Proceedings of the ASME 2023
International Mechanical Engineering Congress and Exposition
IMECE2023
October 29-November 2, 2023, New Orleans, Louisiana

IMECE2023-113127

A ROBOT MOTION PLANNING APPROACH BASED ON ADAPTIVE MULTI-TREE SAMPLING

Bohan Feng[1] **Xinting Jiang**[1] **Youyi Bi**[1*]

[1] University of Michigan – Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Efficient motion planning methods are essential to ensure industrial robots to work properly and safely. Although sampling-based planning algorithms are viable ones, they often struggle to adapt to highly constrained and complex environments. This paper introduces a new robot motion planning approach for such environments, utilizing a multi rapidly-exploring random trees exploration structure. The approach combines the fast exploration property of RRT-based methods with the global exploration property of multi-tree structures. In the subtree generation, an information gain-based method is used to analyze the sampled information from multiple trees to compute the potential information gain at various subtree generation locations. By selecting the locations with higher information gain, our method can effectively improve the exploration quality of the environment. Furthermore, an adaptive local subtree planning method is developed, which relies on local structure information and dynamically updates the sampling distribution to maximize the possibility of forming feasible trajectories in narrow passages. The effectiveness of the proposed approach is tested in 2D, 4D, and 6D environments, along with a complex material picking scenario. These experiments demonstrate that the proposed approach surpasses the performance of other algorithms, particularly in those highly constrained and complex environments. Our study contributes to the development of advanced and highly-adaptive motion planning methods for robots in complex environments.

Keywords: Motion planning, Multi rapidly-exploring random tree, Information gain, Adaptive local planning

1. INTRODUCTION

Robots have become increasingly prevalent in various aspects of manufacturing [1], including assembly, transportation, inspection, etc. As the manufacturing environments and jobs become more complex, robots will have higher risk to collide with obstacles of materials, human workers, and other robots. For example, when using robotic arm for assembly, the arm needs to maneuver around parts and components of different size while avoiding collisions with the table, wires and sensors. Collisions during the assembly process not only can result in loss of equipment, but may also cause interrupted production paces [2]. Consequently, effective motion planning methods are crucial to ensure robots to work safely and efficiently when performing tasks in complex environments.

Motion planning involves finding feasible motion trajectories for robots 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). 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) address this issue through a random sampling strategy. SBPs do not need to explicitly construct the intractable high-dimensional C-space. Instead, they iteratively build a connected roadmap by concatenating valid

[*] Corresponding author, Assistant Professor in Mechanical Engineering, Shanghai Jiao Tong University
Email: youyi.bi@sjtu.edu.cn

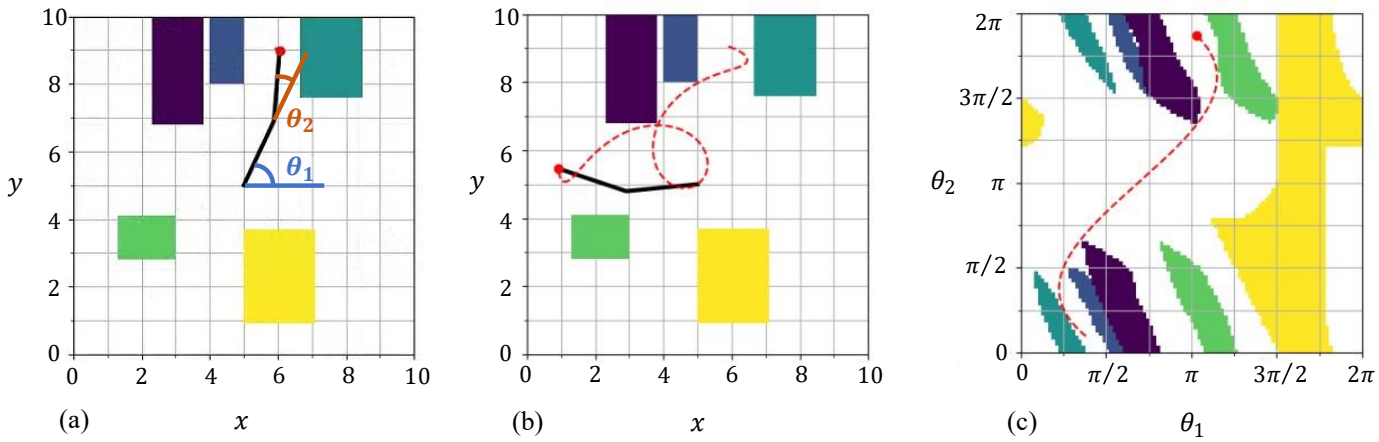

FIGURE 1: Motion planning for a four-degree-of-freedom robot arm (two rotational joints θ_1 and θ_2, and two positional locations x and y). (a) and (b) present the workspace, while (c) illustrates the configuration space containing the robot arm's motion trajectory. Although the workspace seems to have ample free space, the configuration space reveals a significant challenge in motion planning. This is due to the presence of numerous narrow passages and inaccessible areas within the configuration space, making motion planning considerably difficult.

samples of the C-space. Theoretically, SBPs can guarantee probabilistic completeness [3], signifying that given an infinite number of sampling points, a solution can be discovered with a certain probability, if it exists. Further theoretical advancements have demonstrated the asymptotic optimality of SBPs [4], ensuring that these algorithms will find a solution that is close to the optimal one as the number of samples increases.

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 and complex environments, where generating valid samples in the search space is extremely difficult or even impossible. For example, when finding path through a narrow passage, the likelihood of selecting a sample point inside the passage is relatively low due to the limited free space. Additionally, if the sampling tree fails to expand successfully, it may discard the failed sampling points that fall within the narrow passage. Consequently, narrow passages can hinder tree growth until a series of tree expansions successfully establish connections within the restrictive space. As illustrated in Figure 1, the probability of successfully extending a connection from the initial configuration into a narrow passage is very low. Furthermore, such sampling process would need to occur multiple times in the nearby regions to establish the entire path along the narrow passage.

Traditional SBP methods often use incremental sampling of single or double tree structures to construct a roadmap of connected nodes. However, these methods may discard valid samples when there is no free path from the nearest node to the sampled point in C-space. This limitation arises due to the tree expansion's being confined to the local area restricted by the boundary tree nodes. This situation becomes even worse in complex C-spaces, where feasible routes are scarce. To address this issue, we propose a robot motion planning approach that leverages a multi-tree structure to explore the C-space, maintaining high visibility and preserving local connectivity information without overlooking those already planned points. It can effectively address motion planning problems in highly constrained environments. The main contributions of this paper include:

- A multi-tree RRT structure that combines the advantages of rapid and global exploration is developed. This structure enhances the efficiency of the exploration process and effectively mitigates the challenges associated with complex environments often encountered in single-tree planning.

- An information gain-based subtree generation method that enables more efficient environment exploration is designed. This method prevents excessive attention on areas that have already been fully explored and improves the efficiency of the planning process.

