HD-RRT*former: Sampling-Based Motion Planning for High-Dimensional Systems using Transformer [⋆]

Mingyang Feng, Jianing Zhao, Shaoyuan Li and Xiang Yin [∗]

[∗] *School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China.*
E-mail: `{Fmy-135214,jnzhao,syli,yinxiang}@sjtu.edu.cn`

Abstract: In this paper, we investigate the sampling-based motion planning for *high-dimensional* robotic systems in complex environments. Most existing sampling-based approaches make limited use of environmental information or previous sampling states, even though such information essentially provides valuable heuristics for guiding subsequent samples. To this end, we present *HD-RRT*former*, an efficient sampling-based motion planning algorithm which integrates the standard RRT* algorithm with a Transformer architecture, enabling autoregressive guidance of the entire sampling process. To be specific, we first introduce a *physically-informed kinematic graph attention network* (PI-KGAN) to approximate the environmental representations in the high-dimensional space, based on which, a two-stage training method is developed to achieve fast convergence in spite of the large sampling space and numerous invalid samples. We conducted extensive simulation experiments on both 6-DOF UR3 and 7-DOF Franka manipulators to validate the algorithm's performance, and finally deployed our algorithm on a real manipulator. Experiments show that our algorithm outperforms the existing variants of RRT algorithms in multiple metrics, including sampling efficiency and path quality. The relevant code can be found at `https://github.com/fengmingyang666/HD-RRT-former`.

Keywords: Motion planning, sampling method, high-dimensional systems

1. INTRODUCTION

Motion planning in complex environments is a fundamental challenge in robotics and autonomous systems (LaValle and Kuffner Jr, 2001). The central goal of this problem is to navigate a robot from a starting point to a goal point while avoiding collisions with obstacles in the environment. Solving this type of problem has important applications in warehousing and logistics (Butler et al., 2024), autonomous driving (Wang et al., 2022), and factory manipulators (Huh et al., 2018).

Traditional path planning algorithms include search-based methods, such as A* and D* (Stentz et al., 1995), and sampling-based algorithms, such as Rapidly-exploring Random Trees (RRT) and RRT* (LaValle, 2006). The A* algorithm can compute an optimal solution in the configuration space given a well-designed heuristic function. However, designing an effective heuristic becomes extremely challenging in complex planning spaces, and the search process often suffers from a combinatorial explosion as the complexity of the environment grows. In contrast, sampling-based algorithms naturally adapt to irregular and cluttered environments and do not rely on handcrafted heuristics. Nevertheless, due to their inherent randomness, the unguided sampling may result in extensive and inefficient and useless exploration.

In order to improve the efficiency of sampling-based motion planners, a line of work has focused on designing heuristic exploration strategies (Gammell et al., 2014; Tahir et al., 2018). These methods aim to reduce redundant exploration by introducing biased sampling distributions tailored to the task or environment. While effective in specific scenarios, the heuristics must often be redesigned for different environments or planning requirements, which limits their generality and scalability. To overcome these limitations, various learning-based sampling approaches have been proposed (Wang et al., 2020; Liu et al., 2024). These methods learn neural networks as data-driven heuristic functions that guide sampling based on a global perception of the environment. A growing body of work has demonstrated strong performance in planning problems (Johnson et al., 2021; Chaplot et al., 2021; Huang et al., 2024).

However, since global environmental information lacks temporal structure, previously explored regions may still be repeatedly sampled, as the network has no explicit mechanism to account for sampling history. This redundancy ultimately degrades efficiency. To address this issue, subsequent research incorporated historical sampling states as prior information into the network (Qureshi et al., 2020). More recent works Johnson et al. (2023); Feng et al. (2025) further employ Transformer architectures to capture long-term dependencies within sampling sequences. These approaches leverage the temporal relationships be-

[⋆] This work was supported by the National Natural Science Foundation of China (62173226, 62061136004).

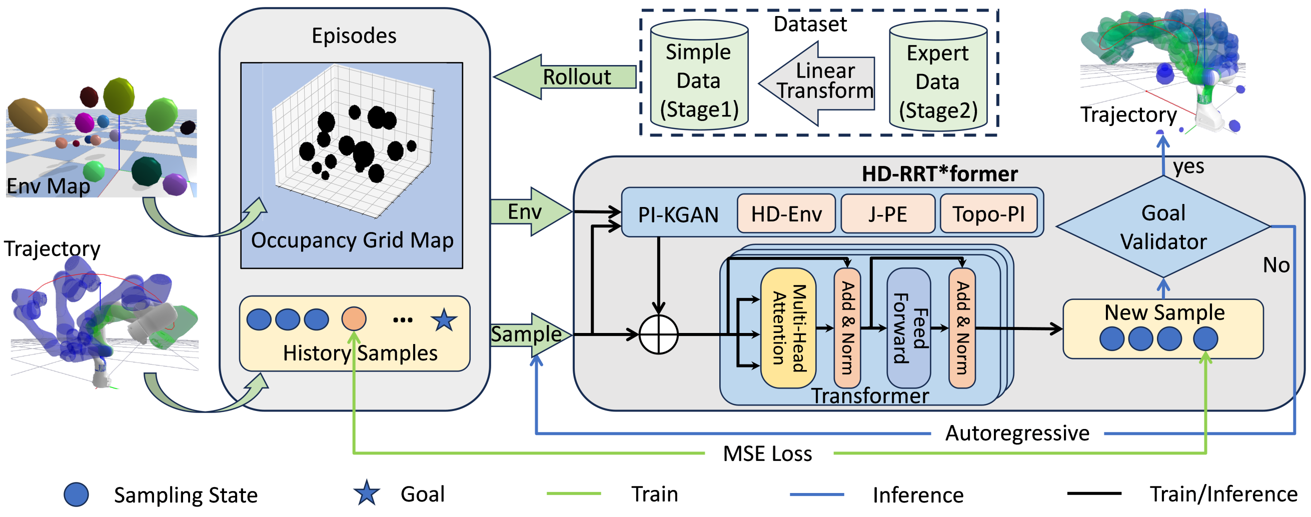

Fig. 1. HD-RRT*former Sampler. The model inputs historical sampling information and 3D environmental information and outputs a new sample. In different training phases, training data is extracted from different datasets to perform single-step training; in the inference phase, an autoregressive approach is used to output the entire sampling path.

