Graph Wasserstein Autoencoder-Based Asymptotically Optimal Motion Planning With Kinematic Constraints for Robotic Manipulation

Graph Wasserstein Autoencoder Based Asymptotically Optimal Motion Planning With Kinematic Constraints for Robotic Manipulation

Chongkun Xia, Yunzhou Zhang, Sonya A. Coleman, *Member, IEEE,* Ching-Yen Weng, Houdei Liu, *Member, IEEE,* Shichang Liu, I-Ming Chen, *Fellow, IEEE,*

***Abstract*—This paper presents a learning based motion planning method for robotic manipulation, aiming to solve the asymptotically-optimal motion planning problem with nonlinear kinematics in a complex environment. The core of the proposed method is based on a novel neural network model, i.e., Graph Wasserstein Autoencoder (GraphWAE) network, which is used to represent the implicit sampling distributions of the configuration space (C-space) for sampling-based planning algorithms. Through learning the implicit distributions, we can guide the planning process to search or extend in the desired region to reduce the collision checks dramatically for fast and high-quality motion planning. The theoretical analysis and proofs are given to demonstrate the probabilistic completeness and asymptotic optimality of the proposed method. Numerical simulations and experiments are conducted to validate the effectiveness of the proposed method through comparisons with typical planning algorithms (e.g., RRT* and PRM*) in a series of problems from 2D, 6D and 12D robot C-spaces in the challenging scenes. In addition, results indicate that the proposed method can be generalised and achieves better planning performance.**

***Index Terms*—Motion planning, graph wasserstein autoencoder, kinematic constraints, collision detection, robotic manipulation**

I. Introduction

For most autonomous robot systems, motion planning is a very important module especially in a complex environment. Although different robots require different types of motion generation and planning methods, the nature of the motion planning problem has never changed in the past decades. Without loss of generality, the motion planning problem can be defined as the following: find a feasible motion sequence ζ in the configuration space that moves a robot or a manipulator from a start state q_{start} to a goal state q_{goal} without collisions in the obstacle environment. Furthermore, the optimality of motion planning is also a major concern for real-world robotic applications. In fact, optimal motion planning in C-space has been proven to be a PSPACE-hard problem, which is difficult to solve using *complete planners* [1]. Due to the low computational complexity and good practicability, sampling-based motion planners (SBMPs) have become popular solutions to optimal motion planning problems in various robotic applications, offering asymptotical optimality and probabilistic completeness guarantees [2]. The SBMPs usually attempt to search the whole configuration space to obtain all possible motion sequences, and then find a feasible solution by kinematic-based collision detection. However, when the environment is very complex such as a cluttered scene or narrow scene, the kinematic constrains of a robot will affect the planning efficiency of the SBMPs significantly. For example, the two most widely used planners in the field of motion planning research, the RRT* [3] and PRM* [4] both require many collision checks to obtain a feasible collision-free motion sequence due to kinematic constrains in the cluttered scene, which typical result in a time-consuming process. Sánchez *et al* indicated that the collision detection required more than 90% of the total time consumption for the SBMPs [5]. The aforementioned facts indicate that collision detection has become the main execution impediment to the planning efficiency of the SBMPs.

Many approaches to reduce collision checks have been proposed to improve the planning procedure for the SBMPs. Hauser [6] presented a lazy strategy to reduce the fraction of edges that underwent collision checking and demonstrated that the approach required less time than previous approaches. While the delayed checking mechanism retried the tree's expansion whenever finding that a collision was detected, the planning performance might degrade with increased probability especially when the number of collision checks was relatively large. Van den Berg *et al.* [7] proposed a watershed labeling method to effectively guide the sampling process towards the interesting region of the scene. Gammell *et al.* [8] performed an ordered search by using batches of samples to accelerate the planning process and converge faster towards the global optimum. While these heuristics-based sampling methods usually perform well in specific domains or problems, it is very hard to determine how they perform on new unseen scenes that are beyond the expected conditions. In recent years, the learning-based approaches have become increasingly popular solutions through learning the non-uniform sampling distributions to dramatically reduce the collision checks for fast and high-quality motion planning. Huh *et al.* in [9]

Manuscript received December, 2020; revised XX, 2021. This work was supported by National Natural Science Foundation of China (No. 61973066, 61803221, U1813216), Equipment Pre-research Foundation(61403120111), Fundamental Research Funds for the Central Universities, China (N172608005), Guangdong Young Talent with Scientific and Technological Innovation (2019TQ05Z111). (Corresponding author: Yunzhou Zhang & Houdei Liu)

C. Xia and H. Liu are with Centre for Artifical Intelligence and Robotics, Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China. {xia.ck; liu.hd}@sz.tsinghua.edu.cn

Y. Zhang is with College of Information Science and Engineering, Northeastern University, Shenyang 110819, China. zhangyunzhou@mail.neu.edu.cn

Sonya A. Coleman is with School of Computing and Intelligent Systems, Derry, Ulster University, UK. sa.coleman@ulster.ac.uk

Shichang Liu is with SIASUN Robot Automation CO., Ltd, Shenyang 110819, China. liushichang@siasun.com

C. Weng and I.-M. Chen are with Robotics Research Centre, Nanyang Technological University, Singapore 639798, Singapore. {WENG0025; michen}@ntu.edu.sg

and [10] proposed a non-uniform sampling strategy based on Gaussian Mixture Models (GMMs) to accelerate the collision detection process, and applied their method in a Rapidly-exploring Random Trees (RRT) planner. However, the GMMs-based method has a rigorous hypotheses, local minimum and difficulty in selecting self-parameters, which has the potential to significantly increase the uncertainties and complexities of the whole planning process. Ichter *et al.* [11] adopted the conditional variational autoencoder (CVAE) to learn the implicit representation of the configuration space for bias sampling. However, the CVAE-based method lacks sufficient representation ability of high-dimensional C-space, especially for complex scenarios. Moreover, the samples generated by the two methods may be ambiguous and prone to resulting in many undesirable checks, which results in the performance degradation of the planning process for the SBMPs.

In a typical planning scenario, a so-called point or configuration q of the planning process represents a certain state that belongs to either collision-free configuration space or not, dividing the C-space χ into two regions, a collision-free region χ_{free} and a collision region χ_{col}. The purpose of collision detection operation is to judge whether all motion configurations belong to the collision-free region χ_{free} of C-space. In essence, the learning-based approaches are to train an implicit representation to further build a generative model of the collision-free region χ_{free} in C-space, and thereby avoid the collision checks as much as possible in the sampling-based planning process. Inspired by the graph neural network [12] [13], we propose a novel graph wasserstein autoencoder (GraphWAE)-based asymptotically optimal motion planning method to solve the optimal motion planning problem. In this work, the sampling distribution in C-space will be transformed into an undirected graph model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is used as the input of the developed wasserstein autoencoder. The configuration samples generated from demonstrations of successful plans and previous experience are used as the vertices that form the vertex set $\mathcal{V}$ of the graph. The potential movements of samples are used as edges that form the edge set $\mathcal{E}$. Through learning the latent model of this distribution, our approach can be used as the sample generative source to deterministically generate samples that belong to the collision-free region, instead of a random generative source. Furthermore, the learning-based generative model promotes the sampling procedure of the planning process occurred in the collision-free region of the configuration space, and further effectively reduces and simplifies the collision checks, which significantly improves the planning performance of the SBMPs.

Contributions: The contributions of this work are summarized as follows: (1) We propose a GraphWAE-based motion planning method to address the asymptotically optimal planning problem with kinematic constrains in the complicated environment; (2) Our approach mainly focuses on the early sampling stage of the planning process without modifying the internal structures of the planners. Thus, the proposed methodology is general and easy to be deployed into other sampling-based motion planners; (3) The learning-based graph generative model can effectively represent the underlying distribution of the C-space in a challenging environment, resulting in a significant improvement in terms of planning time, success rate and path quality.