- An adaptive local subtree planning method that dynamically updates the sampling direction and step size based on real-time information from local subtree planning is introduced. This method explores both successful and failed plannings and increases the likelihood of generating successful sample points and connections.

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 introduces the proposed approach and explains the key techniques involved. Section 4 showcases the effectiveness of the proposed approach through 2D, 4D, and 6D simulation experiments, as well as physical experiments in industrial pick-and-place scenarios. Section 5 provides the summary of this work and highlights potential directions for future research.

2. RELATED WORK

As a class of popular methods in robot motion planning, SBP algorithms have achieved relatively high efficiency and accuracy in motion planning by randomly sampling the environment and iteratively building a connected path from the robot's valid configuration space [5]. SBP algorithms can be categorized into single-query and multi-query approaches. Single-query planners, such as rapidly-exploring random tree (RRT) [6], generate a feasible path by connecting an initial point to a target point. In contrast, multi-query planners (e.g., probabilistic road map (PRM) [7]), construct a roadmap that facilitates efficient execution of multiple path query instances. However, in highly constrained spaces, generating valid samples for either single or multi-query approaches can be challenging, and the possibility of forming valid sampling connections is low. Consequently, these algorithms may struggle to find paths with limited time. To address this issue, researchers have developed various new strategies or methods, such as bridge tests, bias toward regions with narrow structures, heuristic measures of obstacle boundaries, multi-tree structures, and learning-based techniques to improve the sampling process.

The core idea of the bridge test [7] involves checking collisions among two endpoints and the midpoint of a line segment connecting these endpoints. If both endpoints collide while the midpoint remains collision-free, the midpoint is accepted as a new node in the roadmap under construction. This method is difficult to scale as the complexity of the C-space increases and often ignores prior knowledge since each run is a new start.

The motion planning method based on bias toward regions with narrow structures is try to adjust the distribution of sampling points and corresponding computational resources online according to the geometric complexity of local regions [8]. Various methods based on this idea have been developed, including dynamic domain RRT [9], principal component analysis [10], hybrid Gaussian models [11], and virtual force field [8]. However, when bias is overly strong, these methods may be stuck in local areas near a narrow structure, and fail to find feasible solutions elsewhere in the search space.

Motion planning algorithms with heuristic measures of obstacle boundary are designed to generate samples closer to the obstacle boundary by employing heuristic measures. For example, Ma et al. [12] propose a heuristic-based certificate set which maintains information with collision status and minimum distance to the nearest obstacle during the planning process and reuses that information in following iterations. However, obstacles are often not explicitly represented in C-space, making it challenging for heuristic measures to discover multiple narrow passages.

As for the motion planning methods based on multi-tree structures, their basic idea is to generate multiple trees to explore different regions simultaneously. Such methods include bidirectional RRT connection algorithms [9], sampling-based tree roadmaps (SRT) [13], C-FOREST [14] Triple-RRTs [15], RRdT [16] and MT-RRT [17], each offering unique advantages. Multi-tree structures can explore configuration spaces more thoroughly and diversely compared to single-tree structures, as each tree can focus on a different region or direction in the space. These structures can also use information from other trees to sample more points and expand trees in the promising regions of C-space. However, multi-tree structures can increase algorithm complexity with higher requirement of computational resource. Thus, more effective multi-tree generation and planning strategies with balanced resource allocation are needed.

In recent years, learning-based methods have been developed to guide sampling decisions in motion planning by learning from planning examples. Itcher et al. [18] train a conditional variational autoencoder using prior successful planning results to sample and project to promising regions in the working space. Wang et al. [19] propose a learning-based multi-RRT (LM-RRT) method that extracts key locations and selects subtree expansion with a reinforcement learning ε-greedy strategy. Tai et al. [20] utilize the Markov chain method for sampling exploration and update the chain-like sampling order with Bayesian techniques. Khan et al. [21] employ a graph neural network to encode the topology of C-space and calculate sampling distribution parameters. MPNet [22] generates feasible near-optimal paths directly using an encoding network and a planning network. However, challenges still remain for these learning-based methods when dealing with complex environments. Errors due to computational approximations or training-test mismatches can cause these methods to fail, highlighting the importance of enhancing the adaptiveness 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 and subtree planning are optimized and guided by a learning-based approach. Our approach allows better adaptation to highly constrained and complex environments with enhanced algorithmic efficiency and generalizability.

3. METHODS

3.1 The Overall Workflow 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 different regions of the C-space, achieving a balance between global exploration and local exploitation. The pseudo-code of AMRRT is presented in Algorithm 1, which can be split into four parts: (1) Initialization (Line 1–5), setting the growing trees T_{start} and T_{goal}, and other m multi-subtree samplers; (2) Subtree generation (Line 8), generating the subtree at the appropriate location; (3) Subtree selection (Line 9), selecting the corresponding tree to expand based on the energy value $E(\mathcal{T})$ of the trees; (4) Subtree planning (Line 1), if T_i is subtree, we utilize adaptive local subtree sampling to exploit local structure.

In the stage of subtree generation, we design an information gain-based subtree generation method using the mean shift algorithm (see Line 8 in Algorithm 1). This method calculates

the current clustering of sampled points by analyzing the multi-tree structure's point set, treating the centroids of clusters as fully explored regions. New subtree generation then selects locations with higher information gain for more efficient environment exploration.

In the stage of subtree planning, an adaptive subtree planning algorithm (see Line 11 in Algorithm 1) is developed. This algorithm updates the sampling direction and step size for constrained areas by considering past successful and failed expansions, and sequentially creates connected points to maximize the likelihood of forming feasible trajectories in narrow passages.

Algorithm 1: Adaptive Multi-RRT*

Input: $x_{start} \in C_{free}, x_{goal} \in C_{free}, N, n_{ms}, m, \epsilon, \gamma, dim, R$
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 $T_{sub1}, \cdots, T_{subm}$
4 $\mathcal{T} \leftarrow \{T_{start}, T_{goal}, T_{sub1}, \cdots, T_{subm}\}$;
5 $E(\mathcal{T}) \leftarrow$ Initialize the energy of $\mathcal{T}$
6 $n \leftarrow 1$;
7 **while** $n \leq N$ **do**
8 $\mathcal{T} \leftarrow$ `SubtreeGeneration`$(\mathcal{T}, n_{ms}, m, \epsilon)$
9 $T_i \leftarrow$ `PickTree`$(E(\mathcal{T}))$
10 **if** $T_i \notin \{T_{start}, T_{goal}\}$ **then**
11 $x_{new} \leftarrow$ `AdaptivePlanling`(T_i, γ, n, dim, R)
12 **else**
13 $x_{rand} \leftarrow$ `RandomSample`
14 $x_{nearest} \leftarrow$ `Nearest`(T_i, x_{rand})
15 $x_{new} \leftarrow$ `Steer`$(x_{nearest}, x_{rand})$
16 **if** *CollisionFree* **then**
17 Join x_{new} to T_i;
18 **if** x_{new} *joined to* T_{start} **then**
19 `Rewire`(T_{start});
20 $n \leftarrow n + 1$
21 $E(\mathcal{T}) \leftarrow$ `EnergyAdjustment`

It is worth noting that the multi-tree structure from RRdT [16] is incorporated in the proposed AMRRT approach, as depicted in Figure 2. Specifically, we utilize the RRT algorithm's expansion for T_{start} and T_{goal} (e.g., the blue and yellow trees in Fig. 2). For the remaining generated subtrees (e.g., the green, orange, and purple trees in Fig. 2), we employ the expansion of the sequential Markov Chain Monte Carlo (MCMC) to effectively leverage local connectivity, leading to improved exploitation of local regions within the C-space. In the following subsections, the details of subtree generation and adaptive local subtree planning are provided.

