challenge in motion planning. This is due to the presence of numerous narrow passages and inaccessible areas within the C-space, amplifying the difficulties in motion planning. The probability of successfully extending a connection into a narrow passage is low. This often requires to run the sampling process multiple times in the nearby regions to establish the entire path along the narrow passage.

Traditional SBP methods, exemplified by the Rapidly-exploring Random Tree (RRT) and its variants (LaValle, 1998), utilize incremental sampling of the tree structure to establish connections between sample points. However, these methods may discard valid sample points if the nearest node in the single tree fails to find a collision-free path to these sample points in the C-space. This limitation stems from the tree expansion being constrained within a local area defined by the boundary tree nodes. Furthermore, planning attempts, whether successful or unsuccessful, often yield valuable insights that can guide motion planning to concentrate in the regions of the C-space that demands greater attention as well as inform refinements in the planning strategy itself. However, these insights are frequently overlooked by traditional methods. In highly constrained environments characterized by a multitude of narrow passages or complex geometries, traditional methods often struggle to find feasible planning solutions due to these limitations.

To address this issue, we propose a robot motion planning approach that leverages a multi-tree structure to explore the C-space by maintaining both high visibility as well as local connectivity information of the collision-free regions. Compared to traditional methods, our approach can utilize information from previous planning attempts (e.g., those already planned sample points) to improve subsequent planning strategies. It can effectively address motion planning problems in highly constrained environments. The main contributions of this paper include:

– A multi-tree structure for robot motion planning that combines the advantages of rapid exploration and global exploration is developed. This structure enhances the efficiency of the exploration process and effectively mitigates the challenges often encountered in single-tree planning.
– An information gain-based subtree generation method that can identify those inadequately explored regions is designed. This method prevents excessive attention on areas that have already been relatively fully explored.
– An upper-confidence bound (δ)-based subtree selection method that balances exploration versus exploitation is proposed. This method allocates computational resources effectively based on the complexity of C-space.
– A dynamic subtree planning method that flexibly updates the sampling direction and step size is developed. This method can increase the likelihood of generating successful paths by utilizing both successful and failed planning results.

The effectiveness of the proposed approach is examined in both computer simulations and physical experiments. Our approach shows strong motion planning capability and adaptability in complex environments. The rest of the paper is structured as follows. Section 2 presents a literature review of previous methods addressing the sampling problem in highly constrained spaces. Section 3 provides the problem formulation, and Section 4 introduces the proposed approach and explains the key techniques involved. Section 5 showcases the effectiveness of the proposed approach through 2D, 4D, and 6D simulation experiments, as well as physical experiments (7D) in industrial pick-and-place scenarios. Section 6 summarizes this work and highlights potential directions for future research.

2. Related work

SBPs can be broadly categorized into single-query and multi-query planners. Single-query planner, such as Rapidly-exploring Random Tree (RRT), generates a tree by incrementally adding nodes, starting from the robot's initial configuration, and growing towards randomly sampled configuration points in the C-space. In contrast, multi-query planner, such as the Probabilistic Road Map (PRM) (Kavraki et al., 1996), constructs a roadmap by precomputing and storing a graph that captures the connectivity of the robot's free C-space. Nodes in this graph represent randomly sampled configurations, and edges indicate valid paths between them. However, in highly constrained spaces, generating valid samples for either single or multi-query planner can be challenging, and the possibility of forming valid connections is low. Consequently, these algorithms may struggle to find paths with limited time. To address this issue, researchers have developed various techniques, such as bridge tests, bias toward regions with narrow structures, heuristic measures of obstacle boundaries, multi-tree structures, and learning-based methods.

The bridge test (Wilmarth et al., 1999) focuses on collision detection in C-space between two endpoints and their connecting line's midpoint. If the endpoints collide but the midpoint does not, the midpoint may be a valid sample for the developing roadmap. Yet, this approach faces

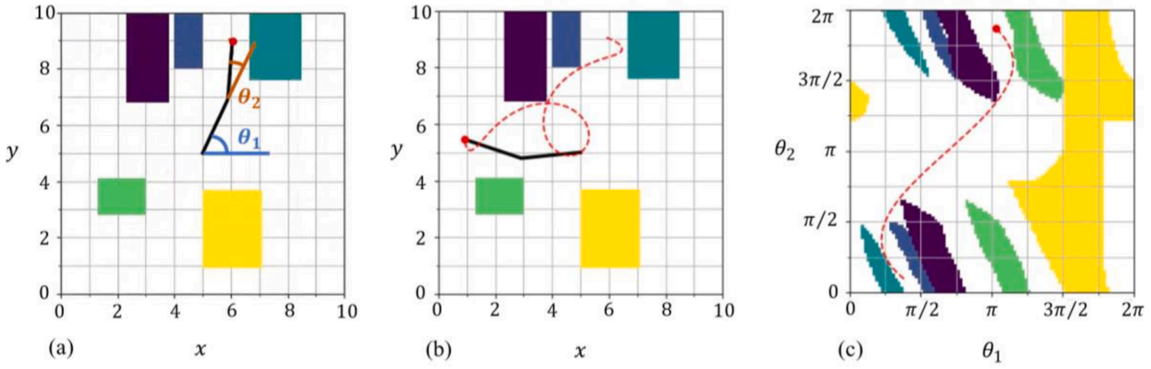

Fig. 1. Motion planning for a four-degree-of-freedom robot arm (two rotational joints θ_1 and θ_2, and two positional locations x and y). (a) and (b) present the workspace, while (c) illustrates the C-space containing the robot arm's motion path.

scalability issues with complex C-space and tends to overlook previously acquired knowledge.

Bias towards narrow structures, based on dynamic adjustments to sample distribution relative to local geometric complexities, has led to several methods. Dynamic domain RRT (Yershova et al., 2006) adapts its sampling to obstacles for better exploration. Principal component analysis (Dalibard & Laumond, 2009) and virtual force fields (Chen et al., 2021) respectively utilize statistical models and emulate repulsive forces for sampling guidance. However, over-focusing on narrow regions can neglect other viable paths, reducing the overall efficiency of path planning.

Heuristic measures of obstacle boundaries, such as the heuristic-based certificate set (Ma et al., 2021), focus on generating samples near these boundaries. This set tracks collision status and minimum distances between tree nodes and nearest obstacles, reusing this data to enhance sampling precision. However, representing real-world obstacles as complex geometries in C-space challenges heuristic navigation.

As for the motion planning methods leveraging multi-tree structures, their significant advancement lies in creating numerous trees to concurrently explore distinct regions of the C-space. Notable methodologies like RRT-Connect* (Klemm et al., 2015) and RRdT* (Lai et al., 2019) exemplify this trend. RRT-Connect* efficiently connects trees grown from start and goal configurations, accelerating the search process. RRdT* introduces a disjoint tree structure for global exploration and local-connectivity exploitation. Alongside these, other notable methods such as Sampling-Based Tree Roadmaps (SRT) (Plaku et al., 2005), C-FOREST (Otte & Correll, 2013), and MR-RRT (Sun et al., 2022) also contribute to the diverse landscape of multi-tree motion planning. The autonomy of each tree within the multi-tree structure empowers it to uniquely explore different regions, enriching the overall exploration. Moreover, the interconnected nature of these trees allows the sharing of sampled points. This enables the algorithms focusing on regions in the C-space showing higher promise or relevance to current planning task. Nevertheless, introducing multi-tree structures also brings challenges. The intricate architecture inherently intensifies the algorithmic complexity, demanding heightened computational resources. As such, the motion planning field needs more efficient strategies for multi-tree generation, selection, and planning. These strategies should aim for a balanced allocation of computational resources, ensuring that the benefits gained from richer exploration will surpass the negative impact of increased computational complexity.

In recent years, learning-based motion planning methods have been developed to replace the entire planning or focus on enhancing specific components of classical motion planning algorithms by advanced machine learning techniques (Wang et al., 2021). Ichter et al. (2018) train a conditional variational autoencoder using prior successful planning results to sample and project to promising regions in the working space. Wang et al. (2018) propose a Learning-based Multi-RRT (LM-RRT*) method that extracts key locations and selects subtree expansion with a ε-greedy strategy. Wang et al. (2020) introduce Neural RRT*, which combines a pretrained convolutional neural network model with RRT* to guide the sampling process and improve path planning performance. Lai et al. (2020) utilize the Markov chain method for sampling exploration and update the chain-like sampling order with Bayesian techniques. MPNet (Qureshi et al., 2021) generates feasible near-optimal paths directly using an encoding network and a planning network. However, challenges remain for these learning-based methods when dealing with highly constrained environments. Errors due to computational approximations or training-test mismatches can lead these methods to fail, highlighting the importance of enhancing the adaptability and robustness of motion planning methods.

In this paper, we combine the strengths of RRT-based planning and the multi-tree structure framework. Critical processes such as subtree generation, selection and planning are improved and guided by both environmental information and robot's planning experience. Our approach allows better adaptation to highly constrained environments with enhanced algorithmic efficiency and generalizability.

3. Problem formulation

Let $\mathscr{C} = (0,1)^d$ be the configuration space, where $d \in \mathbb{N}, d \geq 2$. Let $\mathscr{C}_{obs}$ be the obstacle region, such that $\mathscr{C} \setminus \mathscr{C}_{obs}$ is an open set, and denote the obstacle-free space as $\mathscr{C}_{free} = \mathrm{cl}(\mathscr{C} \setminus \mathscr{C}_{obs})$, where $\mathrm{cl}(\cdot)$ denotes the closure of a set. The start point x_{start} is an element of $\mathscr{C}_{free}$, and the goal region $\mathscr{X}_{goal}$ is an open subset of $\mathscr{C}_{free}$. A path planning problem is defined by a triplet $(x_{start}, \mathscr{X}_{goal}, \mathscr{C}_{free})$. The definitions of key concepts in robot motion planning, such as path, feasible path planning, optimal path planning, probabilistic completeness and asymptotic optimality are provided as follows.