II. Notation and Problem Formulation

A. Notation

Let $\chi \subseteq \mathbb{R}^d$ denote the d-dimensional robot configuration space, where $d \geq 2$. To keep the symbols consistent, we use χ_{free} to denote the collision-free region and χ_{col} to denote the collision region in C-space. χ_{free} and χ_{col} both are the open subsets of χ, and $\chi_{free} \bigcup \chi_{col} = \chi$, $\chi_{free} \bigcap \chi_{col} = \varnothing$. The initial state is $x_{init} \in \chi_{free}$, and the goal region is $\chi_{goal} \subset \chi_{free}$. The control input sequence is denoted as $u(\tau) : [0, s] \longmapsto \mathcal{U}$, which is used to drive a robot or a manipulator from x_{init} to χ_{goal}. The system dynamics of a robot or a manipulator can be briefly described below.

$$\dot{\mathbf{x}} = \Xi(\mathbf{x}, \mathbf{u}) \tag{1}$$

where $\mathbf{x} \in \chi$ denotes the state vector and $\mathbf{u} \in \mathcal{U}$ denotes the piecewise control vector. Ξ denotes the continuous-time system dynamics.

Usually, the basic motion planning problem is briefly defined by a triplet $(x_{init}, \chi_{goal}, \chi_{free})$. Assuming that the function Ξ is smooth with respect to the state vector $\mathbf{x}$ and the control vector $\mathbf{u}$, the definition of a feasible path for the basic motion planning problem is shown as follows. A visualisation of the 2D motion planning problem is depicted in Fig. 1.

Definition 1. *(Feasible path) Defining a function $\zeta(\tau) : [0, s] \to \chi$, if it is continuous and has bounded variation, then we call the function ζ a motion path. If $\forall \tau \in [0, s]$, $\zeta(\tau) \subseteq \chi_{free}$, then $\zeta(\tau)$ is called a collision-free path. Furthermore, if $\zeta(\tau)$ is a collision-free path, and $\zeta(0) = x_{init}, \zeta(s) \in cl(\chi_{goal})$, then $\zeta(\tau)$ is so called a feasible path (or valid path) for the motion planning problem $(x_{init}, \chi_{goal}, \chi_{free})$.*

Let $c(\zeta) : \mathbb{R}^d \longmapsto \mathbb{R}_{\geqslant 0}$ denote the cost function of the motion planning problem as follows:

$$c(\zeta) = \int_0^s [\zeta(\tau)^{\mathrm{T}} A \zeta(\tau) + u(\tau)^T B u(\tau)] dt \tag{2}$$

where both A and B are positive matrices. It should be noted that if $\zeta(\tau)^{\mathrm{T}} A \zeta(\tau) + u(\tau)^{\mathrm{T}} B u(\tau) = 1$, the cost function denotes the Euclidean distance. In this paper, we select the path length as the cost function, i.e., $c((\zeta)|\zeta \in [0, s]) = \int_0^s dt$.

B. Problem formulation

Kinematic constrains usually exist in the real-world robot applications. We call the motion planning problem with kinematic constraints as the feasible planning problem to distinguish them from the basic motion planning problem. The rigorous definitions about the feasible planning problem considered in this paper are formally described as follows.

Problem 1. *(Feasible planning problem) Given a basic motion planning problem $(x_{init}, \chi_{goal}, \chi_{free})$, find a feasible (or valid) path $\zeta(\tau) : [0, s] \mapsto \chi_{free}$ such that $\zeta(0) = x_{init}$ and $\zeta(s) \in \chi_{goal}$. If such a path exists, then output the result; otherwise, report failure.*

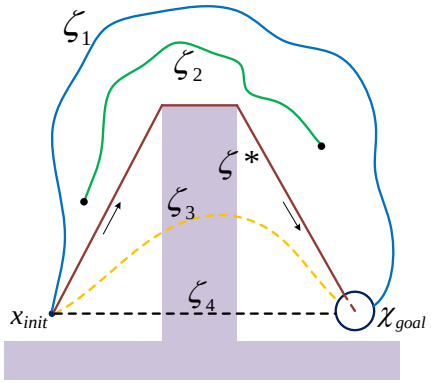

Fig. 1. The diagram of 2D motion planning problem. Due to the kinematic constraints, the robot cannot directly move along the straight line from the start state x_{init} to the goal region χ_{goal}. ζ_1 denotes the feasible (or valid) path. $\zeta_2, \zeta_3, \zeta_4$ all denote the invalid paths. The start state and goal state of ζ_2 do not meet the requirement of motion planning problem. ζ_3 and ζ_4 do not consider the kinematic constraints. ζ^* denotes the desired optimal motion path.

Supposing the valid path set from solving Problem 1 is Θ, we can set $\zeta^* \in \Theta$ as the optimal path. The corresponding optimal planning problem is defined below. It is should be noted that the optimal path considered in this paper is a theoretical optimal solution for the motion planning problem.

Problem 2. *(Optimal planning problem) Given a motion planning problem* $(x_{init}, \chi_{goal}, \chi_{free})$ *and a length cost function* c*, find a feasible motion sequence or path* $\zeta^* \in \Theta$ *such that* $c(\zeta^*) = min\ \{c(\zeta) : \zeta \in \Theta$ *is a feasible path*$\}$*. If one exists, this path* ζ^* *is called an optimal path. If no such path exists, then report failure.*

In real robot applications, the optimal planning problem is hard to solve directly by complete planners. Current planning solutions focus on the sampling-based planning methods (i.e., incomplete planners), which possess asymptotically optimality. Then the asymptotically optimal motion planning problem is defined as follows.

Problem 3. *(Asymptotically optimal planning problem) Given Problem 2 and the cost function* $c(\zeta)$*, find a solution* ζ_a *such that* $c(\zeta_a) \geqslant (\varepsilon + 1)$ *for any small* $\varepsilon > 0$*, when the number of samples* $N \to \infty$*.*

III. Methodology

The purpose of this work is to develop a novel learning-based method capable of approximating the distribution χ_{free} of the collision-free region containing optimal motion sequences, and further generate more similar samples to reduce or avoid collision checks for improving the performance of sampling-based motion planners such as planning time, path quality and successful rate. Since the collision-free region in C-space usually is high-dimensional, unknown and complex, it is very difficult to accurately represent or approximate the corresponding sampling distribution χ_{free}. The state-of-the-art representation learning methods mainly contain two classes: the Gaussian Mixture Models (GMMs) [9] [10] and the variational autoencoders (VAEs) [11]. The GMMs have demonstrated good performance in real robot planning problems, but the appropriate parameters are very difficult to select. Moreover, the VAEs often generate blurry samples when applied to the practical engineering projects despite being quite nifty and elegant in theory. Wasserstein Autoencoder (WAE) [14] is a new regularized autoencoder neural network based on the optimal transport (OT) theory [15], which has a strong representation ability for the Euclidean data such as text, image and voice. The WAE not only characterizes the non-Euclidean data but also captures the related structure accurately.

For asymptotically optimal planning problems, the corresponding sampling distributions usually are non-Euclidean data that are difficult to directly represent by the WAE. Inspired by graph neural networks, we propose a new methodology based on a Graph Wasserstein Autoencoder (GraphWAE) to learn the bias sampling distributions for motion planning, as shown in Fig. 2. Essentially, the GraphWAE is an extension and improvement of the standard Aasserstein Autoencoder (WAE), which lays more emphasis on the integration of the graph model and autoencoder network. This improvement provides a trained graph generative model that approximates the sampling distributions of the collision-free region as the sample generation source of the SMBPs. Furthermore, our method also leverages previous successful plans or demonstrations to guide the SBMP to search trees or road maps in the collision-free region of the configuration space. The corresponding pseudo code of the proposed methodology is depicted in Algorithm 1.

Algorithm 1 Learning bias sampling distributions methodology

Offline:

1: Input: Training data (from previous successful plans, motion trajectories in action, human demonstration, etc.)
2: Train the encoder network and the decoder network of GraphWAE, as shown in Algorithm 2.

Online:

3: Input: The motion planning problem $(\chi_{free}, x_{start}, x_{goal})$, the trained graph neural network model.
4: Use the learned GraphWAE to obtain configuration samples as the sample source.
5: Run the sampling-based planners (e.g., PRM*, FMT, RRT*) without collision detection, and generate the candidate motion path.
6: Check the collisions of the candidate path. Output the final result if the path is qualified; otherwise, return to Step 4.