tween sampling states, improving both sampling efficiency and path quality.

Despite these advances, applying such techniques to high-dimensional systems remains challenging. For instance, manipulators with complex kinematics operate in a high-dimensional configuration space while interacting with a cluttered 3D workspace. In such settings, the sampling space becomes extremely large, causing the performance of RRT-based methods to deteriorate dramatically (Akgun and Stilman, 2011). The probability of generating informative samples decreases with dimensionality, feasible extensions become harder to find, and tree growth slows as the algorithm spends substantial effort exploring irrelevant regions. To alleviate this issue, RRT-Connect (Kuffner and LaValle, 2000) is often employed as a practical high-dimensional motion planner. In high-dimensional configuration spaces, sparsity increases the likelihood that two distant states can be directly connected. Thus, once the *connect* step successfully finds a feasible link between the two trees, the planner can terminate immediately with a complete solution. Nevertheless, standard RRT-Connect still relies on uniform sampling and does not exploit environmental context or historical sampling information, which limits its efficiency in complex high-dimensional planning problems.

The major challenge in leveraging environmental information with learning-based methods in high-dimensional spaces lies in defining a safe sampling domain within the configuration space. Since obstacles reside in the workspace while sampling must be conducted in the configuration space, it is inherently difficult to characterize an exact high-dimensional safe region that could serve as an ideal sampling domain. Consequently, it remains an open problem how to effectively capture environmental information in high-dimensional configuration spaces and, based on that, construct an efficient exploration strategy for high-dimensional planning.

In this work, we address the above challenges by introducing a unified, data-driven framework, HD-RRT*former, that jointly learns a high-dimensional environmental representation and an efficient sampling strategy tailored for complex robotic systems. Built upon the RRT-Connect algorithm, our approach effectively exploits environmental and kinematic information, along with historical sampling states, through a Transformer-based architecture. The overall structure of the proposed algorithm is illustrated in Fig. 1. Our main contributions are summarized as follows:

- We propose a physics-informed neural module (PI-KGAN) that approximates the environmental structure directly within the configuration space, enabling the planner to reason about obstacle geometry in a representation consistent with the robot's kinematics.
- Leveraging the learned representation, we introduce HD-RRT*former, the Transformer-based sampler that integrates both environmental encoding and historical samplings to guide exploration efficiently.
- We develop a two-stage training method designed to mitigate the large proportion of invalid samples and the slow convergence commonly encountered in high-dimensional planning tasks.

Furthermore, we conduct extensive simulation studies comparing our method with several RRT variants, including RRT, RRT*, and RRT-Connect. The experimental results demonstrate that our approach achieves superior performance across multiple metrics, including planning time, the number of generated nodes, trajectory length, and especially trajectory smoothness. Finally, we deploy the proposed algorithm on a real UR3 manipulator, further validating its practical feasibility.

The remainder is organized as follows. Section 2 formally defines the high-dimensional motion planning problem. Section 3 introduces the overall architecture of HD-RRT*former, and Section 4 describes the proposed two-stage training method. Simulation and real-world experimental results are presented in Section 5. Finally, the conclusion is drawn in Section 6.

2. PROBLEM FORMULATION

Let $\mathcal{Q} \subseteq \mathbb{R}^N$ be the joint configuration space of an N-DOF manipulator. Given a joint configuration

$$\mathbf{q} = [q_1, q_2, \ldots, q_N]^\top \in \mathcal{Q}.$$

We denote by $f_{ee} : \mathcal{Q} \to SE(3)$ the forward kinematics map that returns the end-effector pose for a given joint configuration $\mathbf{q}$, that is,

$$f_{ee}(\mathbf{q}) = \begin{bmatrix} R(\mathbf{q}) & \xi(\mathbf{q}) \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where $\xi(\mathbf{q}) \in \mathbb{R}^3$ and $R(\mathbf{q}) \in SO(3)$ denote the position and orientation of the end-effector, respectively.

In a similar manner, we could define the forward kinematics for any body element of the manipulator. For convenience, we denote by $\mathcal{B}(\mathbf{q}) \subset \mathbb{R}^3$ the body geometry of the manipulator at configuration $\mathbf{q}$.