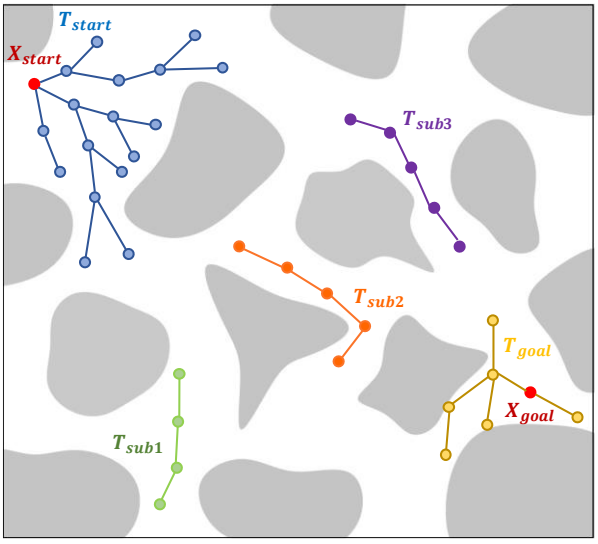

FIGURE 2: Multi-tree structure of AMRRT. The two red dots X_{start} and X_{goal} represent the start and goal points respectively. The blue and yellow trees, T_{start} and T_{goal}, depict the trees constructed from the start and goal points, while the other colored trees ($T_{sub1}, T_{sub2}, T_{sub3}$) represent the remaining generated subtrees. Gray areas are obstacles.

3.2 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 and find feasible paths for a moving robot. However, traditional methods for subtree generation have several limitations. Randomly generating subtrees can lead to inefficient exploration and redundant computation. On the other hand, pre-processing the search space to identify key regions and generating subtrees in those regions may be computationally expensive. The performance of this strategy highly relies on the pre-processing effect, and its exploration capability is limited. Considering these limitations, we propose an information gain-based subtree generation method using the mean shift algorithm.

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, referred as information gain, aims to improve the efficiency of exploration in the multi-tree structure by avoiding wasteful sampling in those already explored regions. To achieve this, we utilize the mean-shift clustering algorithm [23], which is a non-parametric method that does not make assumptions about the shape of the distribution or the number of clusters. It 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 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 pseudo-code of information gain-based subtree generation is showed in Algorithm 2.

Algorithm 2: Information Gain-based Subtree Generation

```
Function SubtreeGeneration(𝒯, m, ϵ):
    if |𝒯_sub| < m then
        Select n_ms sampled points from 𝒯
        s ← MeanShift(n_ms)
        x_newsub ← Propose new subtree location according to s
        if Dis(x_newsub, T_i ∈ 𝒯) < ϵ then
            Join x_newsub to T_i;
        if Not joined to existing tree then
            T_newsub ← (V = {x_newsub}, Edge);
            𝒯 ← 𝒯 ∪ {T_newsub}
return 𝒯
```

The implementation details of the mean shift algorithm are provided below. Given n_{ms} sampled points $x_i \in \mathbb{R}^d$ from the tree $\mathcal{T}$, the multivariate kernel density estimate using a radially symmetric kernel (i.e., Gaussian kernels), $K(x)$, is given by,

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

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

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

where $c_{k,d}$ represents a normalization constant which assures the integral of $K(\mathrm{x})$ from negative infinity to positive infinity is 1. Taking the gradient of the density estimator in Equation (1) and further algebraic manipulation yields,

$$\nabla\hat{f}(x) = \frac{2c_{k,d}}{nh^{d+2}}\underbrace{\left[\sum_{i=1}^{n} g\left(\| \frac{x - x_i}{h} \|^2\right)\right]}_{first\ term}\underbrace{\left[\frac{\sum_{i=1}^{n} x_i g(\| \frac{x - x_i}{h} \|^2)}{\sum_{i=1}^{n} g(\| \frac{x - x_i}{h} \|^2)} - x\right]}_{second\ term} \tag{3}$$

where $g(x) = -k'(x)$ denotes the derivative of the selected kernel function. The first term is proportional to the density estimate at x (computed with the kernel $G = c_{g,d}g(\| x \|^2)$). The second term, called the mean shift vector, m, points toward the direction of maximum increase in density and is proportional to the density gradient estimate at point x obtained with kernel K. The mean shift procedure for a given point x_i is illustrated in Fig. 3 and its main steps are as follows:

(1) Compute the mean shift vector $m(x_i^j)$.

(2) Translate density estimation window: $x_i^{j+1} = x_i^j + m(x_i^j)$.

(3) Iterate Steps (1) and (2) until convergence, i.e., $\nabla f(x_i) = 0$.

(4) Sampled points that converge to the same stationary point are considered as the same cluster class.

The subtree generation sets are denoted by $S = \{s_1, \dots, s_j, \dots, s_k\}$, $1 \le j \le k$ where s_j represents the j-th subtree generation point, and the cluster centroids are denoted by $C = \{C_1, C_2, \dots, C_h\}$, where C_h represents the h-th cluster centroid. The distance between s_j and C can be calculated using the shortest Euclidean distance, denoted as $d(s_j, C)$. After computing the distances, we can normalize them to form probabilities. The probability of selecting s_j from S can be calculated as:

$$p(s_j) = exp(d(s_j, C))/\sum_{i=1}^{k} exp\big(d(s_i, C)\big) \tag{4}$$

Finally, we select new subtree generation point $s \in S$ based on their probabilities $P(S) = \{p(s_1), p(s_2), \dots, p(s_k)\}$.

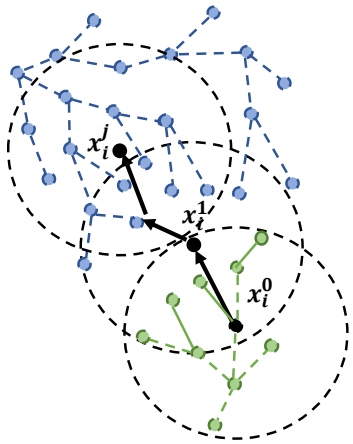

FIGURE 3: An illustration of the mean shift procedure. Starting at sampling point x_i, the mean shift procedure is run to find the stationary points of the density function. The superscripts $0,1,j$ of x_i denote the mean shift iterations. Black dots denote the input data points and successive window centers. The dotted circles denote the density estimation windows.