A. Basic architecture of Graph Wasserstein Autoencoder

The GraphWAE proposed in this paper is a new, novel, variant of the regularized autoencoders, which has fully absorbed the quintessence theory of graph neural networks and the variational auto-encoder network. The structure diagram of the proposed GraphWAE network is presented in Fig. 3. From Fig. 3, we can see that the GraphWAE is divided into four parts: graph representation, encoder, decoder and graph

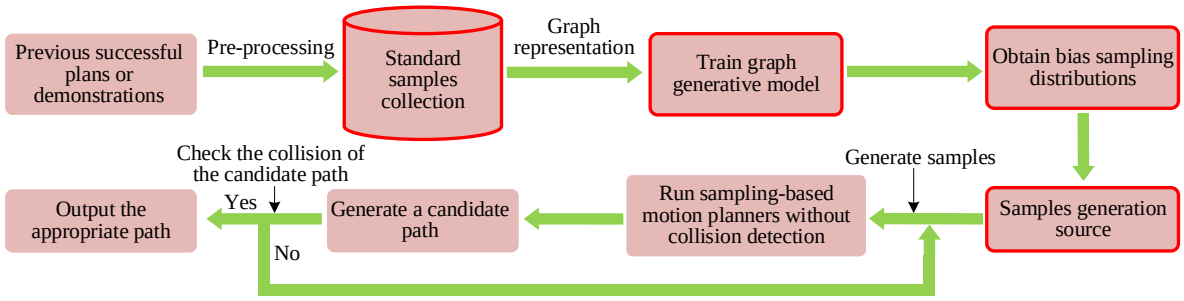

Fig. 2. The system diagram of the proposed methodology. It should be noted that our main contributions in this paper are highlighted in red box.

generation. The graph representation of the planning problem is an important early step of the proposed GraphWAE for learning the bias sampling distributions in the C-space. In this paper, we construct the undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to represent the graph model of the whole configuration space, where $\mathcal{N}$ denotes the configuration set, and $\mathcal{E}$ denotes the edge set among different configurations. It should be noted that the configuration actually is the node in the undirected graph. Since the sampling distributions are the subsets of C-space, the graph model of the sampling distribution χ_{free} can be defined as $\mathcal{G}_{free} = (\mathcal{N}_{free}, \mathcal{E}_{free})$. In the same way, the graph model of the collision region can be defined as $\mathcal{G}_{col} = (\mathcal{N}_{col}, \mathcal{E}_{col})$. When the sampling procedure tends to infinity, $\mathcal{G}_{free} \bigcup \mathcal{G}_{col} = \mathcal{G}$ and $\mathcal{G}_{free} \bigcap \mathcal{G}_{col} = \varnothing$. Obviously, Fig. 3 presents the bias graph representation $\mathcal{G}_{free}$ of the sampling distribution χ_{free} in the motion planning problem.

For the proposed methodology, we use $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{E}, \boldsymbol{F})$ as the input to the encoder network, where $\boldsymbol{A}$ is the adjacency matrix of graph model $\mathcal{G}$, $\boldsymbol{E}$ is the edge attribute tensor and $\boldsymbol{F}$ is the node attribute matrix. When the graph representation is $\mathcal{G}_{free}$, we use $\boldsymbol{G}_{free}$ as the input of the GraphWAE, and the converse is $\boldsymbol{G}_{col}$. It should be noted that we use the $\boldsymbol{G}$ to denote the input for the following theoretical illustration without differentiating between $\boldsymbol{G}_{free}$ and $\boldsymbol{G}_{col}$ in this section. Similar to the WAEs, the GraphWAE can generate the graph model which is similar to the input graph by training a latent variable model $P_{\digamma}$. In fact, the latent variable model is progressed through two stages, that is, sampling a low dimensional code vector $\boldsymbol{Z}$ from a prior distribution $P_{\boldsymbol{Z}}$ defined on the latent space $\mathcal{Z}$ and mapping it to a graph model $\boldsymbol{G} \in \mathcal{G}$ via a conditional distribution $P_{\digamma}(\boldsymbol{X}|\boldsymbol{Z})$. For simplicity, we use the distribution $P_{\digamma}(\boldsymbol{X}|\boldsymbol{Z})$ as the generative distribution (i.e., the decoder), deterministically mapping $\boldsymbol{Z}$ to $\boldsymbol{G} = \digamma(\boldsymbol{Z})$ for a given map function $\digamma : \boldsymbol{Z} \longmapsto \boldsymbol{G}$. Additionally, the GraphWAE will minimize the optimal transport distance between the latent variable model $P_{\digamma}$ and the unknown graph data distribution $P_{\boldsymbol{G}}$. Supposing the cost function between two data distribution is $c(g, g')$, we adopt the following function as the objective of the GraphWAE:

$$\min_{Q(\boldsymbol{Z}|\boldsymbol{X})} \mathop{\mathbb{E}}_{P_{\boldsymbol{G}}} \mathop{\mathbb{E}}_{Q(\boldsymbol{Z}|\boldsymbol{X})} [c(\boldsymbol{G}, \digamma(\boldsymbol{Z}))] + \gamma \mathcal{D}_{\boldsymbol{Z}}(Q_{\boldsymbol{Z}}, P_{\boldsymbol{Z}}) \tag{3}$$

where $Q(\boldsymbol{Z}|\boldsymbol{X})$ denotes the conditional distribution, which is known as the encoder, $Q_{\boldsymbol{Z}} = Q_{\boldsymbol{Z}}(\boldsymbol{Z}) = \int Q(\boldsymbol{Z}|\boldsymbol{X}) P_{\boldsymbol{G}}(\boldsymbol{G}) d\boldsymbol{G}$ is the posterior distribution, $\mathcal{D}_{\boldsymbol{Z}}(\bullet)$ denotes the divergence measure function between two distributions on the latent space $\mathcal{Z}$ and $\gamma > 0$ is a regularization parameter. Like the auto-encoders, the GraphWAE is also encouraged to learn a high-level implicit representation of the input rather than simply copy the given input. Thus, the dimension of the vector $\boldsymbol{Z}$ is usually fairly smaller than the size of the input. Besides, it should be noted that both of the distributions $Q(\boldsymbol{Z}|\boldsymbol{G})$ and $\digamma(\boldsymbol{Z})$ need to be parameterized by deep networks. In fact, we usually use the back propagation algorithm as the stochastic gradient descent module in deep nets to optimize the above objective function.

From the equation (3), we can find that the objective of the proposed GraphWAE contains two terms: $c(\boldsymbol{G}, \digamma(\boldsymbol{Z}))$ and $\gamma \mathcal{D}_{\boldsymbol{Z}}(Q_{\boldsymbol{Z}}, P_{\boldsymbol{Z}})$. The first reconstruction term $c(\boldsymbol{G}, \digamma(\boldsymbol{Z}))$ connects the encoder with the decoder so that the encoded graph can be reconstructed accurately by the decoder network. The second regularization term $\gamma \mathcal{D}_{\boldsymbol{Z}}(Q_{\boldsymbol{Z}}, P_{\boldsymbol{Z}})$ utilizes the aggregated posterior $Q_{\boldsymbol{Z}}$ to match the distribution $P_{\boldsymbol{Z}}$, which can effectively control the size of the whole encoded data. In this paper, we adopt the proposed GraphWAE to learn the bias sampling distribution, aiming to reduce the collision checks and further improve the planning performance for sampling-based planners.

B. Learning bias sampling distribution using GraphWAE

The core step of the proposed methodology is to train an appropriate graph generative model to represent the sampling distributions. More specifically, we will fully utilize the previous planning data to train the encoder network and decoder network of the GraphWAE for obtaining the implicit representation of sampling distributions, and further adopt the learned generative model as the sample source of sampling-based planners. Firstly, high-quality training data acquisition is very critical for the whole training process of the GraphWAE. When a planning task is executed, the collision detection mechanism usually generates huge quantities of extra samples that have been identified to belong to a certain sampling distribution χ_{free} or χ_{col} for sampling-based planners. Thus we can collect and preprocess these checked samples as the training data. In fact, the sources of the previous planning data are exceptionally diverse such as successful plans, human demonstration, etc.