Consider a complex environment in the workspace with obstacles $\mathcal{O} \subset \mathbb{R}^3$. Then, we define

$$\mathcal{Q}_{free} = \{\mathbf{q} \in \mathcal{Q} : \mathcal{B}(\mathbf{q}) \cap \mathcal{O} = \emptyset \wedge \text{no self-collision}\}. \tag{2}$$

Given an initial configuration $\mathbf{q}_s$ and a goal configuration $\mathbf{q}_g$ in the workspace, the objective of motion planning is to compute a path (trajectory)

$$\mathcal{P} = \{\mathbf{q}_i\}_{i=0}^n, \ \mathbf{q}_i \in \mathcal{Q},$$

satisfying

$$\begin{aligned} &\mathbf{q}_0 = \mathbf{q}_s, && \text{(3a)} \\ &\mathbf{q}_i \in \mathcal{Q}_{free}, \forall i = 0, 1, \ldots, n, && \text{(3b)} \\ &\text{Line}(\mathbf{q}_i, \mathbf{q}_{i+1}) \subset \mathcal{Q}_{free}, && \text{(3c)} \\ &\mathbf{q}_n = \mathbf{q}_g, && \text{(3d)} \end{aligned}$$

where $\text{Line}(\mathbf{q}_i, \mathbf{q}_{i+1})$ denotes the linear trajectory between two configurations $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$. Here we assume that such linear interpolation is dynamically feasible. This assumption is made without loss of generality, as one can always ensure feasibility by choosing sufficiently small intervals between successive samples $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$.

To evaluate the quality of a path $\mathcal{P}$, we define its cost as the cumulative Euclidean distance between two consecutive states, i.e.,

$$\text{Cost}(\mathcal{P}) = \sum_{i=1}^{n} \|\mathbf{q}_i - \mathbf{q}_{i-1}\|_2. \tag{4}$$

Furthermore, another important quality of a planned path is smoothness, as smoother paths are generally easier to execute on real robotic systems. To this end, we adopt the notion of *joint acceleration* (Gasparetto and Zanotto, 2007) to measure the smoothness of a path, which is defined as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{N} \left\| \mathbf{q}_{i+1}^j - 2\mathbf{q}_i^j + \mathbf{q}_{i-1}^j \right\|^2, \tag{5}$$

where $\mathbf{q}_i^j$ denotes the j-th component of $\mathbf{q}_i$.

Let $\mathcal{Q}_{tree}(t) = \{\mathbf{q}_i \mid \mathbf{q}_i \in \mathcal{Q}, i < t\}$ denote the set of previously sampled nodes up to step t. Based on the principles of sampling-based algorithms, $\mathcal{Q}_{tree}$ is constructed incrementally by sampling nodes from the configuration space. The quality of the samples generated during this process directly impacts the algorithm's convergence rate and the path's length and smoothness. Therefore, it is crucial to design a sampling strategy that efficiently avoids obstacles. That is, we aim to design a Sampler that generates the next node $\mathbf{q}_{next}$ based on the environment with obstacles $\mathcal{O}$ and the past samples in $\mathcal{Q}_{tree}(t)$. Formally, we express it as

$$\mathbf{q}_{next} = \mathsf{Sampler}(\mathcal{O}, \mathcal{Q}_{tree}(t)), \tag{6}$$

which guides the planning algorithm to explore the configuration space with the constraints of the environment.

Since the obstacles $\mathcal{O}$ reside in the workspace in $\mathbb{R}^3$ while the next sample $\mathbf{q}_{next}$ must be drawn in the configuration space $\mathcal{Q}$, it is difficult to characterize an exact $\mathcal{Q}_{free}$ as the ideal sampling domain strictly according to (2). To this end, we are motivated to first train a neural network that approximates the environmental representation in the high-dimensional configuration space $\mathcal{Q}$. Building on this approximation, we then develop a new data-driven approach for designing the sampling process (6).

3. HD-RRT*FORMER ALGORITHM

In this section, we present our main sampling-based algorithm, HD-RRT*former. We adopt the standard RRT-Connect algorithm as the underlying framework for trajectory sampling. RRT-Connect simultaneously grows two trees from the start and goal states and repeatedly attempts to connect them directly, which significantly accelerates convergence, especially in relatively sparse configuration spaces. When generating the next state during tree expansion, we utilize a Transformer architecture to produce the sampling point. To enable this capability, we first introduce a neural network module, PI-KGAN, to encode a high-dimensional representation of the environment, and subsequently describe how this representation is integrated into the HD-RRT*former sampler.

3.1 High-Dimensional Representation

To embed the information of a complex environment with obstacles $\mathcal{O}$ into the high-dimensional configuration space $\mathcal{Q}$, we design an environmental representation module, termed *physically-informed kinematic graph attention network* (PI-KGAN), whose architecture is shown in Fig. 2. PI-KGAN incorporates three sources of information: 3D environmental features, spatial positions of the joints, and the kinematic structure of the manipulator, which are explained in detail as follows:

1) HD-Env: To encode the environmental information $\mathcal{O}$, we employ a neural network block HD-Env, which processes both obstacle geometry and free-space structure. This module consists of a 3D convolution block (Conv3D) followed by a global receptive field block (GlobalRF). The raw environment data are first processed by Conv3D to extract local 3D spatial features. These features are then passed through GlobalRF, which projects them into a high-dimensional latent space, producing the high-level environmental representation HD Env Feat. This design configures a globally shared, trainable convolution kernel for each joint, effectively emphasizing the receptive field associated with each joint and enabling the model to focus more on regions in the workspace that are sensitive to collisions when mapped back to configuration space.

2) Topo-PI: To capture the forward kinematics $\mathcal{B}(\mathbf{q})$ of the manipulator in a physically informed manner within the learning framework, we introduce a trainable topological chain module, referred to as Topo-PI. Topo-PI employs N cascaded linear layers to mirror the structure of an N-DOF manipulator. Each layer takes as input the previously expanded joint state features Q_{tree} and outputs a high-dimensional representation, denoted as PI Feat. With this chain-like architecture, each linear layer is capable of learning the kinematic characteristics associated with a specific joint, enabling the network to implicitly encode forward kinematics across the entire manipulator.