Additionally, we introduce the concept of energy for each tree, which represents its life cycle with an initial value. The energy of a tree is reduced whenever it fails in planning, which can prevent the tree from getting stuck in local traps such as dead ends or narrow passages. In each step of the planning process, we normalize the energy values of all trees, forming a discrete probability distribution (see Algorithm 3). The tree is then

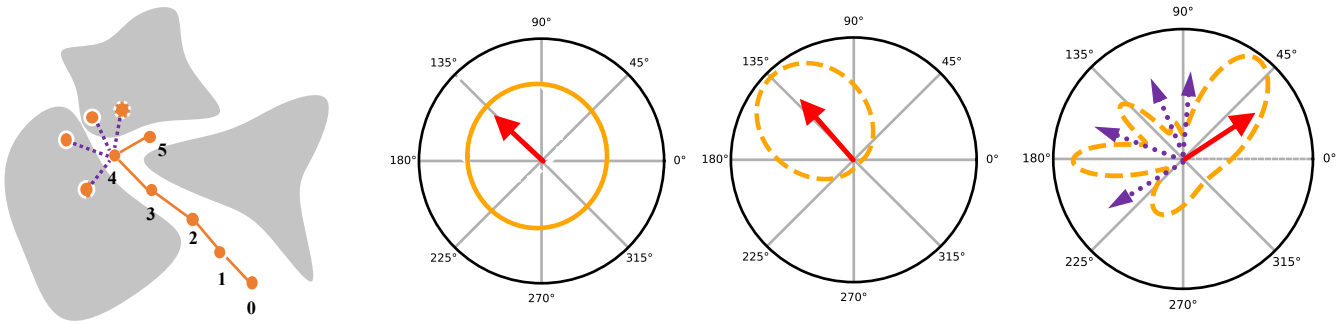

FIGURE 4: An example of sampling distribution updating. (a) shows how the subtree expands from $t = 0$ to $t = 5$, with solid orange lines indicating successful expansion and dashed purple lines indicating failed expansions. (b) depicts the uniform sampling at $t = 0$, with red arrow indicating the sampling 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 attempt and purple arrows representing failed attempts.

selected for expansion based on this probability distribution, considering the relative probabilities of each tree. When the energy of a subtree is exhausted, we regenerate the subtree in a new region, actively exploring new regions for path planning.

Algorithm 3: Pick Tree

Function `PickTree`$(E(\mathcal{T}))$**:**
  for $i = 1$ **to** $|\mathcal{T}|$ **do**
    $P(T_i) \leftarrow \frac{E(T_i)}{\sum_{i=1}^{|\mathcal{T}|} E(T_i)}$
  $P(\mathcal{T}) \leftarrow \{P(T_{start}), \cdots, P(T_{subm})\}$
  $T_i \leftarrow$ Choose from probabilistic distribution of $P(\mathcal{T})$
return T_i

3.3 Adaptive Local Subtree Planning

After selecting a tree based on its energy probability $E(\mathcal{T})$, the next step is to determine whether the selected tree is T_{sub}. If it is, then the process of subtree planning involves determining the sampling direction (d) and expansion step size (ϵ) to extend the current configuration ($x_{current}$) to a new configuration (x_{new}), i.e., $x_{new} \leftarrow x_{\text{current}} + \epsilon \cdot d$, with a higher probability of success. However, previous studies rarely considered to determine the sampling distribution and expansion step size adaptively during subtree planning in an integrated way.

To address this gap, we propose an adaptive local subtree planning algorithm 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 approach effectively leverages information from failed planning points that is often overlooked in traditional methods, providing valuable insight into future planning processes. Furthermore, we leverage past expansion 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 introducing the specific methods, some basic concepts need to be clarified. In our approach, the subtree planner is modeled using the state $\mathcal{S}_{T_i,t}$, where $T_i \in \mathcal{T}$ represents the spatial location and information set in the local planning step t. This state can be represented as the tuple $\mathcal{S}_{T_i,t} = (x_{i,t}, \mathcal{D}_{s,t}, \mathcal{D}_{f,t}, d_{i,t}, \epsilon_{i,t}, \mathcal{B}_i(R))$, with $x_{i,t}$ being the spatial location, $\mathcal{D}_{s,t}$ representing the previously successful unit vector value, $\mathcal{D}_{f,t}$ recording the failed unit vector value, $d_{i,t}$ being the sampling direction, $\epsilon_{i,t}$ being step size and $\mathcal{B}_i(R)$ representing the hypersphere with radius R. Algorithm 4 presents the pseudo-code for adaptive local subtree planning.

Algorithm 4: Adaptive Local Subtree Sampling

Function `AdaptiveLocalSubtreeSampling`(T_i, γ, n, dim, R)**:**
  $S_{T_i,t} \leftarrow$ Get state of T_i
  $\overline{|\mathcal{D}_f|} \leftarrow$ Average number of failures in $\mathcal{B}(R)$
  $\epsilon_i \leftarrow \gamma(log(n)/n)^{\frac{1}{dim}} g(\overline{|\mathcal{D}_f|})$
  if T_i *has previous* j*th sampling history at* t $(t > 0)$ **then**
    if j=1 **then**
      $\mathcal{D}_{prior}(\theta|\theta_{t-1}) \leftarrow$ `VonMisesFisherDistribution`
      $d_{i,t,1} \leftarrow \mathcal{D}_{prior}(\theta|\theta_{t-1})$
    else
      $\mathcal{D}_{i,t,j+1}(\theta|\theta_{t-1}) \leftarrow$`BayesianUpdateDistribution`
      $d_{i,t,j} \leftarrow \mathcal{D}_{i,t,j+1}(\theta|\theta_{t-1})$
  else
    $d_{i,0,1} \leftarrow$ `UniformSampling` ^{dim}
  $x_{new} \leftarrow x_{current} + \epsilon_i \cdot d_i$
return x_{new}

3.3.1 Sampling distribution

For sampling direction $d_{i,t}$, we propose a method that uses Bayesian update rules [20] 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 samples to update the sampling distribution. Figure 4 shows an example of sampling distribution updating.