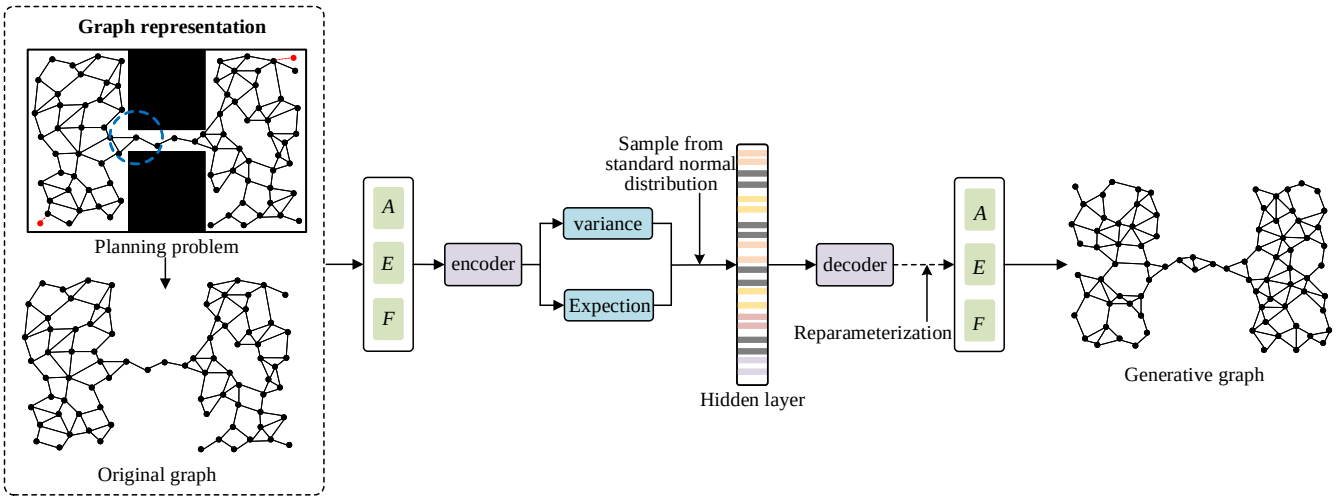

Fig. 3. The setup of graph wasserstein autoencoder (GraphWAE). The encoder maps the bias sampling distribution to the latent space, and then the decoder utilizes the latent samples to reconstruct the distribution information. Besides, the trained encoder can be used as the random sampling source of the configurations for the SBMPs.

Another important consideration is the penalty design of the objective function (i.e., equation (3)), which plays an important role in the training process of the GraphWAE. According to [16], the maximum mean discrepancy (MMD) has good performance on the similarity judgement between two distributions, especially for the high-dimensional standard normal distributions. Considering that the prior distribution $P_{\mathbf{Z}}$ of latent space $\mathcal{Z}$ belongs to a Gaussian distribution, we select the MMD as the penalty of the objective function $\mathcal{D}_{\mathbf{Z}}(Q_{\mathbf{Z}}, P_{\mathbf{Z}})$. The penalty based on maximum mean discrepancy is designed as follows:

$$\mathcal{D}_{\mathbf{Z}}(Q_{\mathbf{Z}}, P_{\mathbf{Z}}) = \left\| \int_{\mathcal{Z}} k(z, \cdot) dP_{\mathbf{Z}}(z) - \int_{\mathcal{Z}} k(z, \cdot) dQ_{\mathbf{Z}}(z) \right\|_{\mathcal{H}_k} \tag{4}$$

where $\mathcal{H}_k$ denotes the reproducing kernel Hilbert space of the generalized real valued functions mapping $\mathcal{Z}$ to $\mathcal{R}$. $k : \mathcal{Z} \times \mathcal{Z} \mapsto \mathcal{R}$ is a positive-definite reproducing kernel. The training details of the GraphWAE is described in Algorithm 2. Additionally, it should be noted that the encoder $Q_\psi(\mathbf{Z}|x)$ in Algorithm 2 is non-random. Usually, deterministically maps these input data to the latent codes. Based on this condition, we can set $Q_\psi(\mathbf{Z}|x) = \delta_{\mu_\psi}$ for a function $\mu_\psi : \mathcal{G} \mapsto \mathcal{Z}$. Thus we just need to return $\mu_\psi(x_i)$ so that the $\hat{z}_i$ can be sampled from the distribution $Q_\psi(\mathbf{Z}|x_i)$.

Algorithm 2 The training procedure based on GraphWAE

1: **Initialization:** the network parameters of encoder Q_ψ and decoder F_ϕ; the latent discriminator D_τ; the regularization coefficient $\gamma > 0$; the characteristic positive-definite kernel k.
2: **Train:** the encoder network ψ and decoder network ϕ.
3: **while** $\{\psi, \phi\}$ are not convergent **do**
4: Input the training data $\boldsymbol{G} = \{g_1, g_2, \cdots, g_n\}$;
5: Do the sampling procedure from the prior distribution $P_{\mathbf{Z}}$ and acquire the samples $\{z_1, z_2, \cdots, z_n\}$;
6: **for** $i = 1 \to n$ **do**
7: Do the sampling procedure from the $Q_\psi(\mathbf{Z}|g_i)$ and acquire $\hat{z}_i$;
8: Save $\hat{z}_i$ to the set $\hat{\mathbf{Z}}$;
9: **end for**
10: Update Q_ψ and F_ϕ by ascending:
11:

$$\begin{aligned} &\frac{1}{n}\sum_{i=1}^{n} c(g_i, F_\phi(\hat{z}_i)) + \frac{1}{n(n-1)}\sum_{l\neq j} k(z_i, z_j) \\ &+ \frac{\gamma}{n(n-1)}\sum_{l\neq j} k(\hat{z}_l, \hat{z}_j) - \frac{2\gamma}{n^2}\sum_{l,j} k(z_l, \hat{z}_j) \end{aligned} \tag{5}$$

12: **end while**

C. Asymptotically Optimal Motion Planning based on GraphWAE

When the training process ends, we can obtain the graph generative model (i.e., the decoder F_ϕ) that can implicitly represent the sampling distribution χ_{free} or χ_{col} in C-space. After training, the GraphWAE can be allowed to generate samples from F_ϕ via the simple sampling operation from the normal distribution of the implicit variable. The online planning task begins with a planning problem with kinematic constrains defined by the triplet $(x_{init}, \chi_{goal}, \chi_{free})$. When the planning task begins, we use the standard normal distribution as the input of the learned decoder F_ϕ and obtain the output graph model. Next, we can obtain the configuration samples as the sample source of the motion planner through random sampling from the output reconstruction graph. Taking a typical 2D planning problem as an example, the planning results based on RRT* and GraphWAE-RRT* at different iteration steps are shown in Fig. 4. Fig. 4 indicates that the GraphWAE-RRT* can generate a higher quality motion path than the standard RRT* with shorter iterations. Moreover, the GraphWAE-RRT* needs fewer collision checks to determine

whether the candidate path is a feasible path or not, which dramatically reduces the planning time. The bias samples from collision-free sampling distributions can be directly used as the tree nodes to extend the search trees without checking collisions. Since the distribution of the output graph model can effectively represent the bias sampling distributions, the collision check within each iteration is not necessary for sampling-based motion planners. In this case, we are facing two options: (1) check the collision every few iterations as the tree expands; (2) check the collision of the final candidate path as the iteration procedure ends. In fact, the two opinions both effectively reduce the planning time and improve the planning quality. Due to the strong representation ability of the proposed GraphWAE, in this paper we select the second option as the collision detection strategy.

IV. Theoretical analysis

A. Probabilistic completeness

In this section, the probabilistic completeness of our approach is considered. Since our approach is general for the SBMPs, we will take the GraphWAE-based RRT* as an example to show the proof process. Firstly, the definition of a robustly feasible planning problem is shown below.

Supposing $\delta > 0$, if the hypersphere with a radius δ centred at the state $x \in \chi_{free}$ is entirely inside χ_{free}, the state x usually is said to be a $\delta - interior\ state$ of χ_{free}. We use the symbol $\text{int}^{\delta}_{\chi_{free}}$ to denote the collection of all $\delta-interior$ of χ_{free} states, i.e., $\text{int}^{\delta}_{\chi_{free}} := \{x \in \chi_{free} | \mathscr{R}_{x,\delta} \subseteq \chi_{free}\}\}$.