3) J-PE: To capture the inherent periodicity of joint angles, we encode the raw joint states using a cosine–sine–based positional encoding layer, called J-PE. This encoding projects each joint angle onto a set of Fourier features, effectively mapping it onto a continuous circular manifold. Such a representation eliminates the numerical discontinuities caused by angle wrap-around (e.g., 0 and 2π representing the same physical configuration), improves the stability of the learning process, and enhances the model's ability to generalize across configurations that are physically equivalent but numerically distinct in their raw angle values.

The above three components are then fused by a multi-layer perceptron MLP to obtain a unified high-dimensional embedding. The whole architecture of PI-KGAN captures both the robot's configuration and the surrounding environment geometry, enabling the model to reason about collision likelihood and reachable directions directly within the high-dimensional joint configuration space.

*3.2 HD-RRT*former Sampler*

We adopt the RRT-Connect algorithm as the baseline framework. Building upon it, we develop HD-RRT*former, whose architecture is shown in Figure 1, a new Sampler that significantly improves the sampling process by incorporating a Transformer architecture capable of leveraging both the environmental representation produced by PI-KGAN and the previously sampled data. The overall architecture of HD-RRT*former consists of three main components: i) environmental information processing, ii) Transformer autoregressive sampling, and iii) termination condition checking, all of which are explained in detail as follows:

1) Feature Extractor: We adopt the previously introduced network PI-KGAN, which incorporates both environmental information and the joint-space state information, thereby reducing the gap between the low-dimensional Cartesian workspace and the high-dimensional joint configuration space. Therefore, PI-KGAN produces a rich and expressive high-dimensional feature representation tailored for sampling-based planning.

2) Transformer Encoder: For the backbone sampling network, we adopt an autoregressive Transformer encoder architecture. The network takes as input the raw historical sampling states after a lightweight linear embedding, while the high-dimensional environmental representation produced by PI-KGAN is used to condition or "position" these states within the configuration space.

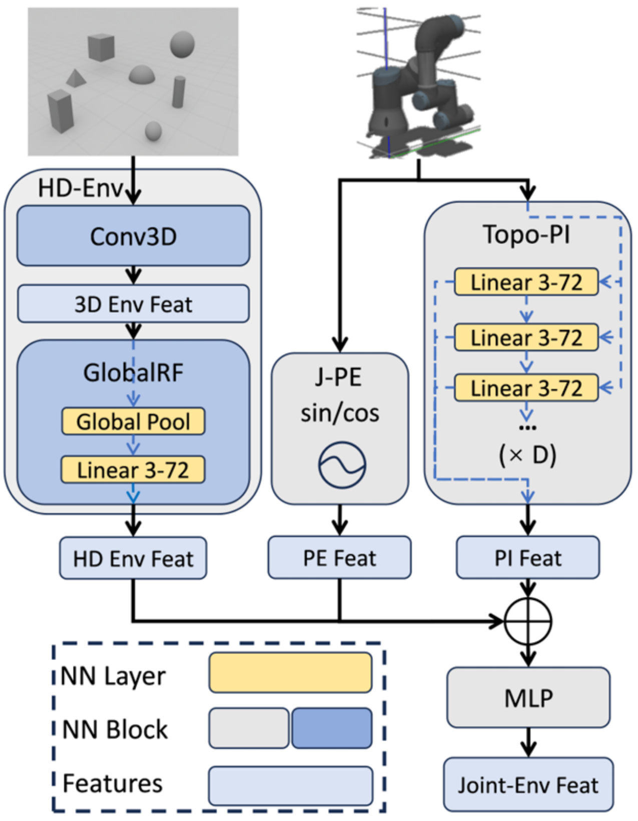

Fig. 2. Architecture of the Physics-Informed Kinematic Graph Attention Network(PI-KGAN). Environment-aware joint representations are fused with high-dimensional environment features generated by HD-Env, joint positional encodings generated by J-PE, and joint-topology embeddings generated by Topo-PI.

The Transformer relies on an attention mechanism that computes relevance between elements:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V.$$

where Q, K, and V denote the query, key, and value vectors of the original input. We define the input as

$$In = Embedding(\mathcal{Q}_{tree}, Env_{HD}),$$

which fuses the embedded historical sampling states $\mathcal{Q}_{tree}$ with the high-dimensional environmental representation Env_{HD}. The corresponding vectors are computed as:

$$Q = \mathcal{W}^Q \times In, \tag{7a}$$
$$K = \mathcal{W}^K \times In, \tag{7b}$$
$$V = \mathcal{W}^V \times In, \tag{7c}$$

where $\mathcal{W}^Q$, $\mathcal{W}^K$, and $\mathcal{W}^V$ are learnable parameter matrices. The scaling factor d_k corresponds to the key dimensionality, and the *softmax* operation normalizes the attention scores. After the multi-head attention, feed-forward layers, and normalization operations, the encoder generates the next sampling state.

This architecture, originally introduced in NLP for text generation (Vaswani et al., 2017), has proven highly effective for sequence modeling and temporal prediction due to its ability to capture long-range dependencies. Moreover, the Transformer is inherently agnostic to input and output dimensionality, which allows us to extend the sampling process directly to high-dimensional configuration spaces without modifying the core architecture.