Definition 1 (Path):. A function $\sigma : [0,1] \rightarrow \mathbb{R}^d$ of bounded variation is called a path if it is continuous. The cost of a path is the path's total variation, i.e., the Euclidean distance traversed by the path in $\mathbb{R}^d$.

The feasibility problem of path planning is to find a feasible path, if one exists, and report failure otherwise:

Definition 2 (Feasible Path Planning):. Given a path planning problem $(x_{start}, \mathscr{X}_{goal}, \mathscr{C}_{free})$, find a feasible path $\sigma : [0,1] \rightarrow \mathscr{C}_{free}$ such that $\sigma(0) = x_{start}$ and $\sigma(1) \in \mathrm{cl}(\mathscr{X}_{goal})$, if one exists. If no such path exists, report failure.

The optimality problem of path planning asks for finding a feasible path with minimum cost:

Definition 3 (Optimal Path Planning):. Given a path planning problem $(x_{start}, \mathscr{X}_{goal}, \mathscr{C}_{free})$ and a cost function $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$, find a feasible path σ^* such that $c(\sigma^*) = \min\{c(\sigma)\}$. If no such path exists, report failure.

Definition 4 (Probabilistic Completeness):. Given any feasible path planning problem $(x_{start}, \mathscr{X}_{goal}, \mathscr{C}_{free})$, a sampling-based motion planning algorithm is said to be probabilistically complete if the probability it returns a feasible path goes to one as the number of samples goes to infinity,

$$\liminf_{n\rightarrow\infty} P(\Sigma_n \neq \varnothing) = 1, \tag{1}$$

where n is the number of samples and $\Sigma_n \subset \Sigma_{feasible}$ is the set of feasible paths found by the planner from those samples.

Definition 5 (Asymptotic Optimality):. Given any feasible path planning problem $(x_{start}, \mathscr{X}_{goal}, \mathscr{C}_{free})$, a sampling-based motion planning algorithm is said to be asymptotically optimal if it converges asymptotically to an optimal solution as the number of samples goes to infinity,

$$P\left(\limsup_{n\rightarrow\infty} \min_{\sigma\in\Sigma_n}\{c(\sigma)\} = c^*\right) = 1, \tag{2}$$

where n is the number of samples, $\Sigma_n \subset \Sigma_{feasible}$ is the set of feasible paths found by the planner from those samples, $c : \Sigma \rightarrow \mathbb{R}_{\geq 0}$ is the cost of a path, and c^* is the optimal solution to the planning problem.

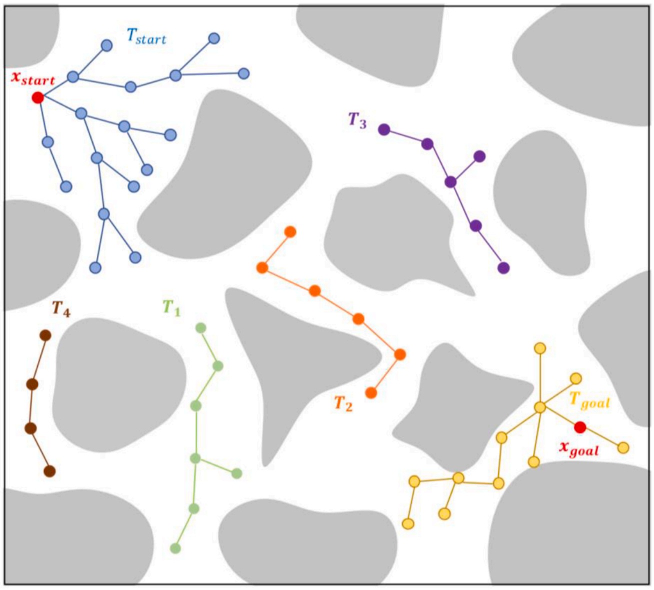

Fig. 2. Multi-tree structure of AMRRT. The two red dots x_{start} and x_{goal} are the start and goal points respectively.

4. Methods

4.1. The overall structure of the proposed approach

The proposed approach, referred to as the Adaptive Multi-Rapidly-exploring Random Trees (AMRRT*) algorithm, employs a multi-tree structure to effectively utilize information from sampled points for exploring and exploiting the C-space. In our approach, the main trees represented as $T_{main} = \{T_{start}, T_{goal}\}$ (visualized as the blue and yellow trees in Fig. 2) employ the RRT* structure. However, for constructing the remaining subtrees (illustrated by the green, orange, brown and purple trees in Fig. 2), we adopt the Markov Chain Monte Carlo property inherent to the RRdT* structure (Lai et al., 2019). This structure offers enhanced benefits, particularly in terms of optimizing local connectivity. The pseudo-code of AMRRT* is presented in Algorithm 1, which can be split into four parts: (1) Initialization (Line 1–6), setting the growing trees T_{start} and T_{goal}, and other M subtree samplers; (2) Subtree generation (Line 8), generating the subtree at the suitable location; (3) Subtree selection (Line 9), selecting the appropriate subtree to plan; (4) Subtree planning (Line 11), if T_i is subtree, we utilize dynamic subtree planning to exploit local structure. In the following subsections, the details of subtree generation, selection and planning are provided.

```
Algorithm 1: Adaptive Multi-RRT*
    Input: $x_{start} \in \mathcal{C}_{free}, x_{goal} \in \mathcal{C}_{free}, M, N, n_{ms}, \epsilon, \rho, \eta$
    Output: $\mathcal{T}$
 1  $V_{start} \leftarrow \{x_{start}\}, V_{goal} \leftarrow \{x_{goal}\}, Edge \leftarrow \emptyset$
 2  $T_{start} \leftarrow (V_{start}, Edge), T_{goal} \leftarrow (V_{goal}, Edge)$
 3  Initialize subtrees $T_1, \cdots, T_M$
 4  $\mathcal{T} \leftarrow \{T_{start}, T_{goal}, T_1, \cdots, T_M\}$
 5  $E(\mathcal{T}), R(\mathcal{T}) \leftarrow$ Initialize the energy and rewards of subtrees in $\mathcal{T}$
 6  $n \leftarrow 1, r \leftarrow 1$
 7  while $n \leq N$ do
 8      $\mathcal{T} \leftarrow$ SubtreeGeneration($\mathcal{T}, M, n_{ms}, \epsilon$)
 9      $T_i \leftarrow$ PickTree($R(\mathcal{T}), M, r$)
10      if $T_i \notin \{T_{start}, T_{goal}\}$ then
11          $x_{new} \leftarrow$ DynamicPlanning($T_i, \rho, n$)
12      else
13          $x_{rand} \leftarrow$ SampleFree
14          $x_{nearest} \leftarrow$ Nearest($T_i, x_{rand}$)
15          $x_{new} \leftarrow$ Steer($x_{nearest}, x_{rand}$)
16      if CollisionFree then
17          Add $x_{new}$ to all $T_i \in \mathcal{T}$ within $\epsilon$-distance
18          if $x_{new} \in T_{start}$ then
19              Rewire($T_{start}$)
20          $n \leftarrow n + 1$
21      else
22          $E(\mathcal{T}) \leftarrow$ EnergyAdjustment($T_i, \eta$)
```

4.2. Information gain-based subtree generation

Subtree generation is a crucial step in AMRRT*, where a new subtree is created from a sampled point in the C-space to explore new regions. Traditional methods for subtree generation, such as random generation or preprocessing the C-space to identify key regions, face several limitations. Random generation often repeatedly explores already known areas of the C-space, which results in inefficient exploration and unnecessary computational overhead. Preprocessing the C-space to guide the subtree generation, while potentially improving the exploration efficiency, introduces significant computational costs. Additionally, this method may lead to misjudgments of the path's feasibility in complex scenarios.

To address these challenges, we turn to clustering algorithms as they allow efficient partitioning of the C-space into regions of interest, facilitating targeted exploration with reduced computational demand. Various clustering algorithms are evaluated, including K-means, hierarchical clustering, and density-based spatial clustering (DBSC), and each comes with specific strengths and weaknesses. K-means, for instance, requires prior specification of the number of clusters and often fails with non-spherical data. Hierarchical clustering scales poorly with large datasets, and DBSC's performance heavily depends on the choice of distance threshold and density parameters, which vary greatly across different regions of the C-space.

Considering the limitations of these methods, we propose an information gain-based subtree generation method using the mean shift algorithm (Comaniciu & Meer, 2002). We choose mean shift for its robustness in handling the non-linear data distributions typically seen in robotic C-space. It does not require predefined cluster shapes, making it versatile for varying data distributions encountered in our application. The information in a configuration space can be seen as a measure of the exploration level in a particular region, and higher information indicates less exploration in a region. Our method aims to improve the efficiency of exploration in the multi-tree structure by avoiding unnecessary sampling in those relatively fully explored regions. The pseudo-code of information gain-based subtree generation is shown in Algorithm 2.

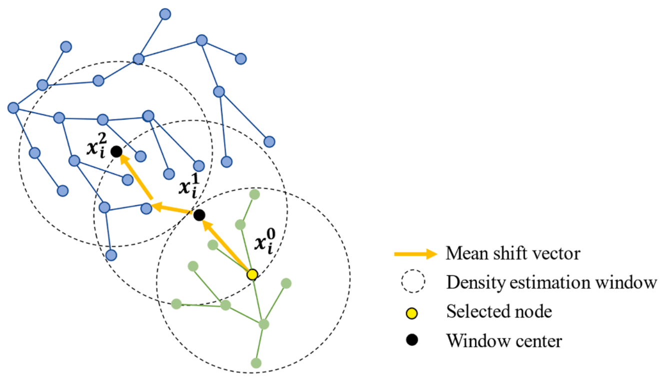

Fig. 3. An illustration of the mean shift procedure. Starting at node x_i, the mean shift procedure is run to find the stationary points of the density function. The superscripts $j = 0, 1, 2$ of x_i^j denote the mean shift iterations.