Definition 2. *(Robustly feasible planning problem) If a feasible path $\zeta : [0, s] \mapsto \chi_{free}$ lies completely inside the $\delta-interior$ of χ_{free}, i.e., $\zeta(\tau) \in \text{int}^{\delta}_{\chi_{free}}$ for all $\delta \in [0, s]$, we consider the path ζ to have the $strong\delta-clearance$. Furthermore, if there exists a path ζ_r with the $strong\delta - clearance$ for $\delta > 0$ and the path $zeta_r$ is a feasible solution on the planning problem $(x_{init}, \chi_{goal}, \chi_{free})$, then the planning problem $(x_{init}, \chi_{goal}, \chi_{free})$ is said to be robustly feasible.*

Then the definition of probabilistic completeness is as follows. It is should be noted that the symbol $V_n^{\mathscr{A}}$ denotes the set of vertices in the tree graph returned by the planning algorithm $\mathscr{A}$ and n is the number of samples.

Definition 3. *(Probabilistic completeness) For any robustly feasible motion planning problem $(x_{init}, \chi_{goal}, \chi_{free})$, a planning algorithm $\mathscr{A}$ is probabilistically complete only when the following condition is satisfied:*

$$\lim_{n\to\infty} \inf \mathbb{P}(\{\exists \chi_{goal} \in V_n^{\mathscr{A}} \bigcap \chi_{goal} | \zeta_r(0) = x_{init}, \zeta_r(s) \in \chi_{goal}\}) = 1 \tag{6}$$

Next, the theorem and proof of probabilistic completeness of our proposed approach (i.e., GraphWAE-RRT*) are shown as follows.

Theorem 1. *(Probabilistic completeness of GraphWAE-RRT*) The GraphWAE-RRT* is probabilistically complete. For any robustly feasible path planning problem $(x_{init}, \chi_{goal}, \chi_{free})$, there exists a constant $a > 0$ and $n_0 \in \mathbb{N}$, both dependent only on χ_{free} and χ_{goal} such that*

$$\mathbb{P}(\{V_n^{GraphWAE-RRT*} \bigcap \chi_{goal} \neq \varnothing\}) > 1 - e^{-an}, \forall n > n_0 \tag{7}$$

Proof. According to the detailed description in Section III, the GraphWAE is used as the non-uniform sample source of RRT* and does not interfere within the internal operation (eg., tree search) of the RRT*. Since GraphWAE theoretically and perfectly represents collision-free space with the training time is infinite, the samples generated by GraphWAE-RRT* are almost the same as those generated randomly. Considering the number of random samples $n \longrightarrow \infty$, RRT * can explore all states of the configuration space and probabilistic completeness [1]. Thus, we can infer that $V_n^{GraphWAE-RRT*}(\omega) = V_n^{RRT*}(\omega)$ for all $\omega \in \chi$ and $n \in \mathbb{N}$. The result of Graph-RRT* follows directly from the probabilistic completeness of RRT*. □

B. Asymptotic Optimality

In this section, the optimality of the planning algorithm is considered. The asymptotic optimality refers to the ability to return planning solutions whose costs converge to the global optimum, as shown in Definition 4.

Definition 4. *(Asymptotic optimality) Let $c*$ denote the cost of an optimal motion path and let $Y_n^{\mathscr{A}}$ denote the extension random variable related with the cost of the minimum-cost solution existing in the tree graph by a sampling-based planner at the end of iteration n. For any motion planning problem $(x_{init}, \chi_{goal}, \chi_{free})$ and the corresponding cost function $c(\zeta) : \mathbb{R}^d \longmapsto \mathbb{R}_{\geqslant 0}$ there exists a robustly optimal solution with a cost $c*$, the algorithm $\mathscr{A}$ has asymptotic optimality only when the following condition is satisfied:*

$$\mathbb{P}(\lim_{n\to+\infty} \sup Y_n^{\mathscr{A}} = c*) = 1 \tag{8}$$

It should be noted that the asymptotic optimality of a planning algorithm suggests that the limit $\lim_{n\to+\infty} Y_n^{\mathscr{A}}$ exists and $\lim_{n\to+\infty} Y_n^{\mathscr{A}} = c*$ due to $\forall n \in \mathbb{N}, Y_n^{\mathscr{A}} \geqslant c*$. It also indicates that the probabilistic completeness is necessary for asymptotic optimality. We use d to denote the dimensionality of the configuration space and use $\mathscr{L}(\chi_{free})$ to denote the Lebesgue measure of collision-free space. Moreover, we utilize $\mathscr{V}_d$ to denote the volume of the unit hypersphere in the d-dimensional Euclidean space. It should be noted that $\mu_{GraphWAE-RRT*}$ is a characterization constant of the connection radius r_n of GraphWAE-RRT* and $r_n = \mu_{GraphWAE-RRT*}(\frac{logn}{n})^{\frac{1}{d}}$. The asymptotic optimality of GraphWAE-RRT* is defined as follows.

Theorem 2. *(Asymptotic optimality of GraphWAE-RRT*) The GraphWAE-RRT* is asymptotically optimal only when the following condition is satisfied:*

$$\mu_{GraphWAE-RRT*} > (2(1 + \frac{1}{d}))^{\frac{1}{d}} (\frac{\mathscr{L}(\chi_{free})}{\mathscr{V}_d})^{\frac{1}{d}} \tag{9}$$

Proof. The GraphWAE-RRT* replaces the random generation mechanism with the GraphWAE as the sample generation

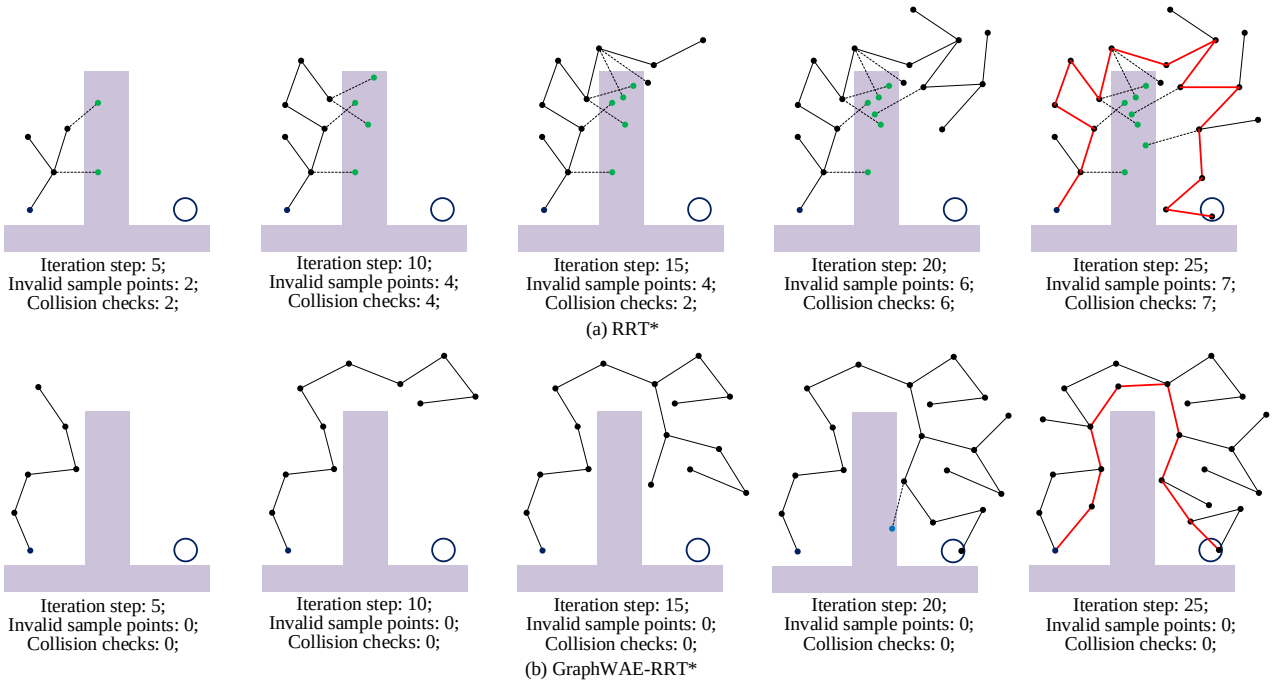

Fig. 4. Comparison of planning results at different iteration steps using RRT* and GraphWAE-RRT* for 2D planning problem. The green solid points denote the invalid sample points in the obstacle regions. The dotted lines denote the removed tree connects after collision detection.

source, which does not interfere with the tree expand and steering process of RRT*. Thus, the asymptotic optimality of GraphWAE-RRT* is the same as one of the RRT*. The detailed proof process is shown in [1][3]. Note that, the important premise of asymptotic optimality for GraphWAE-RRT* is that samples generated from the GraphWAE can occupy all of collision-free configuration space as the number n of samples tends to be infinite (i.e., $n \to +\infty$). □

V. Experiments and Discussion

To verify the proposed methodology and evaluate its performance, we design and implement several numerical simulations and experiments. We begin with a typical geometric planning problem in which we show the generality of our proposed method in different sampling-based planners and demonstrate its outstanding performance. Then we use two high-dimension robot motion planning problems to further explain the benefits of learning bias sampling distributions. The details of experimental deployment, result comparison and analysis are shown below.