3) Condition Validator: For the termination condition of the generated sampling states, we define two exit criteria. The first condition is that the start tree $\mathcal{Q}_{tree}^{start}$ becomes sufficiently close to the goal tree $\mathcal{Q}_{tree}^{goal}$, or vice versa. The second condition is that the two trees can be directly connected without violating collision constraints. Once either of these criteria is satisfied, a valid path $\mathcal{P}$ from the start state to the goal state can be extracted from the constructed trees.

Algorithm 1: HD-RRT*former Algorithm

```
Input: initial configuration $\mathbf{q}_s$, target configuration
       $\mathbf{q}_g$, environment $\mathcal{O}$, uniform sampling rate $\alpha$
Output: path $\mathcal{P}$
$\mathcal{Q}_{tree}^{start} \leftarrow \{\mathbf{q}_s\}$, $\mathcal{Q}_{tree}^{goal} \leftarrow \{\mathbf{q}_g\}$;
while NotConnected($\mathcal{Q}_{tree}^{start}, \mathcal{Q}_{tree}^{goal}$) do
    /* Randomly choose the active tree                */
    if $random(0,1) < 0.5$ then
        $\mathcal{Q}_{active} \leftarrow \mathcal{Q}_{tree}^{start}$, $\mathcal{Q}_{passive} \leftarrow \mathcal{Q}_{tree}^{goal}$;
    else
        $\mathcal{Q}_{active} \leftarrow \mathcal{Q}_{tree}^{goal}$, $\mathcal{Q}_{passive} \leftarrow \mathcal{Q}_{tree}^{start}$;
    end
    /* Take a new sample state                        */
    if $random(0,1) \leq \alpha$ then
        /* Uniform Sampler                            */
        $\mathbf{q}_{rand} \leftarrow UniformSampling(\mathcal{Q})$;
    else
        /* HD-RRT*former Sampler                      */
        $Env_{HD} \leftarrow PI\text{-}KGAN(\mathcal{O}, \mathcal{Q}_{active})$;
        $\mathbf{q}_{rand} \leftarrow Transformer(\mathcal{Q}_{active}, Env_{HD})$;
    end
    $\mathbf{q}_{nearest} \leftarrow Nearest(\mathcal{Q}_{active}, \mathbf{q}_{rand})$;
    $\mathbf{q}_{next} \leftarrow Steer(\mathbf{q}_{nearest}, \mathbf{q}_{rand}, \epsilon)$;
    if CollisionFree($\mathbf{q}_{nearest}, \mathbf{q}_{next}$) then
        $\mathcal{Q}_{active} \leftarrow \mathcal{Q}_{active} \cup \{\mathbf{q}_{next}\}$;
        /* Attempt to connect two trees               */
        Connect($\mathcal{Q}_{passive}, \mathbf{q}_{next}$);
    end
end
/* Construct the final path                           */
$\mathcal{P} \leftarrow GetBiDirectionalPath(\mathcal{Q}_{tree}^{start}, \mathcal{Q}_{tree}^{goal}, \mathbf{q}_s, \mathbf{q}_g)$;
```

The overall sampling procedure is summarized in Algorithm 1. To preserve the probabilistic completeness of the planner while still incorporating learned guidance, we introduce a parameter α that controls the proportion of uniform sampling. Specifically, each sample is drawn from uniform random sampling with probability α, and from the HD-RRT*former sampler with probability $1 - \alpha$.

4. TRAINING METHOD

In this section, we describe the design of the overall training process, including the construction of the high-dimensional dataset, the development of an efficient two-stage training strategy for HD-RRT*former, and the loss functions used during training. The complete training and inference pipelines have been illustrated in Figure 1.

4.1 Datasets

We construct two datasets for the two-stage training procedure, referred to as the **expert dataset** and the **simple dataset**. The overall data structure is illustrated in Fig. 3. We first generate the expert dataset and subsequently derive the simple dataset through additional post-processing.

Expert Dataset To obtain the expert dataset, we place a manipulator in the `pybullet` simulation environment and randomly generate spherical obstacles within its workspace. We then use `pybullet-planning` to compute collision-free joint-space trajectories, which serve as expert demonstrations for training. Formally, the training samples are expressed as

$$\mathbb{D}_{\text{expert}} = \{\, (\mathcal{P}_{\text{expert}}(t),\, \mathcal{O},\, q_{t+1})\, \}.$$

For each step t, we collect the trajectory prefix $\mathcal{P}_{\text{expert}}(t)$ up to step t, together with the environment description $\mathcal{O}$ in Cartesian space, as input. The corresponding expert sampling point q_{t+1} at step $t+1$ is used as the ground truth. For each type of manipulator, we collected approximately 40,000 data pairs across roughly 3,000 different scenarios.

Simple Dataset To construct the simple dataset, we further process the expert data by generating trajectories through linear interpolation from the start joint configuration $\mathbf{q}_s$ to the goal configuration $\mathbf{q}_g$. For a given pair $(\mathbf{q}_s, \mathbf{q}_g)$, the linear trajectory is defined as

$$\mathcal{P}_{\text{linear}} = \{\mathbf{q}'_t\}_{t=0}^{T-1},$$

where T is the number of interpolation points, and each point is given by

$$\mathbf{q}'_t = \left(1 - \frac{t}{T-1}\right)\mathbf{q}_s + \left(\frac{t}{T-1}\right)\mathbf{q}_g, \quad t \in \{0, \ldots, T-1\}.$$

We denote by $\mathcal{O}'$ the modified environment obtained by removing any obstacles that intersect the linear trajectory $\mathcal{P}_{\text{linear}}$. Based on these processed trajectories and environments, we construct the simple dataset:

$$\mathbb{D}_{\text{simple}} = \{\, (\mathcal{P}_{\text{linear}}(t),\, \mathcal{O}',\, q'_{t+1})\, \}.$$

For each episode in $\mathbb{D}_{\text{expert}}$, we constructed a basically similar scene $\mathcal{O}'$ and a linear trajectory $\mathcal{P}_{\text{linear}}$ in joint space in $\mathbb{D}_{\text{simple}}$.