Algorithm 2: Information Gain-based Subtree Generation

```
1  Function SubtreeGeneration($\mathcal{T}, M, n_{ms}, \epsilon$):
2      if $|\mathcal{T}_{sub}| < M$ then
3          Select $n_{ms}$ nodes from $\mathcal{T}$
4          $P(s) \leftarrow$ MeanShift($n_{ms}$,SampleFree)
5          $x_{new} \leftarrow$ Propose new subtree location according to $P(s)$
6          if Dis($x_{new}, T_i \in \mathcal{T}$) $< \epsilon$ then
7              $T_i \leftarrow$ Add($x_{new}$)
8          else
9              $T_{new} \leftarrow (V = \{x_{new}\}, Edge)$
10             $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{new}\}$
11 return $\mathcal{T}$
```

In the proposed approach, the mean shift algorithm treats the data points in the feature space as an empirical probability density function and identifies dense regions as local maxima or modes of the distribution. For each data point, a gradient ascent procedure is performed on the local estimated density until convergence is reached, resulting in stationary points that represent the modes. Data points associated with the same stationary point are considered to belong to the same cluster, allowing us to identify areas of the C-space that have been relatively fully explored. Subsequently, we compare the distances between the random subtree generation sets and the cluster centroids. Based on these distances, we normalize them to form probabilities, and use these probabilities to select new subtree generation points. The implementation details are provided below.

Given n_{ms} nodes $x_i \in \mathbb{R}^d$ from tree $\mathcal{T}$, the multivariate kernel density estimate using a radially symmetric kernel $K(x)$ (i.e., Gaussian kernels) is given by

$$\hat{f}_K(x) = \frac{1}{n_{ms}h^d}\sum_{i=1}^{n_{ms}} K\left(\frac{x - x_i}{h}\right), \tag{3}$$

where the bandwidth parameter h defines the radius of kernel function. The radially symmetric kernel is defined as

$$K(x) = c_{k,d}\, k(\|x\|^2), \tag{4}$$

where $c_{k,d}$ represents a normalization constant which assures the integral of $K(x)$ from negative infinity to positive infinity is 1 and $k(\cdot)$ denotes the selected kernel function. Taking the gradient of the density estimate in Eq. (3), we can get:

$$\nabla\widehat{f}_K(x) = \frac{2c_{k,d}}{n_{ms}h^{d+2}} \underbrace{\left[\sum_{i=1}^{n_{ms}} - k^{'}\left(\left\|\frac{x-x_i}{h}\right\|^2\right)\right]}_{\text{first term}} \underbrace{\left[\frac{\sum_{i=1}^{n_{ms}} x_i k^{'}\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_{ms}} k^{'}\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x\right]}_{\text{second term}}, \tag{5}$$

where $k^{'}(\cdot)$ denotes the derivative of the selected kernel function. The first term in Eq. (5) reflects the density estimate at node x. Meanwhile, the second term denoted by m signifies the direction of the highest density increment and is proportional to the density gradient estimate at the same node.

In the mean shift procedure for a given node x_i (as illustrated in Fig. 3), we first compute the mean shift vector $m\left(x_i^j\right)$ where j of x_i^j denotes the mean shift iterations. Then we modify the density estimation window using the formula $x_i^{j+1} = x_i^j + m\left(x_i^j\right)$, and continue this process until the convergence criterion $\nabla\widehat{f}_K(x_i) = 0$ is satisfied. It's crucial to note that the selected nodes converging to an identical stationary point are categorized within the same cluster class. In the end we get the cluster centroids $C = \{C_1, \cdots, C_c\}$ where C_c represents the c-th cluster centroid from the selected nodes of the tree $\mathscr{T}$. The subtree generation sets are from SampleFree $\omega\mapsto\{\text{Sample}_n(\omega)\}_{n\to\infty}\subset\mathscr{C}_{free}$ which returns sequences of independent and identical distributed (i.i.d.) uniform samples in $\mathscr{C}_{free}$. The sets are denoted as $S = \{s_1, \cdots, s_j, \cdots, s_M\}, 1 \le j \le M$ where s_j represents the j-th subtree generation point. The distance between s_j and C can be calculated using the shortest Euclidean distance, denoted as $d(s_j, C)$. The distance measures its separation from the relatively fully explored C-space areas, with greater distance indicating higher potential information gain. Taking the normalization of all the distances to form probabilities, we can get

$$p(s_j) = \frac{\exp(d(s_j, C))}{\sum_{i=1}^{M}\exp(d(s_i, C))}. \tag{6}$$

Finally, we select new subtree generation point $s \in S$ based on their probability distribution set $P(s) = \{p(s_1), p(s_2), \cdots, p(s_M)\}$. To mitigate the computational cost associated with clustering, we can control the magnitude of n_{ms} while running the clustering process in parallel. In summary, our method can efficiently prioritize the searching of regions in the C-space that remain under-explored, ensuring that the computational resources won't be overly allocated on those relatively fully-explored regions.

4.3. Upper-confidence bound (δ)-based subtree selection

In the multi-tree structure, subtree selection is challenged by the need to balance global exploration and local utilization, especially given resource constraints. In our study, this challenge is approached under the framework of Multi-armed Bandit (MAB) problem, a decision-making problem aiming to select the best among a set of actions by sequentially observing rewards. Each arm, i.e., subtree, acts as a dynamic density estimator for connections within $\mathscr{C}_{free}$. In each round r, we choose a subtree to plan and receive a randomized reward $R_{T_i,r}$ based on the planning result. The reward $R_{T_i,r}$ is intrinsically linked to the C-space, with its values fluctuating based on distinct exploration outcomes within this space. The cumulate sequence of rewards $R(\mathscr{T}) = \{R_{T,1}, \cdots, R_{T,r}\}$ is a stochastic sequence with an unknown distribution of rewards that can change rapidly depending on the complexity of the C-space. The objective of MAB is to maximize the average return in r rounds, i.e., efficient allocation of resources based on the complexity of C-space.

To solve the MAB problem for subtree selection, we employ the Upper-confidence Bound δ (UCB (δ)) (Abbasi-yadkori et al., 2011) algorithm, characterizing by a confidence set of random values constructed from past selection of subtrees. A pseudo-code representation of UCB (δ) is provided in Algorithm 3.

```
Algorithm 3: Upper-Confidience Bound (δ)-based Subtree Selection
1 Function PickTree($R(\mathcal{T}), M, r$):
2     $p \sim$ UniformDistribution(0,1)
3     if $0 \le p < \frac{1}{M+2}$ then
4         $T_i \leftarrow T_{start}$ or $T_{goal}$
5     else
6         Calculate $c_{T_i,r}$ and $\bar{R}_{T_i,r}$ from $R(\mathcal{T})$
7         $T_i \leftarrow \text{argmax}_i\left(\bar{R}_{T_i,r} + c_{T_i,r}\right)$
8         $r \leftarrow r + 1$
9 return $T_i$
```

Suppose μ_{T_i} is the expected reward of choosing the subtree $T_i, i = 1, 2, \cdots, M$. Let $\mu_* = max\mu_{T_i}$ be the expected reward of the best tree, and $\Delta_{T_i} = \mu_* - \mu_{T_i}$ be the regret with respect to the best subtree. We assume that if the tree T_i is chosen in round r, we obtain a reward $\mu_{T_i,r} + \eta_r$. Here η_r represents the random noise associated with the reward of choosing a particular subtree in round r. $N_{T_i,r}$ denotes the number of times that we have chosen T_i up to round r, and $\overline{R}_{T_i,r}$ indicates the average of the rewards received by the planning result of T_i up to round r. We construct confidence intervals for the expected rewards $\mu_{T_i,r}$ based on $\overline{R}_{T_i,r}$ in the following lemma. For $\forall i \in \{1, 2, \cdots, M\}, \forall r \ge 0$:

$$\left|\overline{R}_{T_i,r} - \mu_{T_i,r}\right| \le c_{T_i,r}. \tag{7}$$

The length of confidence interval $c_{T_i,r}$ is independent of r, and it can be formulated as:

$$c_{T_i,r} = \sqrt{\frac{(1+N_{T_i,r})}{N_{T_i,r}^2}\left(1 + 2\log\left(\frac{M(1+N_{T_i,r})^{\frac{1}{2}}}{\delta}\right)\right)}. \tag{8}$$

The $c_{T_i,r}$ can help achieve high-probability constant regret Δ. Hence, at round r, we choose the tree:

$$T_i = \underset{i}{\mathrm{argmax}} \overline{R}_{T_i,r} + c_{T_i,r}. \quad (9)$$

Subtrees situated in constrained regions face sampling challenges, primarily due to their limited feasible sampling space. In contrast, areas with high visibility can be rapidly explored when accessed. Consequently, a greater allocation of resources is essential for subtrees in these constrained regions to effectively capture connectivity. To make $R_{T_i,r}$ responsive to the connectivity of the C-space, we design the rewards according to different planning results by drawing upon previous research (Wang et al., 2018).

(1) If the subtree planner can directly connect the feasible sample points with the given step size, assign a reward $R_{T_i,r} = 0.1$. This suggests the tree is in an open region with few obstacles.

(2) If a sample point is feasible with the step size but cannot be directly connected collision-free by the subtree planner, the reward is set to $R_{T_i,r} = 0.2$.

(3) If conditions (1) and (2) are not satisfied, the subtree planner is considered in the cluttered region, and the reward is set to $R_{T_i,r} = 0.3$.