To effectively inherit insights from successful sampling directions 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 $d_{i,t,0} \sim \mathcal{D}_{i,t,0}(\theta) = p(\theta;\kappa,\mu)$ [24]. The von Mises-Fisher distribution is a probability distribution on the unit sphere, specifically designed to model directional data.

$$p(\theta;\kappa,\mu) = \frac{1}{2\pi I_v(\kappa)}\exp(\kappa[\theta-\mu]) \tag{5}$$

$$I_v(\kappa) = \frac{1}{\pi}\int_0^{\pi}\exp(\kappa\cos\theta)\cos(v\theta)d\theta \tag{6}$$

where θ is the random angular variable, μ is the average direction, i.e., the previously sampled successful direction, where $-\pi \le \theta,\mu < \pi$, $v = 0$. κ is a measure of the concentration of the probability density function, located in the semi-infinite interval $[0,\infty)$. The larger κ means the selection of θ is more concentrated in the mean direction. $I_v(\kappa)$ 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:

$$\mathcal{D}_{i,t}(\theta) = \mathcal{D}_{posterior}(\theta) \propto \mathcal{D}_{likelihood}\cdot\mathcal{D}_{prior}. \tag{7}$$

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 $\mathcal{D}_{likelihood} \propto 1-k(\cdot)$. The kernel function $k(\cdot)$ is formulated as follows:

$$k(\theta,\theta_{i,t,j}) = \sigma_f^2\exp\left(-\frac{2}{\ell_{eq}^2}\sin^2\left(\pi\frac{\theta-\theta_{i,t,j}}{p}\right)\right) \tag{8}$$

where $\theta_{i,t,j}$ represents the jth sampling of subtree i at time t, ℓ_{eq} is the length scale and σ_f^2 is the scaling factor, corresponding to the period of the spherical distribution, and repetition period p is set to 2π. Compared to previous work, we introduce the parameter ℓ_{eq}, which is positively correlated with the size of the set $\mathcal{D}_{f,t}$. This approach offers the advantage of making sampling more exploratory after multiple failures, which in turn helps in finding feasible direction in complex and narrow environments. The length scale ℓ_{eq} is calculated as:

$$\ell_{eq} = \alpha + \beta\left(1-e^{-|\mathcal{D}_{f,t}|}\right) \tag{9}$$

where α is a constant term, β is a coefficient term and $|\cdot|$ represents the size of the set $\mathcal{D}_{f,t}$.

Therefore, we can rewrite the update of the sampling distribution

$$\begin{aligned}&\mathcal{D}_{i,t,j+1}(\theta)\\ &= \mathcal{D}_{i,t,j+1}\left(\theta \mid \theta_{t-1}, \mathcal{D}_{f,t} \in \mathcal{S}_{T_{i,t}}\right)\\ &= \mathcal{D}_{i,t,j}\left(\theta \mid \theta_{t-1}, \mathcal{D}_{f,t} \in \mathcal{S}_{m_{i,t}}\right)\sigma_{j+1}\left(1-k\left(\theta,\theta_{i,t,j}\right)\right)\end{aligned} \tag{10}$$

where σ_{j+1} is the scaling factor, j is larger than 0. It is important to note that, $\mathcal{D}_{t,i,1}(\theta \mid \theta_{t-1}, \mathcal{D}_{f,t} \in \mathcal{S}_{T_i,t})$ reduces to $\mathcal{D}_{\text{prior}}(\theta \mid \theta_{t-1}, \mathcal{D}_{f,t} := \emptyset)$. Finally, we get the sampling direction $d_{i,t,j+1} \sim \mathcal{D}_{i,t,j+1}(\theta)$. In summary, our method can adaptively update the sampling distribution based on historical results of sampling.

3.3.2 Step size

In our approach, we consider the selection of step size in sampling comprehensively at the regional scale to incorporate the information from the surrounding environment. To adjust the step size, we consider the previous sampling results of the local 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.

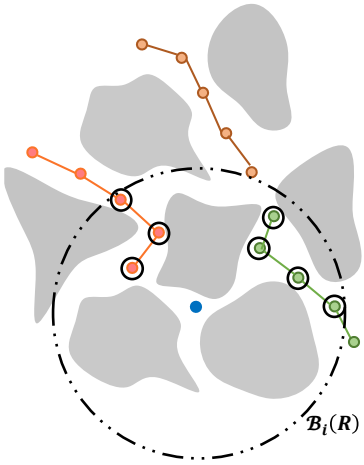

FIGURE 5: An illustration of the step size selection.The process centers on the new subtree, represented by the blue point. State information is collected from other subtrees within the range of $\mathcal{B}_i(R)$, which are highlighted by small black circles. The average number of failures, represented by $\overline{|\mathcal{D}_f|}$, is then calculated.

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

$$\varepsilon_i = \gamma\left(\frac{\log(n)}{n}\right)^{\frac{1}{dim}} g(\overline{|\mathcal{D}_f|}) \tag{11}$$

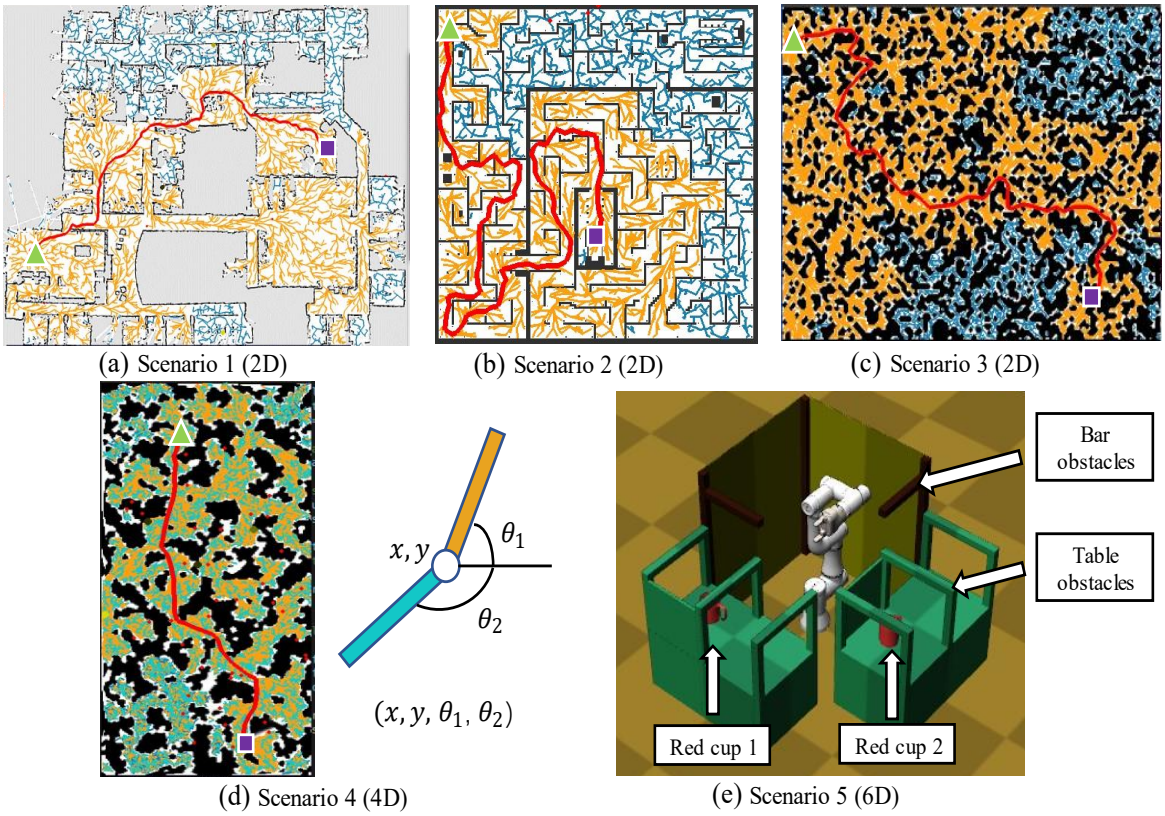

FIGURE 6: Three 2D (a, b, c), one 4D (d) and one 6D (e) experimental scenarios in our study.