4.2 Two-Stage Training

Since the high-dimensional sampling space is vast while the proportion of truly effective samples is relatively small, it is challenging for the model to converge when learning collision avoidance and optimal path generation simultaneously. Motivated by this, we introduce the following two-stage training method.

First Stage We train the model on the simple dataset in this first stage. Since linear interpolation produces the optimal path in the absence of obstacles, the model learns to rapidly converge toward the effective sampling region and to identify high-quality trajectories within that region. At the same time, because the environmental configuration is largely preserved, the model also acquires an initial understanding of the workspace through the high-dimensional environmental representation.

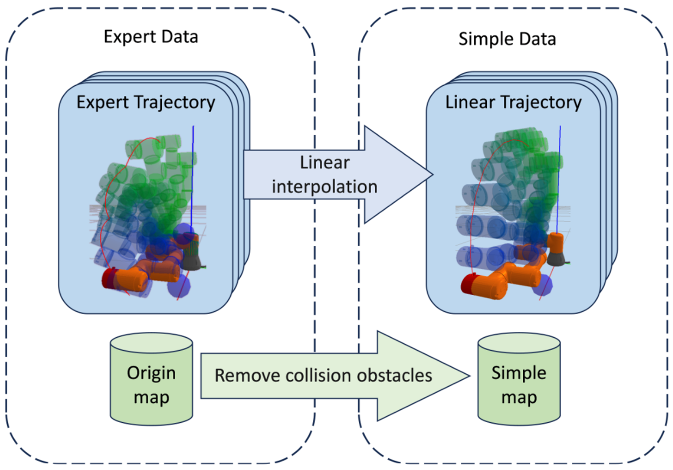

Fig. 3. Data composition. Simple data is used for the first stage of fast convergence learning, while expert data is used for the second stage of high-quality collision avoidance learning.

Second Stage In the second stage, we retrain the model using the original expert dataset, which contains complex environments and nontrivial expert trajectories. Having already learned fast convergence and near-optimal behavior in the first stage, the model now only needs to refine its ability to avoid collisions while still maintaining high-quality solutions.

In summary, the first stage teaches the model convergence behavior and near-optimal sampling patterns, while the second stage focuses on collision avoidance under more realistic and complex planning conditions. By decomposing the high-dimensional planning problem into these two learning phases, we ultimately obtain a heuristic sampler that efficiently explores the configuration space while avoiding collisions.

4.3 Loss Function

Regarding the design of the loss function, we use the Mean Squared Error (MSE) loss, which is the mean square error between the sampled points predicted by the model and the sampled points of the expert trajectory. The loss is given by

$$L_{MSE} = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{N}\left(q_j^i - \hat{q}_j^i\right)^2,$$

where B is the batch size, N is the dimension of the sampling space, q_j^i represents the j-th dimension value of the model output sampling point in the i-th batch and $\hat{q}_j^i$ represents the value of the expert trajectory.

5. EXPERIMENTAL EVALUATION

In this section, we evaluate our algorithm on 6-DOF and 7-DOF manipulators through extensive batch testing and compare its performance against traditional high-dimensional motion planning methods. We further deploy the proposed algorithm on a real UR3 manipulator to demonstrate its practical feasibility.

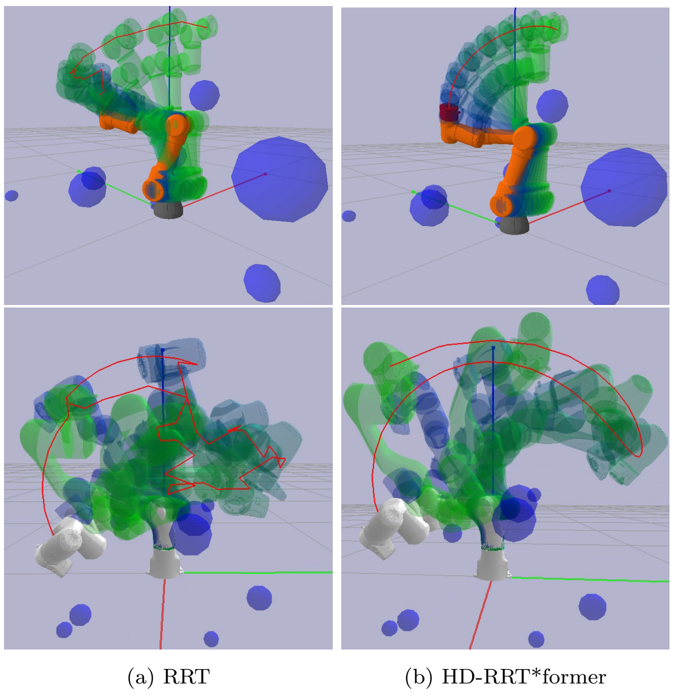

Fig. 4. Case Study. The two images on the left show the rugged trajectory planned by RRT in joint space; the two images on the right show the relatively smooth trajectory planned by our algorithm.