Meanwhile, by integrating an energy lifecycle-based mechanism (Lai et al., 2019), subtrees are prevented from getting stuck in dead ends and consequently wasting resources. When the energy of a subtree is exhausted due to the failure count, we regenerate the subtree in a new region, actively exploring new regions for path planning. The pseudo-code for energy adjustment is detailed in Algorithm 4. Briefly, our method allocates computational resources based on the intricacies of the C-space to achieve a balance between global exploration and local utilization.

4.4. Dynamic subtree planning

The process of subtree planning involves determining the sampling direction (θ) and expansion step size ($\in$) to extend the current configuration ($x_{current}$) to a new configuration (x_{new}), i.e., $x_{new} \leftarrow x_{current} + \in \cdot \theta$. However, previous studies rarely consider to determine the sampling distribution and expansion step size adaptively during subtree planning in an integrated way.

To address this gap, we propose a dynamic subtree planning method that considers the sampling distribution as a Markov process with unobservable states, and models the proposed distribution using tree sampling information from prior successful and failed samples. This method effectively leverages information from failed planning points that is often overlooked in traditional methods, providing valuable insight into subsequent planning processes. Furthermore, we utilize past planning information to support the precise determination of the step size of the local subtree planning. Notably, this process takes the complexity of the surrounding environment into consideration.

Before delineating the specific methods, we need to first elucidate some foundational notions. Within our method, the subtree planner is conceptualized through the state $\mathscr{S}_{T_i,t}$, wherein $T_i \in \mathscr{T}$ signifies the spatial location and planning information set in the local planning step t. The state is represented as the tuple $\mathscr{S}_{T_i,t} = \left(x_t, \theta_t, \in_t, \theta_{t-1}, \mathscr{D}_t^f, \mathscr{B}(\rho)\right)$. Herein, x_t denotes the spatial location of subtree sampler, θ_t typifies the sampling direction, $\in_t$ is the step size, θ_t represents the successful planning direction unit vector value at step $t-1$, while $\mathscr{D}_t^f$ is the unsuccessful value set at step t, and $\mathscr{B}(\rho)$ embodies the hypersphere with radius R. Algorithm 5 presents the pseudo-code for dynamic subtree planning method. The details of determining the sampling direction and step size are explained as follows.

Algorithm 4: Energy Adjustment

```
1 Function EnergyAdjustment($T_i, \eta$):
2     if $T_i \notin \{T_{start}, T_{goal}\}$ then
3         $E(T_i) \leftarrow E(T_i) - 1$
4         if $E(T_i) < \eta$ then
5             $\mathcal{T} \leftarrow \mathcal{T} \backslash T_i$
6 return $E(\mathcal{T})$
```

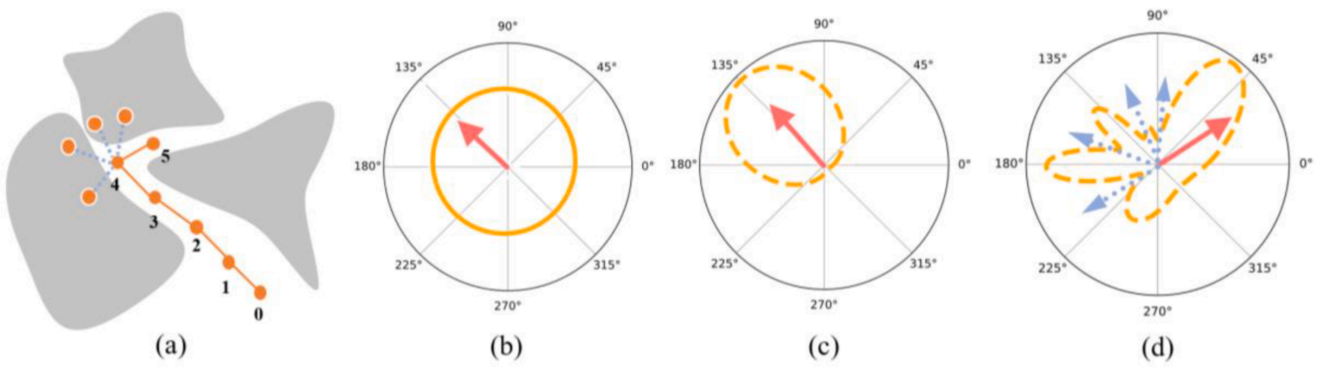

Fig. 4. An example of updating sampling distribution. (a) shows how the subtree plans from $t = 0$ to $t = 5$, with solid orange lines indicating successful plannings and dashed blue lines indicating failed plannings. (b) depicts the uniform sampling at $t = 0$, with red arrow indicating the sampling direction drawn from it. (c) presents the von Mises-Fisher distribution at $t = 1$, with red arrow showing the sampling direction drawn from it. (d) illustrates the Bayesian update distribution at $t = 4$, with red arrow denoting the successful sampling direction and blue arrows representing failed attempts.

Algorithm 5: Dynamic Subtree Planning

```
1  Function DynamicPlanning($T_i, \rho, n$):
2      $x_t, \theta_{t-1}, \mathcal{D}_t^f \leftarrow S_{T_i,t}$
3      $\overline{|\mathcal{D}^f|} \leftarrow$ Average failed attempts in $\mathcal{B}_i(\rho)$
4      $\epsilon_i \leftarrow$ StepSizeAdjustment($\overline{|\mathcal{D}^f|}, \rho, n$)
5      if $t > 0$ then
6          if $j > 0$ then
7              $\theta_{t,j} \sim \mathcal{D}_{t,j}(\theta) \leftarrow$BayesianUpdateDistribution($\theta_{t-1}, \mathcal{D}_t^f$)
8          else
9              $\theta_{t,j=0} \sim \mathcal{D}_{t,j=0}(\theta) \leftarrow$ VonMisesFisherDistribution($\theta_{t-1}$)
10     else
11         $\theta_{t=0,j=0} \sim$ UniformSampling
12     $x_{new} \leftarrow x_t + \epsilon_i \cdot \theta_{t,j}$
13     $R(\mathcal{T}) \leftarrow$ Receive $R_{T_i,t}$ according to planning result
14 return $x_{new}$
```

4.4.1. *Sampling direction*

For sampling direction θ, we propose a method that uses Bayesian update rules (Lai et al., 2020) to model the proposed distribution, and optimize the parameter selection of the kernel function. This method aims to incorporate information from both successful and failed planning attempts to update the sampling distribution. Fig. 4 shows an example of updating sampling distribution.

To effectively inherit insights from successful sampling direction while incorporating a stochastic component, we opt for the von Mises-Fisher distribution, denoted as $p(\theta;\kappa,\mu)$, to serve as the initial prior distribution $\theta_{t,j=0} \sim \mathscr{D}_{t,j=0}(\theta) = p(\theta;\kappa,\mu)$ (Banerjee et al., 2005). Here, j represents the number of attempts at local planning step t for subtree planner i. The von Mises-Fisher distribution is a probability distribution on the unit sphere, specifically designed to model directional data as shown in Eqs. (10) and (11):

$$p(\theta;\kappa,\mu) = \frac{1}{2\pi I_\nu(\kappa)}\exp(\kappa[\theta-\mu]), \tag{10}$$

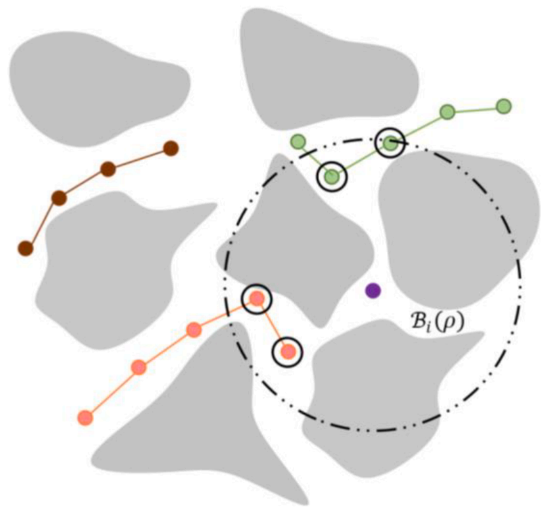

Fig. 5. An illustration of the step size selection. The process centers on the new subtree, represented by the purple point.

$$I_\nu(\kappa) = \frac{1}{\pi}\int_0^\pi \exp(\kappa cos\theta)\cos(\nu\theta)d\theta, \tag{11}$$

where θ is the random angular variable, μ is the average direction, i.e., the previously sampled successful direction, and $-\pi \le \theta,\mu < \pi$, $\nu = 0$. κ is a measure of the concentration of the probability density function, located in the semi-infinite interval $[0, \infty]$. A larger κ means the selection of θ is more concentrated in the mean direction. $I_{\nu=0}(\kappa)$ is the first class of modified Bessel functions. Based on the Bayesian update principle, the update of the sampling distribution can be expressed as follows:

$$\mathscr{D}_{t,j}(\theta) = \mathscr{D}_{posterior}(\theta) \propto \mathscr{D}_{likelihood} \bullet \mathscr{D}_{prior}. \tag{12}$$

Then the periodic kernel function $k(\cdot)$ that encapsulates the idea of having a decreasing nature to resample in previously sampled regions is used to construct $\mathscr{D}_{likelihood} \propto 1 - k(\cdot)$. The kernel function $k(\cdot)$ is formulated as follows:

$$k(\theta,\theta_{primer}) = \sigma^2\exp\left(-\frac{2}{l^2}\sin^2\left(\pi\frac{\theta-\theta_{primer}}{p}\right)\right), \tag{13}$$

where l denotes the length scale which directly influences the decay rate of the kernel function and σ^2 serves as the scaling factor, corresponding to the period of the spherical distribution, and the repetition period p is set to 2π.