A. Geometric planning problem

For most sampling-based planning algorithms, the geometric planning problem is very classical and plays an important role on the evaluation of planning performance. In order to make the results target-oriented, we design two highly represented geometric planning subproblems: (a) planning in a narrow passage (as shown in Fig. 5(a)); (b) planning in a complex scene (as shown in Fig. 5(b)). To verify the effectiveness of the proposed methodology, we select four typical sampling-based algorithms (i.e., RRT [17], RRT*[3], PRM* [4] and FMT*[18]) as the benchmark planning algorithms. Furthermore, we use the GMMs-based learning method as a comparative method, through the detailed comparative analysis, to prove that our approach performs better on learning the sampling distributions. For the convenience of comparative analysis, we adopt the success rate and the path cost as the evaluation indexes of the planning performance. The path cost function τ is given as:

$$\tau = \frac{l}{l_0} \tag{10}$$

where l is the real length of the motion trajectory. l_0 is the optimal path length in the scene.

Since the sampling-based planning algorithms have randomness, we carry out 40 tests and calculate the average value as the final result in the experiment. In addition, we set $\mathcal{Z} = \mathcal{R}_z^d$ to denote the Euclidean latent distribution where d_z depends upon the complexity of the input data. We use $P_Z(Z) = \mathcal{N}(Z; 0, \sigma_z^2 \cdot \boldsymbol{I}_d)$ over $\mathcal{Z}$ to denote the isotropic Gaussian prior distributions. The cost function $c(\bullet)$ in Algorithm 2 is $c(x, y) = \|x - y\|_2^2$ where $x, y \in \mathcal{R}_z^d$. Moreover, we select the Adam optimizer [19] with $\beta_1 = 0.25, \beta_2 = 0.84$. In Algorithm 2, the parameter λ is set to 8. Furthermore, the key parameters of the GMMs-based learning method is the number of components. We determined 5 as the optimal number of components in Fig. 5(a) and determined 16 as the optimal number of components in Fig. 5(b) while running the GMMs-based learning methods. In fact, we spend a lot of time determining the appropriate component number for the GMMs-based methods. Compared to the GMMs-based

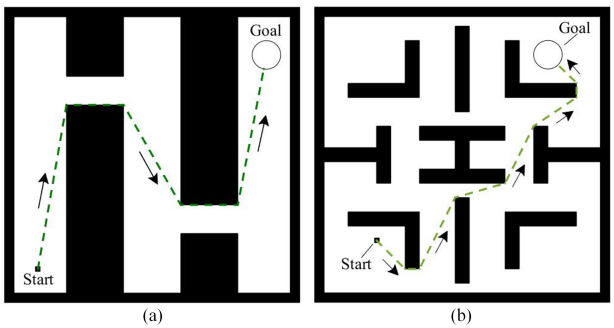

Fig. 5. The geometric planning problem: (a) planning in narrow passage; (b) planning in complex scene. The sizes of two maps in (a) and (b) both are 1×1. The black parts denote the obstacle areas. The green dotted line represents the optimal path in (a) or (b). The optimal path length in (a) is equal to about 1.8711. The optimal path length in (b) is equal to about 1.1982. In the two planning scenes, we select the circle region with a diameter of 0.1 as the goal region.

method, one key advantage of the proposed method is that it can adaptively acquire the clusters based on the "encoder-decoder" mechanism along with the continuous iterations of training process. The solutions to the geometric planning subproblem (a) with different algorithms are depicted in Fig. 6. The comparisons of the success rates and costs at different times for the geometric planning subproblem (a) are shown in Fig. 7 and Fig. 8. respectively. The solutions to the geometric planning subproblem (b) with different algorithms are depicted in Fig. 9. The detailed comparisons of the success rates and costs at different times for the subproblem (b) are depicted in Fig. 10 and Fig. 11 respectively.

Intuitively, Fig. 6 shows that our approach has better planning paths than the GMMs-based learning method for the geometric planning subproblem (a). From Fig. 7, we find that our approach clearly improves the success rates of sampling-based planning algorithms. Compared with the GMMs-based learning method, our approach also has better success rates for different sampling-based planning algorithms. Moreover, Fig. 8 indicates that the proposed GraphWAE-based learning method can effectively reduce the costs of the planning process for the standard sampling-based planning algorithms and generate higher quality paths compared with the GMMs-based learning method with an increase in planning time. The results indicates that the proposed GraphWAE can accurately represent the configuration space of collision free regions, especially for the narrow regions. Additionally, Fig. 9 shows that our approach can provide a better planning solution for subproblem (b) compared with the GMMs-based learning method. Fig. 10 and 11 further indicate our approach can effectively assist these basic sampling-based planning algorithms to improve their success rates and reduce the path costs. It shows that our approach performs better than GMMs-based learning method for the planning subproblem in the complex scene. On the whole, the two simulations have indicated that the proposed GraphWAE-based learning method is generalisable, and effectively improves the planning performance of the SBMPs.

B. 6-DOFs robot motion planning problem

Although sampling-based planning algorithms can basically solve the aforementioned geometric planning problem, it is not easy to handle the high-dimension motion planning problem in a complex scene. To further evaluate the proposed method, we use the 6 DOFs planar robot motion planning problem as the foundation of the simulation experiment. The reason for selecting the planar robot simulation is that the planar robot can provide the clear, intuitive and easy-to-be-quantified planning results. The detailed simulation environment is shown in Fig. 12 where we can see that the simulation contains multiple obstacles which present a cluttered and narrow planning scene. For sampling-based planning algorithms, it will take a considerable amount of time to check collisions in such a scene. In the simulation experiment we use the RRT and RRT* as the benchmark planning algorithms. Furthermore, we use the GMMs-based learning method as the comparative method. The key parameter λ of the proposed method is set to 7. By repeated testing, we select 17 as the optimal cluster number of the GMMs-based method. Additionally, we adopt the iteration number as the control variable in the simulation experiment. For each iteration number node, we conduct 40 tests to acquire the average of the planning results as the final result. Moreover, we select the running time, success rate and cost function ρ as the planning performance evaluation indexes. It should be noted that ρ reflects the level of motion path quality, which shows a negative correlation, that is to say, the increase of the cost value will decrease the motion path quality. The cost function ρ is given as follows:

$$\rho = \sum_{i=1}^{N_{\text{link}}} l_i \tag{11}$$

where l_i denotes the motion path length of the i-th link end of the robot arm and N_{link} denotes the link number. For this simulation, $N_{\text{link}} = 6$.

The planning results with different planning algorithms in one test at 6000 iterations are shown in Fig. 13. The comparison of solutions to 6-DOFs motion planning problem at 6000 iterations is shown in Fig. 14. Observing Fig. 13 and Fig. 14, we can find that our approach can effectively improve the motion path qualities of the basic sampling-based planning algorithms. Furthermore, we also intuitively see that the proposed method has shorter planning time than the GMMs-based method in Fig. 14. The detailed experimental results with different iteration numbers are presented in Table. I. From Table. I, we find that the GraphWAE-based learning approach can effectively improve the planning performance (i.e., planning time, cost and success rate) of the benchmark algorithms. More specifically, the planning time of our approach is reduced by approximately 33.4% and 37.2% on average respectively than the RRT and RRT*. The cost of our approach is reduced by approximately 18.5% and 14.2% on average respectively than the RRT and RRT*. With the increasing of iterations, the success rate of our approach has noticeable improvement compared with the RRT and RRT*. These results indicate that the proposed methodology is generalisable, and the GraphWAE-based learning method

Fig. 6. The solutions to the geometric planning subproblem (a) with different planning methods.