5.1 Simulation Settings

Our model is implemented in PyTorch and trained in two stages on a single GPU (RTX 5090). The first stage lasted approximately 50 epochs and took 3 hours, while the second stage lasted approximately 200 epochs and took approximately 20 hours. The batch size was set to 128, and the optimizer and other configurations were based on Adam's default parameters. The initial learning rate was set to 0.0001. Regarding model configuration, for 6D UR3 manipulator, the Transformer network had $d_{model} = 72$, $n_{head} = 6$, and a 6-layer encoder; for 7D Franka manipulator, the Transformer network had $d_{model} = 70$, $n_{head} = 7$, and a 6-layer encoder. The PI-KGAN parameter configuration matched that of the Transformer. Both models have a total of 0.63M around parameters.

We used the joint space of the manipulator as the sampling space and randomly generated 10 to 20 spherical obstacles of random size and position within its workspace, conducting a total of 500 successful episodes in scenarios completely different from the training set. For the perception of the 3D environment, we used an occupancy grid map. Since the workspace of the manipulator is spherical, we took the smallest cube that could encompass the workspace to represent the 3D environment, and divided it into $50 \times 50 \times 50$ occupancy grids, with each grid cell approximately $0.02m$ in size.

The algorithm's hyperparameters mainly include step size and alpha. In the joint space of the manipulator, step size represents the maximum allowable joint rotation when propagating or updating sampled configurations. Alpha denotes the ratio of uniformly sampled nodes relative to

Table 1. Statistics on the comparison of our algorithm with other algorithms.

Scenario	Method	Time(s)	Nodes	Path Cost	Joint Acceleration
6-DOF UR3	RRT	0.18±0.29	389.89±246.56	6.53±1.23	1.05±0.27
	RRT*	1.13±0.71	360.35±204.02	4.92±0.90	1.71±0.59
	RRTConnect	0.03±0.09	68.77±106.49	5.66±2.26	0.39±0.54
	HD-RRT*former	**0.01**±0.13	**27.74**±42.58	**4.40**±0.73	**0.05**±0.04
7-DOF Franka	RRT	0.09±0.11	308.05±137.19	9.76±1.96	1.76±0.46
	RRT*	1.28±0.81	324.67±183.17	7.08±1.42	2.93±0.84
	RRTConnect	**0.01**±0.03	47.04±50.17	6.64±2.16	0.33±0.50
	HD-RRT*former	0.03±0.16	**34.73**±20.67	**5.74**±1.24	**0.06**±0.06

Note: Results are presented as Mean ± Standard Deviation.

heuristically guided samples. In our experimental setup, the step size is 0.2, and alpha is 0.1.

5.2 Simulation Results

We compared our algorithm with several sampling-based algorithms, primarily analyzing its performance based on solution time, path quality, and path smoothness. A detailed description of each metric is as follows:

- **Time**: The mean and standard deviation of time for the algorithm to find a feasible solution.
- **Nodes**: The mean and standard deviation of the number of nodes the algorithm has expanded by the time it finds a feasible solution.
- **Path Cost**: The mean and standard deviation of the length of a feasible path in joint space. The distance is defined as the Euclidean distance between two consecutive samples at the same joint.
- **Joint Acceleration**: The mean and standard deviation value of all joint accelerations of the feasible solution.

The statistical metrics are shown in Table 1. The experimental results demonstrate that HD-RRT*former consistently outperforms classical sampling-based planners across almost all evaluated metrics.

The reduced planning time is a direct result of this heuristic sampling process. Although the model takes longer to perform one inference operation than to perform a random sampling operation, the actual average running time of HD-RRT*former is shorter than that of the random sampling algorithm due to the reduced number of iterations required. And the model predicts expansion directions that are more likely to be collision-free and goal-directed, so fewer invalid extensions and fewer collision checks are required. This also explains the substantially smaller number of generated nodes. By compressing the sampled space in the first training stage, HD-RRT*former builds a compact, efficient search structure rather than expanding a large, bushy tree across the full space.

In terms of path cost, our algorithm finds shorter paths compared to other sampling algorithms. This is because our algorithm utilizes historical sampling information to selectively sample states in the environment that are biased towards the destination, rather than selecting significantly off-target locations. Furthermore, due to the incorporation of environmental information, the algorithm has a global perceptual field when generating paths, avoiding the sampling of useless states.

The improvement in joint acceleration highlights the benefit of incorporating motion priors during training. Thanks to the introduction of the PI-KGAN module, the model learns from expert demonstrations and feasible reference trajectories that embody natural joint coupling and smooth kinematic behavior. Consequently, the generated trajectories avoid the oscillations common in sample-based planners, resulting in smoother motion that better aligns with physics-driven constraints.

In Figure 4, we compared the performance of our algorithm with that of RRT. The results clearly show that our algorithm plans a smoother path, as demonstrated in both 6D and 7D manipulator experiments. This is crucial for the execution of the underlying controller and the path optimization of the local planner.

5.3 Real-world experiment

To verify that our algorithm can be deployed in real-world scenarios, we conducted experiments on real manipulators. Detailed experimental screenshots are shown in Figure 5. We placed four balloons in pre-defined positions to simulate obstacles. We used ROS to read the current joint state of the manipulator as the starting joint state $\mathbf{q}_s$ and pre-defined a goal joint state as the algorithm's $\mathbf{q}_g$. Then, the proposed HD-RRT*former as implemented to plan a collision-free path in joint space. Finally, we published this trajectory to the real UR3 manipulator via ROS, and it was successfully executed. Videos are available at `https://youtu.be/ePD9mgIfMiY`.