where ε_i is the step size, r is a constant, n is the number of nodes, dim is the dimensionality of the planning problem and $g(|\overline{\mathcal{D}_f}|)$ is a function that maps the complexity of the state information within the hypersphere to a value between 0 and 1, reflecting the complexity of the surrounding environment. The function $g(|\overline{\mathcal{D}_f}|)$ is defined as:

$$g(\overline{D_f}) = \ell e^{-|\overline{\mathcal{D}_f}|/E} \tag{12}$$

where ℓ is a positive constant that controls the rate of step size adjustment based on regional integration, and E is the energy of the subtree. Our approach can optimize step size selection by integrating previous expansion information across the predefined region.

4. Experiments

4.1 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 robot arm, respectively. Additionally, we test our approach in a real material pick-and-place scenario typically encountered in manufacturing environments.

Figure 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., $(x, y, \theta_1, \theta_2)$ shown in Fig.6(d)). The 6D scenario presents a

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

Algorithms	Scenario1 (2D)	Scenario2 (2D)	Scenario3 (2D)	Scenario4 (4D)	Scenario5 (6D)
AMRRT	79.6	**62.6**	**623.6**	**784.4**	**392.3**
RRT	234.6	953.7	8130.4	1996.2	-
RRT-Connect	101.9	494.8	5040.2	984.1	1391.8
BIT	72.4	599.5	-	2494.3	2127.4
RRdT	95.3	86.1	815.6	901.0	572.6
LMRRT	**68.2**	278.9	4000.2-	2100.7	-

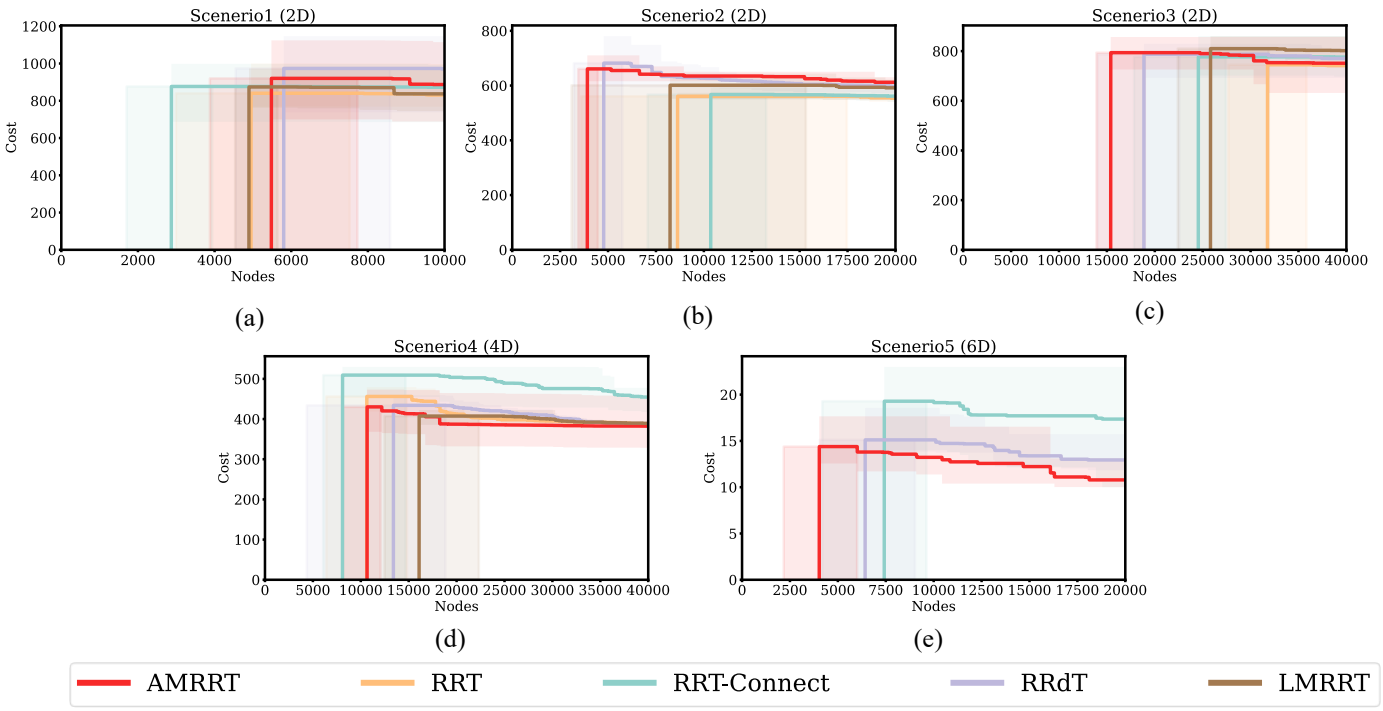

FIGURE 7: Cost values against the number of sampling nodes for different algorithms. The solid line represents the mean value, while the shaded area indicates the range between the maximum and minimum values. It is worth noting that when the algorithm fails to find an initial solution, the cost is set to 0. Due to the batch sampling nature of the BIT algorithm, it is the least sampled points in all scenarios for finding the initial solution and cannot be plotted in this figure like other algorithms.