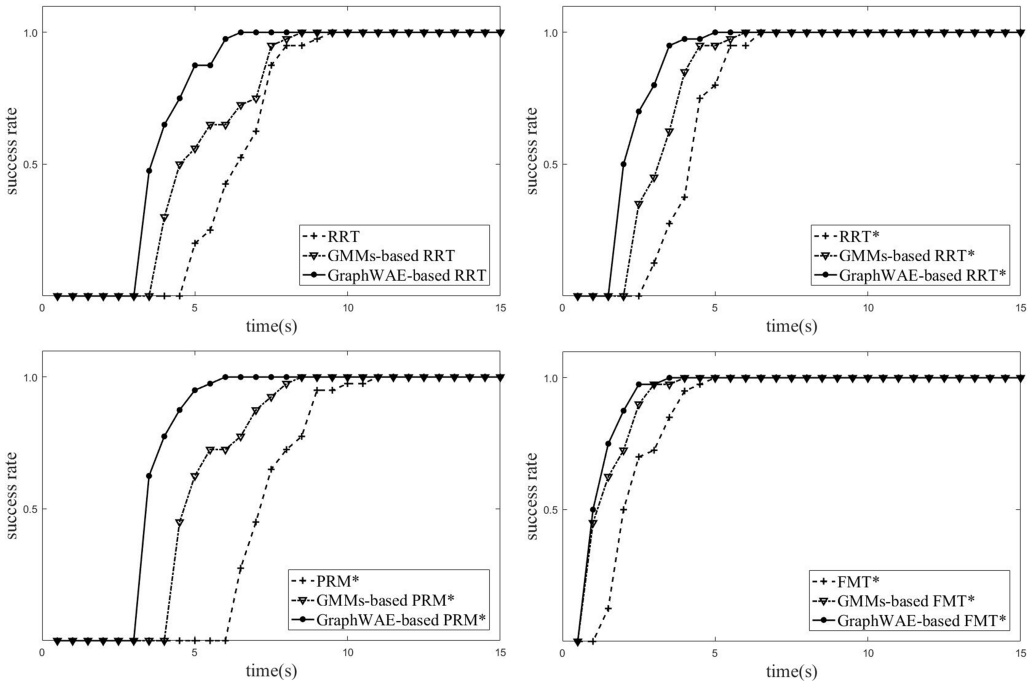

Fig. 7. Comparisons of success rates at different times for the geometric planning subproblem (a).

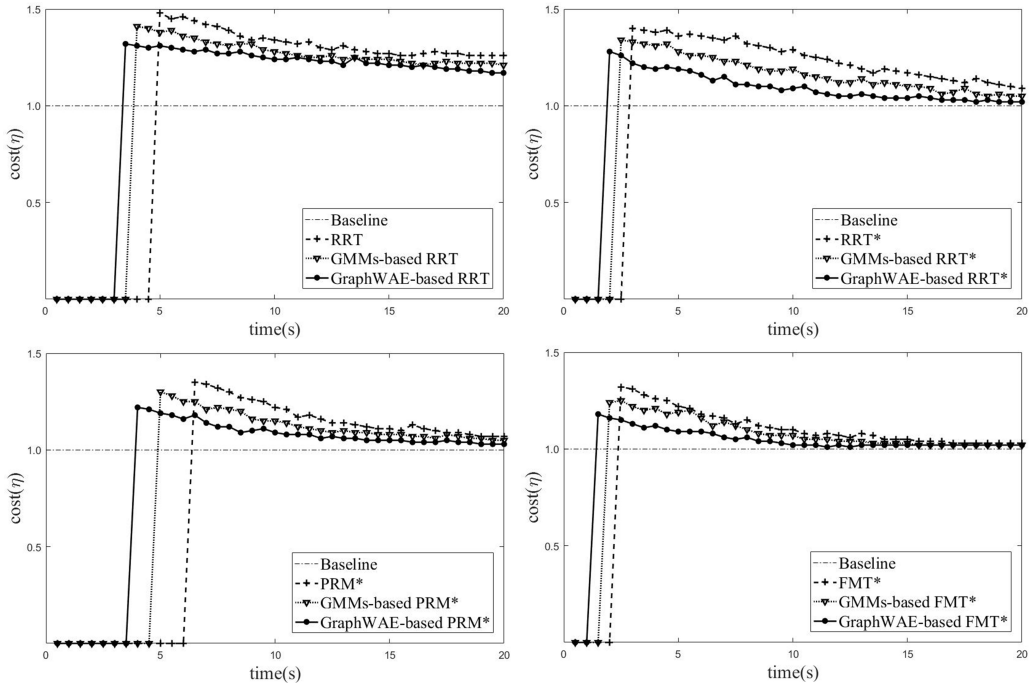

Fig. 8. Comparisons of costs at different times for the geometric planning subproblem (a).

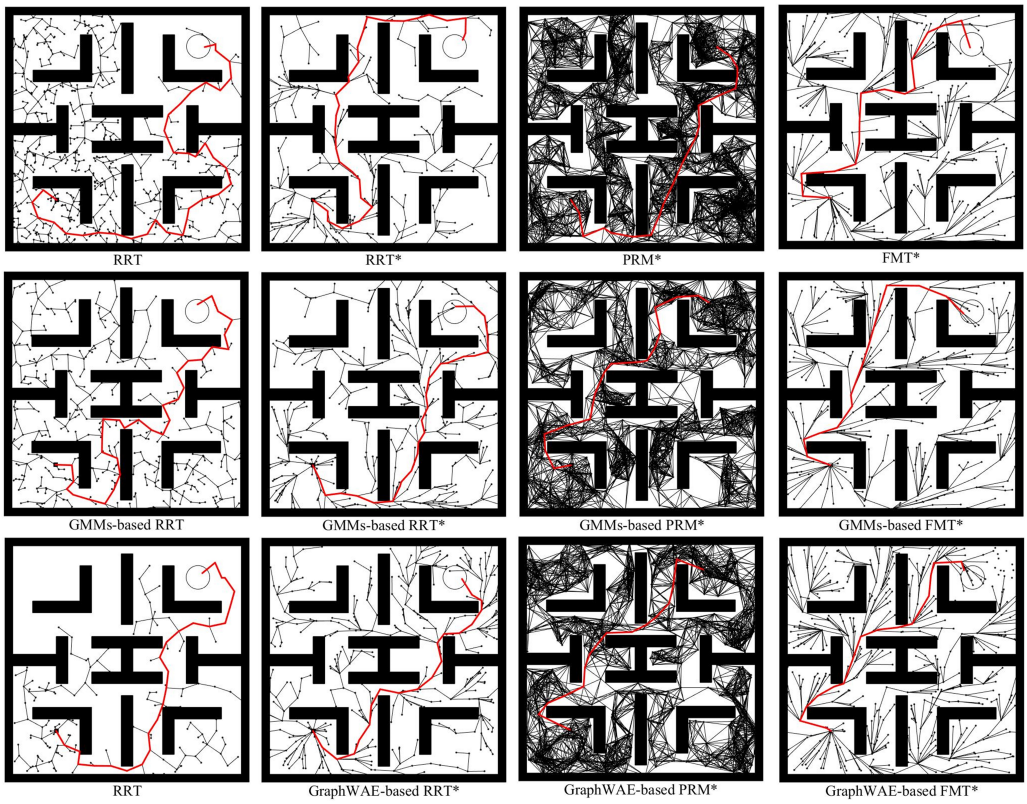

Fig. 9. The solutions to the geometric planning subproblem (b) with different planning methods.