Compared to previous work, our method leverages an exponential function to get a positive correlation between l and the size of the unsuccessful value set $\mathscr{D}^f$. This correlation allows dynamic modification of l during the energy lifecycle of the subtree. The length scale l is calculated as:

$$l = \alpha + \beta\left(1 - \exp^{-|\mathscr{D}^f|}\right), \tag{14}$$

where α is a constant term, β is a coefficient term and $|\cdot|$ represents the size of the set $\mathscr{D}^f$. Notably, as $\mathscr{D}^f$ increases, the change of l leads to the decrease of the decay rate of the exponential component $k(\theta,\theta_{primer})$. Consequently, it contributes an enhanced correlation between θ and θ_{primer}, suggesting their interrelation remains significant despite increasing distances. It leads to a more exploratory selection of sampling directions after successive failures, which assists with finding feasible direction in highly constrained environments.

We can rewrite the update of the sampling distribution when $j \geq 1$:

$$\begin{aligned} \theta_{t,j} \sim \mathscr{D}_{t,j}(\theta) &= \mathscr{D}_{t,j}\left(\theta | \theta_{t-1}, \mathscr{D}_t^f\right) \\ &= \mathscr{D}_{t,j-1}\left(\theta | \theta_{t-1}, \mathscr{D}_t^f\right)\left(1 - k(\theta, \theta_{t,j-1})\right). \end{aligned} \tag{15}$$

It is important to note that, $\mathscr{D}_{t,0}\left(\theta | \theta_{t-1}, \mathscr{D}_t^f\right)$ reduces to $\mathscr{D}_{t,0}\left(\theta | \theta_{t-1}, \mathscr{D}_t^f := \emptyset\right) = p(\theta; \kappa, \mu)$. Finally, we get the sampling direction $\theta_{t,j} \sim \mathscr{D}_{t,j}(\theta)$. In summary, our method can adaptively update the sampling distribution based on historical planning results.

4.4.2. Step size

In our approach, we consider the selection of step size in subtree planning comprehensively at the regional scale to incorporate the information from the surrounding environment. To adjust the step size, we consider the previous planning results of the subtree planner. If the results indicate a smaller search space near the current location, the step size is reduced to better explore the region, i.e., the path planner becomes more "discreet" in this situation.

As shown in Fig. 5, $\mathscr{B}_i(\rho)$ denotes the hypersphere with a radius of ρ, which contains the state information of all subtree planners at multiple time steps. We use this regional scope to extract the average failed attempts ($\overline{|\mathscr{D}^f|} \in \mathscr{B}_i(\rho)$) from all local subtree planners and optimize the step size selection as follows:

$$\epsilon_i = g\left(\overline{|\mathscr{D}^f|}\right)\left(\frac{log(n)}{n}\right)^{\frac{1}{d}}, \tag{16}$$

where ϵ_i is the step size, n is the number of sampling nodes, d is the dimensionality of the planning problem and $g\left(\overline{|\mathscr{D}^f|}\right)$ is a function that maps the complexity of the state information within the hypersphere, reflecting the complexity of the surrounding environment. The function $g\left(\overline{|\mathscr{D}^f|}\right)$ is defined as:

$$g\left(\overline{|\mathscr{D}^f|}\right) = \gamma \exp\left(-\frac{\overline{|\mathscr{D}^f|}}{E}\right), \tag{17}$$

where γ is a positive constant and E is the initial energy of the subtree. Our approach can optimize step size selection by integrating previous planning information across the predefined region.

4.5. Analysis of the probabilistic completeness and asymptotic optimality for AMRRT*

In this section, we provide proofs of the probabilistic completeness and asymptotic optimality for AMRRT*.

Theorem 1 (Probabilistic Connection of Subtrees:. (Wang et al., 2018)**):** For a set of sampled points $\{x_1(x_{start}), x_2, \cdots, x_k(x_{goal})\}$ serving as root nodes and the subsequent extension of k trees $\{T_1, T_2, \cdots, T_k\}$, the total number of sample nodes n extended by all trees satisfies the inequality:

$$n \geq k(\alpha\beta\epsilon)^{-1} ln[4(1-\epsilon)] ln\left(\frac{3ln\left[2k^2(1-\epsilon)\right]}{\gamma\beta}\right) + k\xi^{-1} ln\left(\frac{3k^2}{2\gamma}\right), \tag{18}$$

where $\epsilon, \alpha, \beta, \gamma$ are constants in $(0, 1)$ (Hsu et al., 1997) and ξ is defined as the proportion of $\mathscr{C}_{free}$ to $\mathscr{C}$. If the inequality is satisfied, then the probability that every pair among these k trees can be successfully connected is at least $1 - \gamma$.

Theorem 2 (Infinite Sampling Uniformity):. AMRRT*can cover $\mathscr{C}_{free}$ in uniform sampling as the number of random samples goes to infinity.

Proof. Given Lines 6 to 7 of Algorithm 2, if a subtree $T_i \in \mathscr{T}$ exists within a distance ϵ, the proposed sample point will be incorporated into

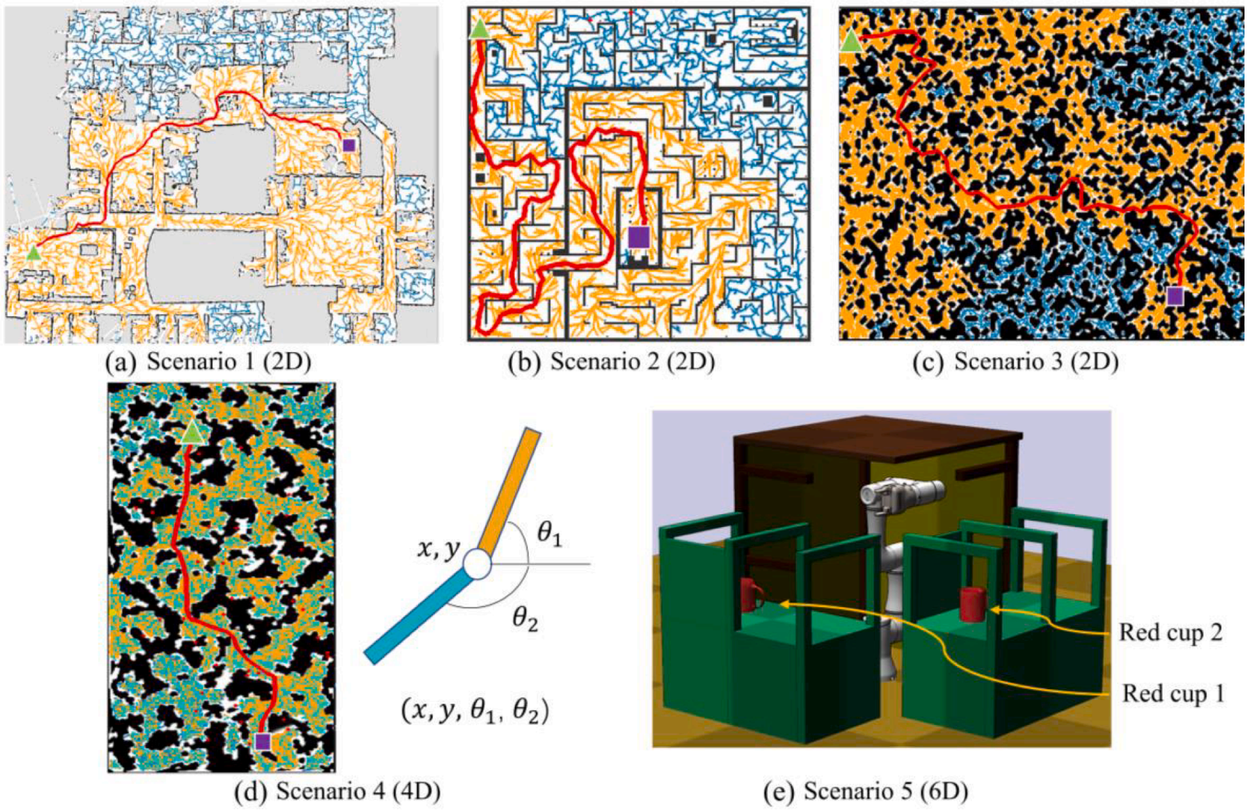

Fig. 6. Three 2D (a, b, c), one 4D (d) and one 6D (e) experimental scenarios.

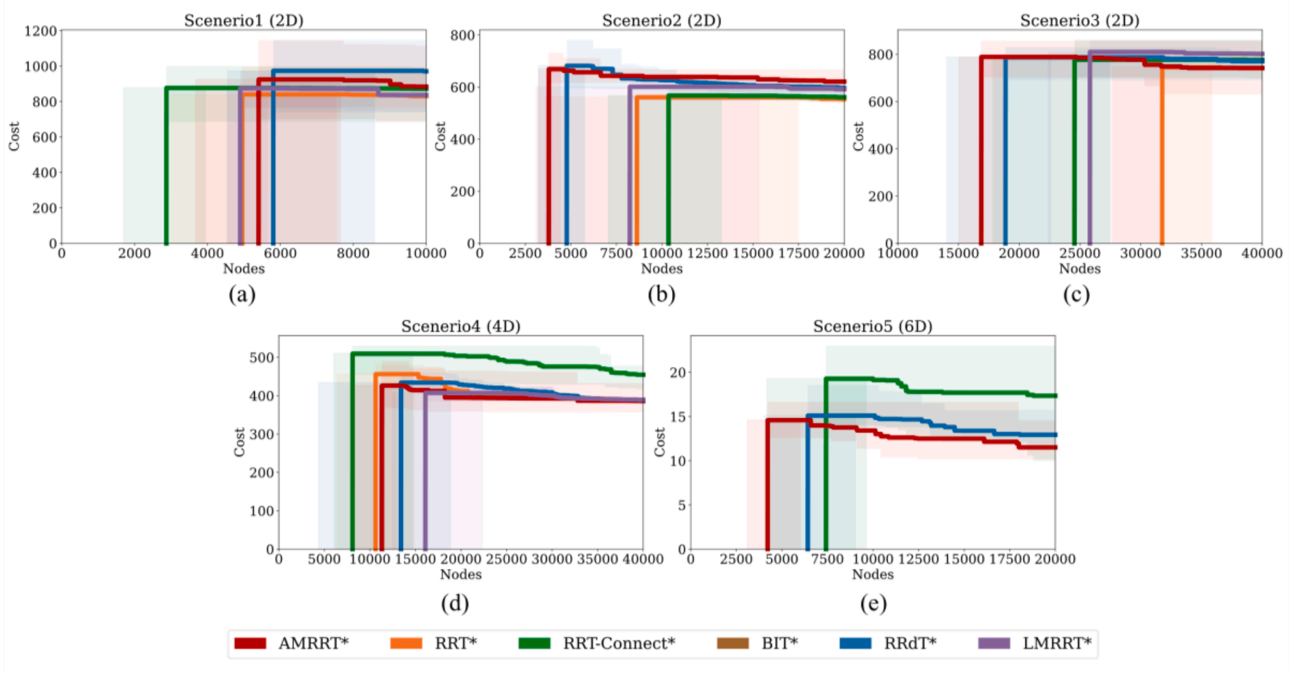

Fig. 7. Cost values against the number of sampling nodes for different algorithms. The solid line represents the mean value, while the shaded area indicates the range between the maximum and minimum values.