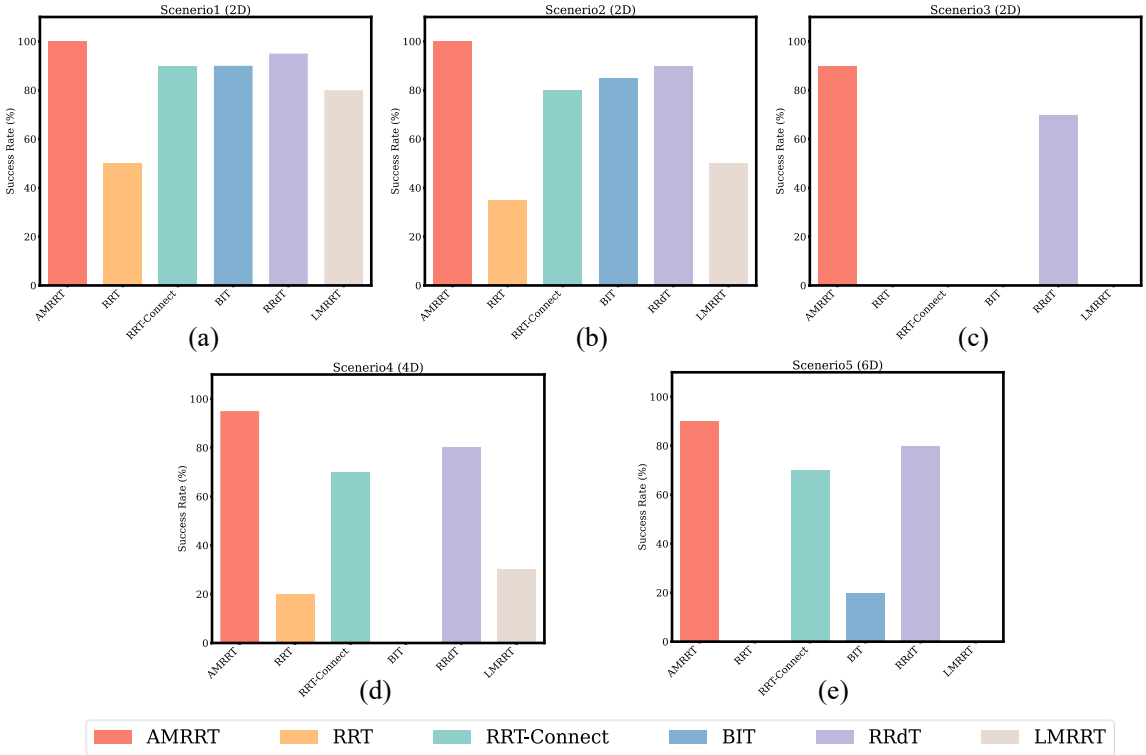

FIGURE 8: Average success rate of different algorithms under a specified planning time limit. (a) is limited to 150s, (b) is limited to 500s. (c), (d) and (e) are limited to 1000s. If the histogram of an algorithm is not plotted in the figure, it means that this algorithm is unable to generate paths successfully within the given time limit.

task of moving a robot arm between two tables constructed on the Klapmt platform [25]. The six degrees of freedom of the robot in this scenario include rotational degrees of six joints. In the 2D and 4D scenarios, the starting points (x_{start}) are depicted as green triangles, while the goal points (x_{goal}) are represented by purple squares. 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, RRT-Connect, LM-RRT [19], RRdT [26], and Batch-Informed RRT (BIT) [27]. The BIT algorithm employed a selection of 20 sampling points for a single batch. 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.

4.2 Experiment results

After conducting a series of experiments for our proposed approach and the compared algorithms, we depict the relationship between cost values and the number of sampling nodes in Figure 7. Here the cost values are the sum of the distances of the points in the final solution. The average running time for different algorithms to obtain initial solutions are compared and summarized in Table 1. The success rates of obtaining initial solutions under diverse constraints for these algorithms are compared in Figure 8. In the following sections, we will provide a comprehensive analysis of the performance of the proposed approach across different experimental scenarios.

(1) 2D scenarios with mass point

In this study, we test three different types of 2D scenarios as shown in Fig. 6 (a), (b) and (c). Scenario 1 is an indoor environment where the prime robot must navigate through multiple rooms and obstacles to reach the target goal. Scenario 2 illustrates a maze environment with numerous turns and dead ends. Scenario 3 is a noisy map with cluttered obstacles. The planning difficulty of these three scenarios increases according to their respective (α, β, ϵ)-expansiveness [28].

In Figure 7 (a), it is evident that 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 pre-processing 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 stable than that of the proposed AMRRT approach, as shown in Figure 8 (a). Furthermore, in complex congested environments, as illustrated in Figures 7 (b) and 7 (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 more complex 2D environments where RRT, RRT-connect and BIT algorithms cannot even obtain initial solutions, indicating the effectiveness of our information gain-based subtree generation and adaptive local subtree planning. Additionally, Figures 8 (b), (c), and (d) further highlight the superior stability of our approach compared to other algorithms. Notably, the information gain-based subtree generation in the proposed AMRRT approach eliminates the need for preprocessing the map environment and avoids over-concentration on already fully explored regions, resulting in a higher success rate within limited time compared to other 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.

Figure 7 (d) illustrates that the performance of the proposed AMRRT approach is between the RRT and RRT-Connect algorithms in terms of the number of valid samples required to find a feasible solution. Furthermore, Table 1 confirms that the AMRRT approach exhibits a lower time cost of sampling valid points compared to other algorithms. Figure 8 (d) demonstrates that multi-tree structure-based algorithms exhibit higher success rates compared to RRT, RRT-Connect, and BIT algorithms. Specifically, the proposed AMRRT approach performs the best among the multi-tree structure algorithms, indicating its adaptive nature and superior performance even in a 4D environment. In complex 4D environment, the AMRRT approach outperforms other algorithms in terms of sampling efficiency, planning efficiency and success rate.

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

In Scenario 5 (see Fig. 6 (e)), a robot arm with a single-degree-of-freedom mechanical gripper need 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.

Figure 7 (e) shows that the proposed AMRRT approach only requires 4000 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, Figure 8 (e) demonstrates that AMRRT has 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 solutions for the 6D scenario, probably because it struggles to identify key points for subtree generation during preprocessing. In summary, the experimental findings highlight the superior performance of the proposed AMRRT approach in high-dimensional complex environment, further validating its adaptability and stability.

(4) Realistic material pick-and-place scenario

In order to validate the effectiveness of the AMRRT approach in real industrial applications, we test the performance of AMRRT in a material pick-and-place scenario as shown in Figure 9. The scenario includes a Jaka Zu3 robot arm equipped with a mechanical gripper, and two Realsense cameras for accurate object detection and positioning (two cameras are set on each side of the table, although only one is shown in the figure). In the material pick-and-place task, the robot arm will transfer the material block from one side of the baffle plate to the other side while avoiding the obstacles of the baffle and the conical barrel. We first create the simulation environment using the Klampt platform and implement the AMRRT approach for path planning on the physical robot arm. The successful trajectory planned by the AMRRT approach is also illustrated in Figure 9 (b), demonstrating the capability of the proposed approach in a realistic industrial application.

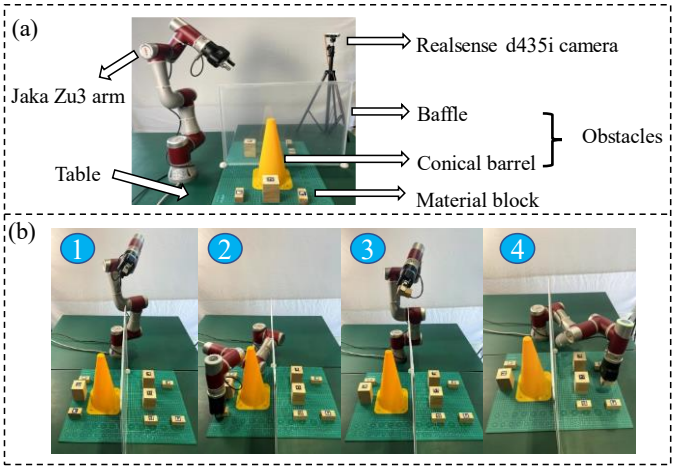

FIGURE 9: The realistic material pick-and-place scenario (a) and the major motions of the robot arm (b). The sequential pictures labeled with 1, 2, 3, and 4 correspond to the seeking, picking, moving, and placing actions in the pick-and-place task.