6. CONCLUSION

We present a sampling-based motion planning algorithm for high-dimensional configuration spaces. The method introduces an approximate environmental representation that captures structural features of the space, coupled with a two-stage training strategy to address challenges of sampling inefficiency and slow convergence. Experimental results demonstrate that our approach achieves efficient and high-quality path planning in complex environments, outperforming traditional algorithms in manipulator joint space applications.

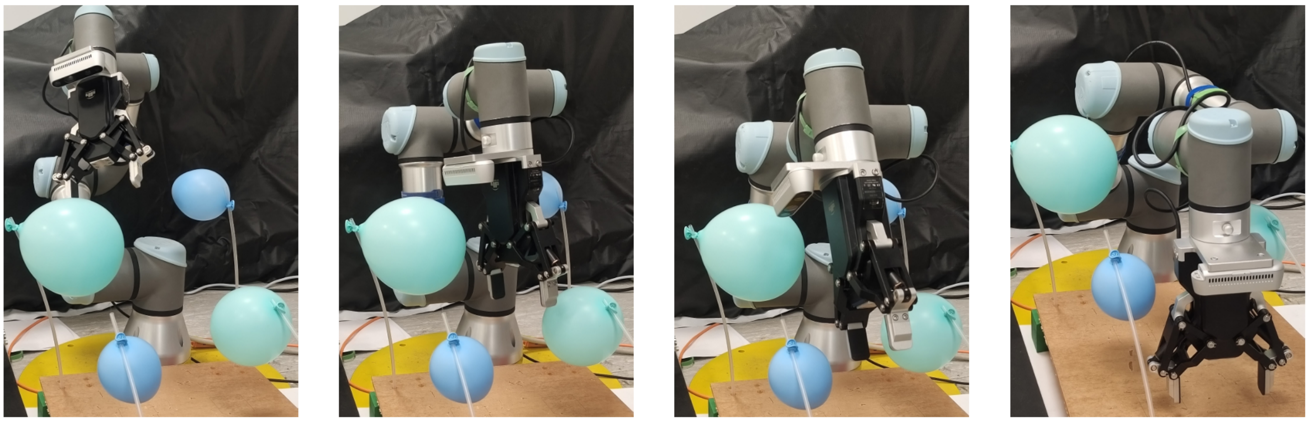

Fig. 5. Real-world scenario experiment. Four balloons represent spherical obstacles of different sizes. The UR3 manipulator needs to move from the starting state to the goal state without touching the obstacles.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work the authors used ChatGPT in order to polish. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

Akgun, B. and Stilman, M. (2011). Sampling heuristics for optimal motion planning in high dimensions. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2640–2645. IEEE.

Butler, C.L., Smith, R.D., and Alleyne, A.G. (2024). Sampling-based planning for guaranteed safe energy management of hybrid uav powertrain under complex, uncertain constraints. *IEEE Transactions on Control Systems Technology.*

Chaplot, D.S., Pathak, D., and Malik, J. (2021). Differentiable spatial planning using transformers. In *International conference on machine learning*, 1484–1495. PMLR.

Feng, M., Li, S., and Yin, X. (2025). Rrt* former: Environment-aware sampling-based motion planning using transformer. *arXiv preprint arXiv:2511.15414*.

Gammell, J.D., Srinivasa, S.S., and Barfoot, T.D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, 2997–3004. IEEE.

Gasparetto, A. and Zanotto, V. (2007). A new method for smooth trajectory planning of robot manipulators. *Mechanism and machine theory*, 42(4), 455–471.

Huang, Z., Chen, H., Pohovey, J., and Driggs-Campbell, K. (2024). Neural informed RRT*: Learning-based path planning with point cloud state representations under admissible ellipsoidal constraints. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 8742–8748. IEEE.

Huh, J., Lee, B., and Lee, D.D. (2018). Constrained sampling-based planning for grasping and manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 223–230. IEEE.

Johnson, J.J., Kalra, U.S., Bhatia, A., Li, L., Qureshi, A.H., and Yip, M.C. (2021). Motion planning transformers: A motion planning framework for mobile robots. *arXiv preprint arXiv:2106.02791.*

Johnson, J.J., Qureshi, A.H., and Yip, M.C. (2023). Learning sampling dictionaries for efficient and generalizable robot motion planning with transformers. *IEEE Robotics and Automation Letters*, 8(12), 7946–7953.

Kuffner, J.J. and LaValle, S.M. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, 995–1001. IEEE.

LaValle, S.M. (2006). *Planning algorithms.* Cambridge university press.

LaValle, S.M. and Kuffner Jr, J.J. (2001). Randomized kinodynamic planning. *The international journal of robotics research*, 20(5), 378–400.

Liu, W., Eltouny, K., Tian, S., Liang, X., and Zheng, M. (2024). Kg-planner: Knowledge-informed graph neural planning for collaborative manipulators. *IEEE Transactions on Automation Science and Engineering.*

Qureshi, A.H., Miao, Y., Simeonov, A., and Yip, M.C. (2020). Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1), 48–66.

Stentz, A. et al. (1995). The focussed dˆ* algorithm for real-time replanning. In *IJCAI*, volume 95, 1652–1659.

Tahir, Z., Qureshi, A.H., Ayaz, Y., and Nawaz, R. (2018). Potentially guided bidirectionalized rrt* for fast optimal path planning in cluttered environments. *Robotics and Autonomous Systems*, 108, 13–27.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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

Wang, J., Li, T., Li, B., and Meng, M.Q.H. (2022). GMR-RRT*: Sampling-based path planning using gaussian mixture regression. *IEEE Transactions on Intelligent Vehicles*, 7(3), 690–700.