Table 1
Average running time in seconds for different algorithms to obtain initial solutions. Empty cells indicate cases when the algorithm fails to obtain a solution.

Algorithm	Scenario1	Scenario2	Scenario3	Scenario4	Scenario5
AMRRT*	82.6	**65.4**	**637.6**	**754.2**	**381.3**
RRT*	234.6	953.7	8130.4	1996.2	–
RRT-Connect*	101.9	494.8	5040.2	984.1	1334.2
BIT*	72.4	339.5	–	2235.6	1772.5
RRdT*	95.3	86.1	815.6	901.0	572.6
LMRRT*	**68.2**	278.9	4000.2	2100.7	–

the existing subtree T_i and no new subtree will be created. Denote the closed sphere with radius $\epsilon > 0$ centered at a node x in $\mathscr{T}$ as $\mathscr{B}_x(\epsilon)$. Every new subtree T_{new} will not be located inside the sphere $\mathscr{B}_x(\epsilon)$ from $T_i \in \mathscr{T}$ where $T_i \neq T_{new}$. Referring to the dense set theory (Munkres, 2020), as the number of nodes increases, the union of the volume of $\mathscr{B}_x(\epsilon)$ will cover $\mathscr{C}_{free}$. Consequently, the total number of subtrees is restricted by the sequence of connected ϵ-sphere until their union volume completely fills $\mathscr{C}_{free}$. As all subtrees in $\mathscr{T}$ incorporated into $T_{main} = \{T_{start}, T_{goal}\}$, AMRRT* sampling will be chiefly influenced by Lines 13 to 15 of Algorithm 1, transitioning the algorithm to predominantly use SampleFree within $\mathscr{C}_{free}$. This ensures that as the sample count escalates, AMRRT* can realize infinite sampling uniformity.

Theorem 3. (Probabilistic Completeness):. AMRRT* attains probabilistic completeness for motion planning as the number of samples goes to infinity.

Proof. As stated in Theorem 2, there will be no new subtrees being created as the number of samples goes to infinity. Therefore, the behavior of AMRRT* sampling will be dominated by Lines 13 to 15 of Algorithm 1 when all subtrees in $\mathscr{T}$ are joined to T_{start}, meanwhile the AMRRT* algorithm changes to typical RRT structure. Therefore, AMRRT* will be reduced to use SampleFree. According to the probabilistic completeness of RRT (LaValle, 1998), there exists a constant $a > 0$ that :

$$\liminf_{n\to\infty} P\left(\left(\Sigma_n^{AMRRT} \cap \mathscr{X}_{goal}\right) \neq \varnothing\right) > \liminf_{n\to\infty} \left(1 - e^{-an}\right) = 1, \tag{19}$$

where n is the number of samples, and $\Sigma_n^{AMRRT} \subset \Sigma_{feasible}$ is the set of feasible paths found by AMRRT* planner from those samples. The probabilistic completeness proof for the AMRRT* algorithm is complete.

Theorem 4 (Asymptotic Optimality):. AMRRT* inherits asymptotic optimality as the number of samples goes to infinity.

Proof. From Theorem 1, all trees in the $\mathscr{C}_{free}$ will be connected to a single tree T_{start}. According to Theorem 2, there will be infinite sampling available to improve that tree. Together with adequate rewiring procedure at Line 19 of Algorithm 1, if step size is larger than $2\left(1+\frac{1}{d}\right)^{\frac{1}{d}}\left(\frac{\mu(\mathscr{C}_{free})}{\zeta_d}\right)^{\frac{1}{d}}$, our approach inherits the asymptotic optimality (Karaman & Frazzoli, 2011):

$$P\left(\limsup_{n\to\infty} \min_{\sigma\in\Sigma_n^{AMRRT}} \{c(\sigma)\} = c^*\right) = 1, \tag{20}$$

where n is the number of samples, d is the dimension of the space $\mathscr{C}$, $\mu(\mathscr{C}_{free})$ denotes the Lebesgue measure of the obstacle-free space, ζ_d is the volume of the unit sphere in the d-dimensional Euclidean space. The asymptotic optimality proof for the AMRRT* algorithm is complete.

5. Experiments

5.1. Experiment scenarios and settings

We examine the effectiveness of the proposed approach in three kinds of simulated motion planning problems, including 2D scenarios with a mass-point robot, 4D scenario with a robot's rotating arm, and 6D scenario with a multi-joint robot arm, respectively. Each simulation is designed to mirror common layouts found in real-world settings. Specifically, the 2D and 4D scenarios can reflect mobile robots moving in search or rescue missions, while the 6D scenario simulates a typical use

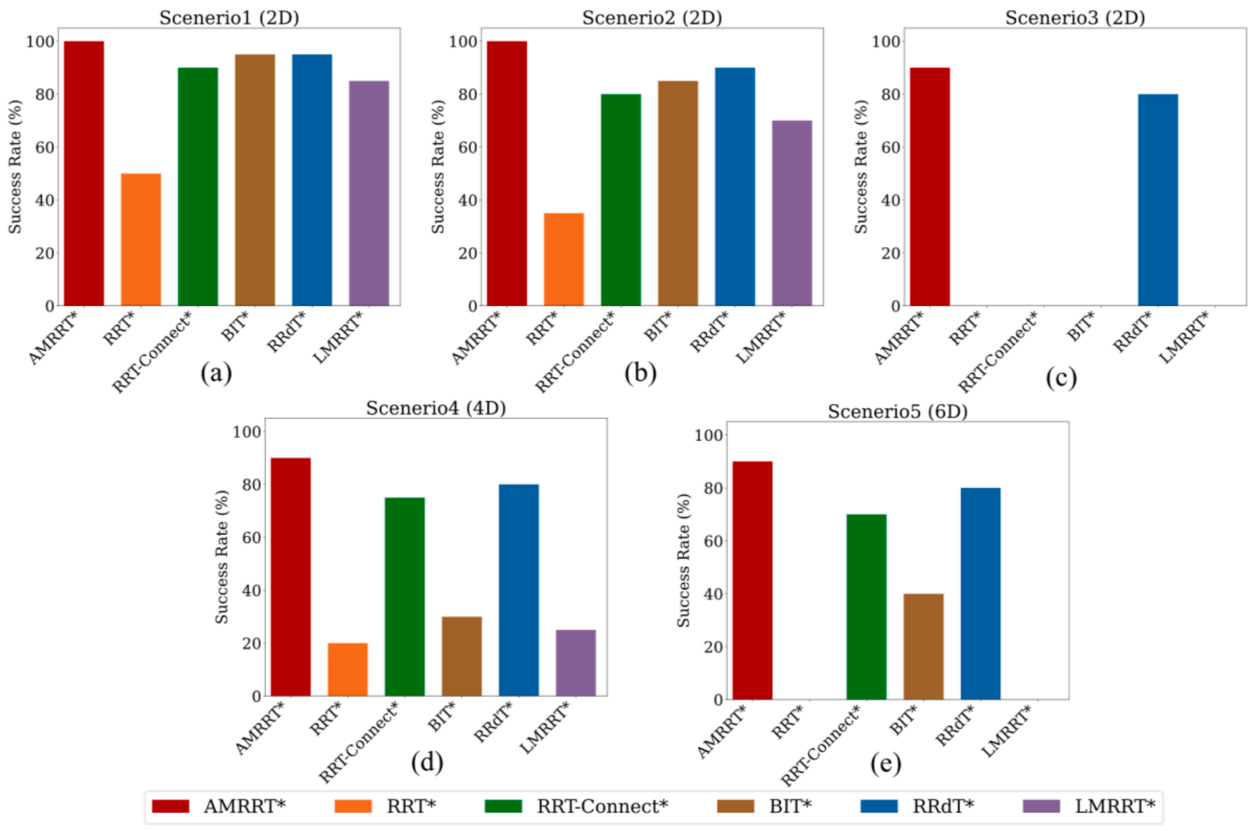

Fig. 8. Average success rate of different algorithms under a specified planning time limit. (a) is limited to 150 s, (b) is limited to 500 s. (c), (d) and (e) are limited to 1000 s.

of robot in material handling tasks. Additionally, we test our approach in a real object pick-and-place scenario typically encountered in manufacturing environments.