5. CONCLUSION

In this paper, we propose a robot motion planning approach based on adaptive multi-tree sampling to efficiently generate collision-free paths in highly constrained environments. Our approach leverages the exploration structure of the multi rapidly exploring random tree, combining the rapid exploration property of the RRT method and the global exploration property of the multi-tree structure. We develop an information gain-based subtree generation method utilizing the mean shift algorithm, which can select locations with higher information gain to generate subtrees to explore the environment more effectively. In addition, we develop an adaptive local subtree planning method that dynamically updates the sampling direction and step size based on local planning results, maximizing the likelihood of forming feasible trajectories in narrow passages.

The results from computer simulation experiments demonstrate that the proposed approach outperforms other methods in complex environments, particularly in success rate of path planning and computational time to find initial solutions. The physical experiment also validates the effectiveness of our approach in realistic material pick-and-place scenario. Future work may involve further optimizing the algorithm's computational efficiency while preserving its high-quality solutions and exploring the algorithm's performance in more diverse and challenging scenarios, such as environments with moving obstacles. Our proposed approach offers promising solutions for adaptive robot motion planning in complex environments, and we hope our work will inspire the development of more advanced methods in this area.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] Evjemo, L. D., Gjerstad, T., Grøtli, E. I., and Sziebig, G., 2020, "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories," Current Robotics Reports, **1**(2), pp. 35–41.

[2] Inkulu, A. K., Bahubalendruni, M. V. A. R., Dara, A., and K., S., 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), pp. 226–239.

[3] Karaman, S., and Frazzoli, E., 2011, "Sampling-Based Algorithms for Optimal Motion Planning," Int J Rob Res, **30**(7), pp. 846–894.

[4] Gammell, J. D., and Strub, M. P., 2021, "Asymptotically Optimal Sampling-Based Motion Planning Methods," Annu Rev Control Robot Auton Syst, **4**(1), pp. 295–318.

[5] Elbanhawi, M., and Simic, M., 2014, "Sampling-Based Robot Motion Planning: A Review," IEEE Access, **2**, pp. 56–77.

[6] LaValle, S. M., 1998, "Rapidly-Exploring Random Trees: A New Tool for Path Planning."

[7] Wilmarth, S. A., Amato, N. M., and Stiller, P. F., 1999, "MAPRM: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, IEEE, pp. 1024–1031.

[8] Chen, G., Luo, N., Liu, D., Zhao, Z., and Liang, C., 2021, "Path Planning for Manipulators Based on an Improved Probabilistic Roadmap Method," Robot Comput Integr Manuf, **72**, p. 102196.

[9] Yershova, A., Jaillet, L., Simeon, T., and Lavalle, S. M., 2006, "Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain," *IEEE International Conference on Robotics & Automation*.

[10] Dalibard, S., and Laumond, J.-P., 2009, "Control of Probabilistic Diffusion in Motion Planning," pp. 467–481.

[11] Burns, B., and Brock, O., 2007, "Single-Query Motion Planning with Utility-Guided Random Trees," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3307–3312.

[12] Ma, H., Liu, J., Meng, F., Pan, J., Wang, J., and Meng, M. Q.-H., 2021, "A Nonuniform Sampling Strategy for Path Planning Using Heuristic-Based Certificate Set," *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1359–1366.

[13] Plaku, E., Bekris, K. E., Chen, B. Y., Ladd, A. M., and Kavraki, L. E., 2005, "Sampling-Based Roadmap of Trees for Parallel Motion Planning," IEEE Transactions on Robotics, **21**(4), pp. 597–608.

[14] Otte, M., and Correll, N., 2013, "C-FOREST: Parallel Shortest Path Planning With Superlinear Speedup," IEEE Transactions on Robotics, **29**(3), pp. 798–806.

[15] Wang, W., Xu, X., Li, Y., Song, J., and He, H., 2010, "Triple RRTs: An Effective Method for Path Planning in Narrow Passages," Advanced Robotics, **24**(7), pp. 943–962.

[16] Lai, T., Ramos, F., and Francis, G., 2019, "Balancing Global Exploration and Local-Connectivity Exploitation with Rapidly-Exploring Random Disjointed-Trees," *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 5537–5543.

[17] Sun, Z., Wang, J., and Meng, M. Q.-H., 2022, "Multi-Tree Guided Efficient Robot Motion Planning," Procedia Comput Sci, **209**, pp. 31–39.

[18] Ichter, B., Harrison, J., and Pavone, M., 2018, "Learning Sampling Distributions for Robot Motion Planning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 7087–7094.

[19] Wang, W., Zuo, L., and Xu, X., 2018, "A Learning-Based Multi-RRT Approach for Robot Path Planning in Narrow Passages," J Intell Robot Syst, **90**(1–2), pp. 81–100.

[20] Lai, T., Morere, P., Ramos, F., and Francis, G., 2020, "Bayesian Local Sampling-Based Planning," IEEE Robot Autom Lett, **5**(2), pp. 1954–1961.

[21] Khan, A., Ribeiro, A., Kumar, V., and Francis, A. G., 2020, "Graph Neural Networks for Motion Planning."

[22] Qureshi, A. H., Miao, Y., Simeonov, A., and Yip, M. C., 2021, "Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners," IEEE Transactions on Robotics, **37**(1), pp. 48–66.

[23] Comaniciu, D., and Meer, P., 2002, "Mean Shift: A Robust Approach toward Feature Space Analysis," IEEE Trans Pattern Anal Mach Intell, **24**(5), pp. 603–619.

[24] Banerjee, A., Dhillon, I. S., Ghosh, J., and Sra, S., 2005, "Clustering on the Unit Hypersphere Using von Mises-Fisher Distributions," J. Mach. Learn. Res., **6**, pp. 1345–1382.

[25] Hauser, K. K., 2013, "Robust Contact Generation for Robot Simulation with Unstructured Meshes," *International Symposium of Robotics Research*.

[26] Lai, T., 2021, "Sbp-Env: A Python Package for Sampling-Based Motion Planner and Samplers," J Open Source Softw, **6**(66), p. 3782.

[27] Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S., 2020, "Batch Informed Trees (BIT*): Informed Asymptotically Optimal Anytime Search," Int J Rob Res, **39**(5), pp. 543–567.

[28] Hsu, D., Latombe, J.-C., and Motwani, R., 1997, "Path Planning in Expansive Configuration Spaces," *Proceedings of International Conference on Robotics and Automation*, pp. 2719–2726 vol.3.