Fig. 6 illustrates the 2D, 4D and 6D experimental scenarios. In the 2D scenarios, the robot is considered as a mass point moving on a two-dimensional plane. In the 4D scenario, a robot with its rotating arm has four degrees of freedom (i.e., $(x, y, \theta_1, \theta_2)$ as shown in Fig. 6(d)). We construct 2D and 4D scenarios using image-based mapping. In these images, white areas indicate free spaces, while other colors represent obstacles. The starting points (x_{start}) are depicted as green triangles, while the goal points (x_{goal}) are represented by purple squares. The 6D scenario presents the task of moving a robot arm between two tables constructed on the Klapmt platform (Hauser, 2013). In this scenario, the six degrees of freedom of the robot include the rotational degrees of six

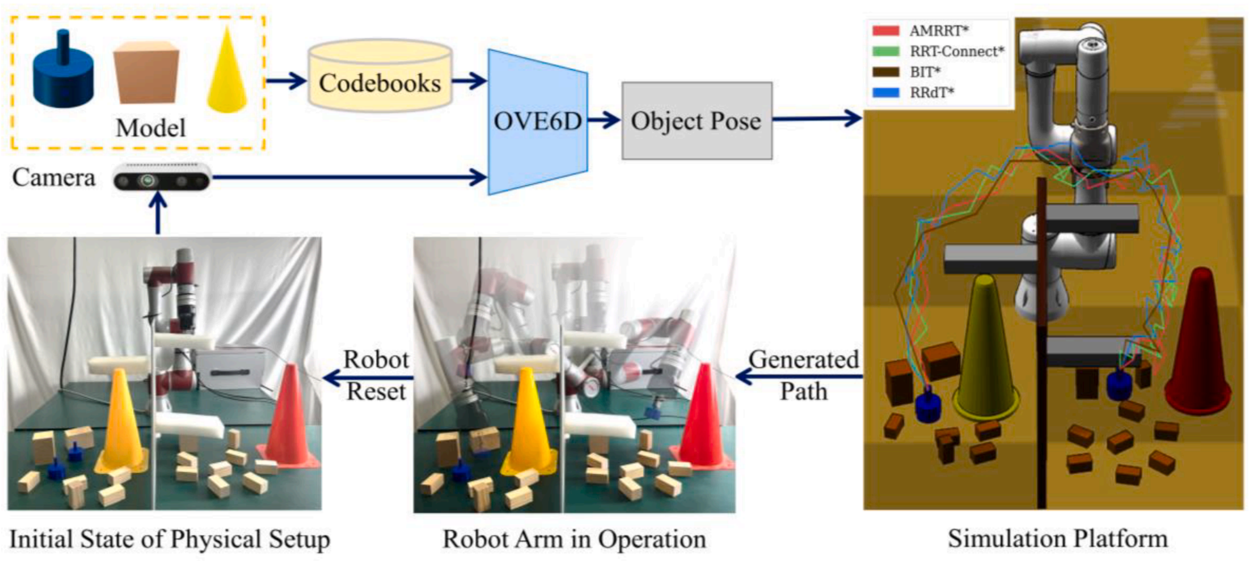

Fig. 9. The realistic object pick-and-place experiment setup and workflow. Exemplary motion paths generated with different algorithms are visualized and differentiated by color on the simulation platform.

joints. To augment the complexity of the task, the robot's workspace is deliberately populated with wooden planks and horizontal beams. The wooden planks are placed to obstruct movement in three dimensions. Additionally, horizontal beams are strategically positioned to obstruct certain feasible paths to the tables, particularly when the robot arm attempts to retrieve the red cups. These settings are intentionally integrated to escalate the planning difficulty, requiring advanced computational approach to navigate successfully. In the 6D scenario, red cup 1 denotes the start position and red cup 2 signifies the goal position. In all five scenarios, we need to find feasible paths for these robots to move from initial positions to goal positions while avoiding collisions with obstacles.

We compare our approach (AMRRT*) with other notable motion planning algorithms such as RRT* (Karaman & Frazzoli, 2011), RRT-Connect* (Klemm et al., 2015), LMRRT* (Wang et al., 2018), RRdT* (Lai, 2021), and Batch-Informed RRT* (i.e., BIT*) (Gammell et al., 2020). The BIT* algorithm is configured to use a batch size of 20 sampling points. We implement these algorithms in Python using the same planning framework and test them on a computer with Intel i7-10300 CPU and 32 GB RAM. Each algorithm is executed 20 times in one scenario to obtain reliable statistics.

5.2. *Experiment results*

In this section, the results of a series of experiments on our proposed AMRRT* approach and the compared algorithms are presented. Fig. 7 illustrates the relationship between cost values and the number of sampling nodes for different algorithms. The cost value represents the cumulative length of the path in the final solution. It's worth noting that the cost is set to 0 when the algorithm fails to find an initial solution. Due to its batch sampling nature, the BIT* algorithm sampled the least points in all scenarios for finding the initial solution and cannot be plotted in Fig. 7 like other algorithms. Table 1 summarizes the average running time of different algorithms when deriving initial solutions. Meanwhile, Fig. 8 compares the success rates of these algorithms in achieving initial solutions across various constraints and scenarios. If the histogram of an algorithm is not plotted in Fig. 8, it means that this algorithm is unable to generate paths successfully within the given time limit.

(1) 2D scenarios with mass point

In this simulated motion planning problem, we test three different types of 2D scenarios as shown in Fig. 6(a), (b) and (c). Scenario 1 simulates an indoor environment where the prime robot must navigate through a complex layout of multiple rooms and obstacles to reach the target goal. Scenario 2 illustrates a maze environment, characterized with numerous twists, turns, and dead ends. Scenario 3 is a noisy map with cluttered obstacles. Scenario 3 depicts a challenging and intricate environment, characterized by a noisy, cluttered terrain densely packed with diverse obstacles.

In Fig. 7(a), the proposed AMRRT* approach requires a higher number of samples to find a feasible solution compared to RRT* and RRT-Connect* algorithms in Scenario 1. However, the results presented in Table 1 support the notion that the multi-tree structure algorithms, including the LMRRT* and AMRRT* algorithms, require significantly less time for sampling valid points compared to RRT* and RRT-Connect* algorithms. Hence, despite the higher sampling point requirement, the multi-tree structure algorithm offers a distinct advantage in terms of planning time for simpler environments. Among the multi-tree structure algorithms, LMRRT* algorithm exhibits the lowest valid sampling requirement and average running time. This is attributed to the preprocessing of the environment map in the LMRRT* algorithm, which identifies critical path nodes that constitute the solution and uses them as nodes for subtree growth. Despite requiring fewer valid sampling points for successful planning, the success rate of LMRRT* under the given constraints is comparatively less (85 %) than that of the proposed AMRRT* approach (100 %), as shown in Fig. 8(a).

Furthermore, in more complex congested environments, as

Table 2
Average running time and success rate for different algorithms to obtain initial solutions. Empty cells indicate cases when the algorithm fails to obtain a solution in 1000 s.

Algorithm	Average Running Time (s)	Success Rate
AMRRT*	**312.5**	**95 %**
RRT*	–	–
RRT-Connect*	787.3	70 %
BIT*	942.1	30 %
RRdT*	499.8	85 %
LMRRT*	–	–

illustrated in Fig. 7(b) and (c), the AMRRT* approach demonstrates a significant advantage in the number of required valid sampling points to obtain the initial solution and the average running time. As Fig. 8(c) shows, this advantage is more pronounced in additionally complex 2D environments where RRT*, RRT-Connect* and BIT* algorithms cannot even obtain initial solutions. Additionally, Fig. 8(b), (c), and (d) further highlight the superior stability of our approach compared to other algorithms. Compared to other multi-tree algorithms, our approach does not need the preprocessing of the map environment as required by LMRRT*. Furthermore, it diverges from the entirely randomized tree generation and selection in the RRdT* algorithm. These unique attributes position our algorithm as a standout performer among multi-tree structure algorithms.

(2) 4D scenario with robot's rotating arm

As shown in Fig. 6(d), the environment of Scenario 4 consists of narrow passages where the robot's arm must rotate to meet the angle transformation requirement and the position of the robot should also satisfy the passing condition.

Fig. 7(d) illustrates that the performance of the proposed AMRRT* approach is between the RRT* and RRdT* in terms of the number of valid samples required to find a feasible solution. AMRRT* requires fewer valid samples than other multi-tree algorithms, indicating its capability dealing with narrow passages and its efficiency in allocating computational resources on areas of significance. Furthermore, Table 1 confirms that the AMRRT* approach exhibits a lower time cost of sampling valid points compared to other algorithms. Fig. 8(d) demonstrates that multi-tree structure-based algorithms exhibit higher success rates compared to RRT*, RRT-Connect*, and BIT* algorithms. The high success rate of our algorithm further validates the effectiveness of our subtree generation and selection strategies over other multi-tree algorithms. Overall, the AMRRT* approach demonstrates superior adaptability and efficiency, outshining other multi-tree structure algorithms, particularly in complex 4D environments.

(3) 6D scenario with robot arm's transport task

In Scenario 5 (see Fig. 6(e)), a multi-joint robot arm with a single-degree-of-freedom mechanical gripper needs to start from the position of red cup 1, navigate through the complex table obstacles and the horizontal bar obstacles in the middle area, and reach the position of red cup 2. The clamping angle of the mechanical gripper is fixed.

Fig. 7(e) shows that the proposed AMRRT* approach only requires 4500 sampling points to construct the initial solution. Also, according to Table 1, AMRRT* can find the sampling points that constitute the initial solution in the shortest time. Moreover, Fig. 8(e) demonstrates that AMRRT* gets the highest success rate among all compared algorithms. In contrast, the RRT* algorithm is less effective in the 6D scenario, likely due to its inefficient single-tree structure. The other multi-tree algorithms such as RRdT* shows a slightly lower success rate than AMRRT*. LMRRT* is unable to find feasible solution. This limitation stems from LMRRT*'s fundamental principle of extending subtrees from pre-identified key positions within the C-space. The dense distribution of obstacles in 6D scenario complicates the identification of these key positions, potentially leading to serious misidentifications. Furthermore, the absence of a subtree regeneration strategy in LMRRT* significantly impedes its ability to discover viable solutions in environments characterized by denser obstacles and higher complexity. Conclusively, the experimental evaluation offers evidence of the superior efficiency and adaptability of the AMRRT* approach, especially within the high-dimensional, intricately structured environments.

(4) Realistic object pick-and-place scenario

To validate the effectiveness of the AMRRT* approach within realistic operational settings, we design and execute a set of experiments anchored on an object pick-and-place application, as illustrated in Fig. 9. Our experimental setup comprises a Jaka Zu3 multi-joint robotic arm equipped with a mechanical gripper, which presents a 7D planning problem. To ensure accurate object detection and pose estimation, our setup includes a dual-camera system. This system features two Realsense cameras, each equipped with advanced depth sensing technology. These depth cameras, positioned strategically on opposite sides of the work table, can capture high-resolution three-dimensional spatial information. Both blue cylinders and rectangular wooden blocks are operation objects. In the depicted pick-and-place task, the robot arm transfers an object from one side of the baffle plate (i.e., the white centerline shown in the physical setup of Fig. 9 since only the thickness of the plate can be seen from the front view) to the other, navigating around three white cuboid obstacles on the baffle and two conical barrels. For precise pose estimation of objects, the OVE6D algorithm (Cai et al., 2022) is utilized. The images captured by Realsense cameras are processed by the OVE6D to obtain the pose data. These data are then fed into the Klampt simulation platform, where the motion paths for robot are generated.

In Fig. 9, an exemplary path generated by the AMRRT* algorithm is visualized and highlighted with red color. The paths generated by other methods are also illustrated in different colors. These visualizations reveal that the paths generated by AMRRT*, RRT-Connect*, and RRdT* are similar, demonstrating their ability for successful navigation in complex real environments. Particularly, the path generated by BIT* notably contains less unnecessary turning points compared to other methods. This performance could be attributed to the informed-connection nature of BIT*, but it comes at the cost of higher computational overhead and reduced robustness under complex planning conditions.

Table 2 presents the comparison of the average computation time and success rate for these algorithms in the realistic object pick-and-place task (each algorithm has been executed 20 times for this task), revealing the operational efficiency and reliability of each. The AMRRT* algorithm not only exhibits a high success rate of 95 % but also maintains moderate computational demand, showcasing its balance between algorithmic performance and computational efficiency. In contrast, other algorithms do not compare favorably with our method in terms of average computation time and success rate. The results of this physical experiment underscore the effectiveness of our proposed approach in real-world robotic applications.

6. Conclusion

In this paper, we propose the AMRRT* robot motion planning approach to efficiently generate collision-free paths in highly constrained environments. Our approach leverages the rapid exploration property of the RRT method and the global exploration property of the multi-tree structure. Our contributions lie in the development of three methods to adaptively improve the subtree generation, selection, and planning within the multi-tree structure. Firstly, we develop an information gain-based subtree generation method utilizing the mean shift algorithm, which can select subtree generation locations with higher information gain to explore the environment efficiently. Secondly, we model the subtree selection as a MAB problem, and utilize the UCB (δ) method to allocate resources by considering the balance between global exploration and local utilization. Thirdly, we design a dynamic subtree planning method that updates the sampling direction and step size based on local planning results, maximizing the likelihood of forming feasible paths. We also give theoretical proofs of the probabilistic completeness and asymptotic optimality of our approach. The experiment results demonstrate that our proposed approach outperforms other methods, particularly in success rate of path planning and computational time to find initial solutions.

While the AMRRT* approach demonstrates considerable efficacy in highly constrained scenarios, its performance in simpler settings warrants careful consideration. In environments with less intricate requirements, the multi-tree structure of AMRRT* may lead to less efficiency when compared to single or dual-tree algorithms. This outcome primarily stems from the computational burden inherent in the global exploration capabilities of multi-tree structures. Such environments may not fully utilize or require the comprehensive exploratory scope offered by AMRRT*, leading to potential inefficiencies.

Future work may involve further optimizing the algorithm's computational efficiency in different scenarios while preserving its high-quality solutions. Our proposed approach offers promising solutions for adaptive robot motion planning in highly constrained environments, and we hope our work will inspire the development of more advanced methods in this area.

CRediT authorship contribution statement

Bohan Feng: Conceptualization, Formal analysis, Methodology, Writing – original draft. **Xinting Jiang:** Validation, Visualization. **Boyan Li:** Validation. **Qi Zhou:** Investigation, Validation. **Youyi Bi:** Supervision, Project administration, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Youyi Bi reports financial support was provided by National Key R&D Program of China.

Data availability

Data will be made available on request.

Acknowledgements

The authors would like to acknowledge the financial support from National Key R&D Program of China (2022YFB4702400).

References

Abbasi-yadkori, Y., Pál, D., & Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 24). Curran Associates Inc.

Banerjee, A., Dhillon, I. S., Ghosh, J., & Sra, S. (2005). Clustering on the unit hypersphere using von Mises-Fisher distributions. *J. Mach. Learn. Res., 6*, 1345–1382.

Cai, D., Heikkia, J., & Rahtu, E. (2022). OVE6D: Object viewpoint encoding for depth-based 6D object pose estimation. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 6793–6803).

Chen, G., Luo, N., Liu, D., Zhao, Z., & Liang, C. (2021). Path planning for manipulators based on an improved probabilistic roadmap method. *Robotics and Computer-Integrated Manufacturing, 72*, Article 102196.

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(5), 603–619.

Dalibard, S., & Laumond, J.-P. (2009). *Control of probabilistic diffusion in motion planning* (pp. 467–481).

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access, 2*, 56–77.

Evjemo, L. D., Gjerstad, T., Grøtli, E. I., & Sziebig, G. (2020). Trends in smart manufacturing: Role of humans and industrial robots in smart factories. *Current Robotics Reports, 1*(2), 35–41.

Gammell, J. D., Barfoot, T. D., & Srinivasa, S. S. (2020). Batch Informed Trees (BIT*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research, 39*(5), 543–567.

Gammell, J. D., & Strub, M. P. (2021). Asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems, 4*(1), 295–318.

Hauser, K. K. (2013). *Robust contact generation for robot simulation with unstructured meshes*. International Symposium of Robotics Research.

Hsu, D., Latombe, J.-C., & Motwani, R. (1997). Path planning in expansive configuration spaces. *Proceedings of international conference on robotics and automation*, *3*, Pp. 2719–2726 vol.3.

Ichter, B., Harrison, J., & Pavone, M. (2018). Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 7087–7094).

Inkulu, A. K., Bahubalendruni, M. V. A. R., & Dara, A. (2022). Challenges and opportunities in human robot collaboration context of Industry 4.0 - A state of the art review. *Industrial Robot: The International Journal of Robotics Research and Application, 49*(2), 226–239.

Jan, Z., Ahamed, F., Mayer, W., Patel, N., Grossmann, G., Stumptner, M., & Kuusk, A. (2023). Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities. *Expert Systems with Applications, 216*, Article 119456.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*(7), 846–894.

Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation, 12*(4), 566–580.

Klemm, S., Oberländer, J., Hermann, A., Roennau, A., Schamm, T., Zollner, J. M., & Dillmann, R. (2015). RRT*-Connect: Faster, asymptotically optimal motion planning. In *2015 IEEE international conference on robotics and biomimetics (ROBIO)* (pp. 1670–1677).

Lai, T. (2021). sbp-env: A Python package for sampling-based motion planner and samplers. *Journal of Open Source Software, 6*(66), 3782.

Lai, T., Morere, P., Ramos, F., & Francis, G. (2020). Bayesian local sampling-based planning. *IEEE Robotics and Automation Letters, 5*(2), 1954–1961.

Lai, T., Ramos, F., & Francis, G. (2019). Balancing global exploration and local-connectivity exploitation with rapidly-exploring random disjointed-trees. *International Conference on Robotics and Automation (ICRA), 2019*, 5537–5543.

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning.*

Liang, Y., & Zhao, H. (2023). CCPF-RRT*: An improved path planning algorithm with consideration of congestion. *Expert Systems with Applications, 228*, Article 120403.

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communication ACM, 22*(10), 560–570.

Ma, H., Liu, J., Meng, F., Pan, J., Wang, J., & Meng, M.-Q.-H. (2021). A nonuniform sampling strategy for path planning using heuristic-based certificate set. In *2021 IEEE international conference on robotics and biomimetics (ROBIO)* (pp. 1359–1366).

Munkres, J. R. (2020). *Topology* (2nd ed.). Prentice Hall Inc.

Otte, M., & Correll, N. (2013). C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics, 29*(3), 798–806.

Plaku, E., Bekris, K. E., Chen, B. Y., Ladd, A. M., & Kavraki, L. E. (2005). Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics, 21*(4), 597–608.

Qureshi, A. H., Miao, Y., Simeonov, A., & Yip, M. C. (2021). Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics, 37*(1), 48–66.

Sun, Z., Wang, J., & Meng, M.-Q.-H. (2022). Multi-tree guided efficient robot motion planning. *Procedia Computer Science, 209*, 31–39.

Wang, J., Chi, W., Li, C., Wang, C., & Meng, M.-Q.-H. (2020). Neural RRT*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering, 17*(4), 1748–1758.

Wang, J., Zhang, T., Ma, N., Li, Z., Ma, H., Meng, F., & Meng, M.-Q.-H. (2021). A survey of learning-based robot motion planning. *IET Cyber-Systems and Robotics, 3*(4), 302–314.

Wang, W., Zuo, L., & Xu, X. (2018). A Learning-based multi-RRT approach for robot path planning in narrow passages. *Journal of Intelligent & Robotic Systems, 90*(1–2), 81–100.

Wilmarth, S. A., Amato, N. M., & Stiller, P. F. (1999). MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 1024–1031.

Yershova, A., Jaillet, L., Simeon, T., & Lavalle, S. M. (2006). Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. *IEEE International Conference on Robotics & Automation*.