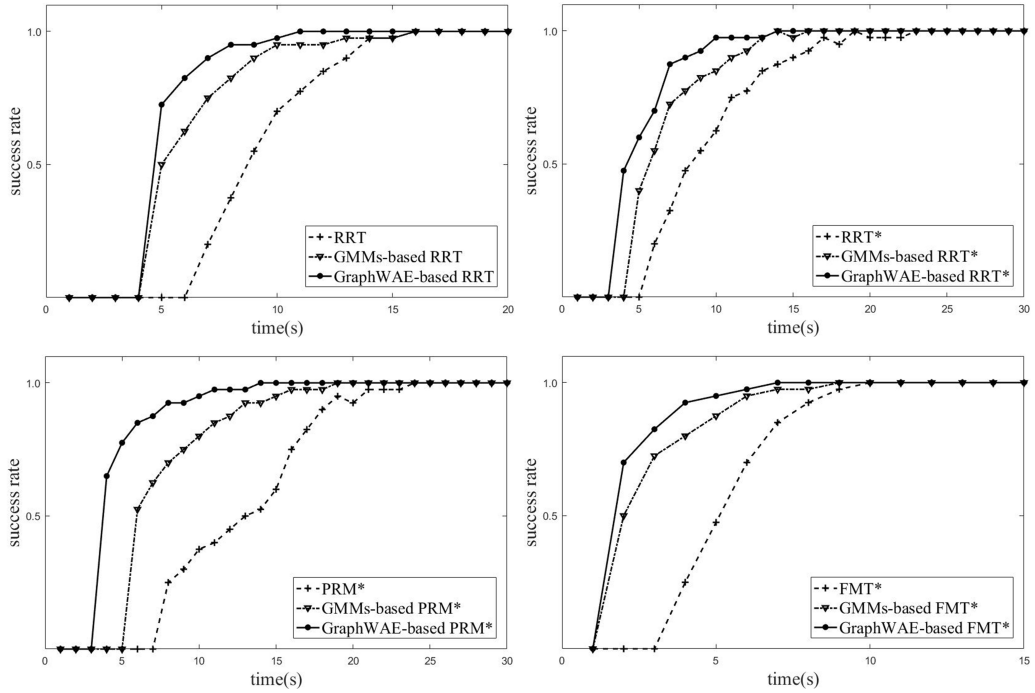

Fig. 10. Comparisons of success rates at different times for the geometric planning subproblem (b).

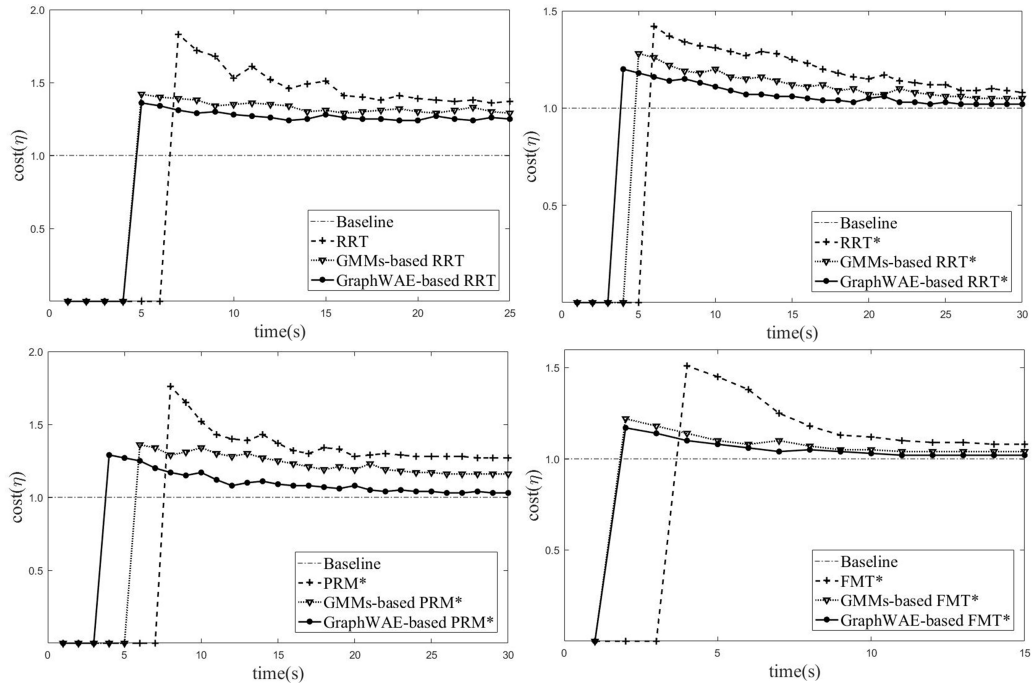

Fig. 11. Comparisons of costs at different times for the geometric planning subproblem (b).

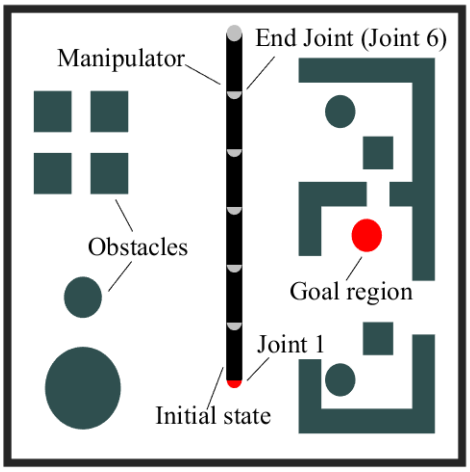

Fig. 12. The 6-DOFs planar robot motion planning problem. The size of the simulation environment is 100 × 120. The radius of the goal region is 4.0. The robot arm contains six links and six joints. The size of each link is 12 × 3. The variation range of each joint angle is $[-135°, 135°]$. For sampling-based planning methods, the joint angle variation range at each iteration is $[-5°, 5°]$. Moreover, the maximum joint angle at each neighbor search is 30°. We use the oriented bounding boxes (OBBs) as the collision detection strategy in the simulation. In post-processing, we adopt the third order B-spline curve fitting algorithm as the motion trajectory smooth method.

can effectively improve the planning performance of sampling-based planning algorithms. Furthermore, our approach has better planning performance than the GMMs-based learning method, especially for the run time and cost. For example, the planning time of our approach is reduced by approximately 17.6% and 21.6% on average respectively than the GMMs-based RRT and the GMMs-based RRT*. In addition, the cost of the GraphWAE-based RRT* is reduced by 14.7% on average compared with the GMMs-based RRT*, which presents a significant improvement in the planning path quality. The comparison results regarding success rate also indicate that the GraphWAE-based learning method can generate the qualified planning path in a shorter time than the GMMs-based method. In fact, these results reflect that the proposed GraphWAE-based learning method has a strong representation ability for the sampling distributions of the SBMPs. Furthermore, the aforementioned analyses also indicates that the proposed method is general and has a better planning performance than the GMMs-based method.

C. 12-DOFs robot motion planning problem

The dual-arm collaborative robot motion planning is a typical high-dimensional motion planning problem, which is still a challenging topic for robotic autonomous manipulation. Thus, we select the 12-DOFs robot collaborative planning problem as an example to further validate the proposed methodology. We adopt the dual-arm robot "NEXTAGE" system as the main carrier of the high-dimensional motion planning problem.

The whole simulation experiment runs on the MoveIt! platform [20] within Robot Operating System (ROS) in Ubuntu 16.04 with a computer (Intel core 2 Duo, 16GB RAM, CPU 2.50 GHZ). We use the OMPL [21] as the underlying planning repository, which can be embedded in the MoveIt! platform. The dual-arm robot system contains one 1-DOF head, one 1-DOF waist, two 1-DOF grippers and two 6-DOFs arms. In our simulation, we select two 6-DOFs arms as the manipulators to perform the collaborative planning task. We design the obstacle scene with the typical narrow passage as the simulation environment, as shown in Fig. 15. In the simulation experiment, we use the RRT and RRT* as the benchmark planning algorithms. Moreover, we still use the GMMs-based learning method as the comparative method. The key parameter of our approach is set to 10. The cluster number of the GMMs-based method is set to 14. We select the success rate, planning time and cost (i.e., path quality) as the evaluation indexes. It should be noted that we adopt the total motion path length of two arm ends as the cost value.

To ensure the evaluation quality, we conduct 40 tests and calculate the average value as the final result for every planning methods. The planning results with different methods in MoveIt! are shown in Fig. 16. The comparison of simulation results with different methods is shown in Table. II. From Fig. 16, we can intuitively see that our approach has shorter planning paths than the GMMs-based learning method, especially for the asymptotically optimal motion planner (i.e., RRT*). Table. II also shows that the cost values of the proposed methodology are reduced by approximately 16.26% and 13.06% on average than the RRT* and the GMMs-based RRT* respectively. The planning time is also a critical factor for the planning performance. Compared with the GMMs-based RRT*, the planning time of our approach is reduced by approximately 36.16%, which indicates that our approach can provide more appropriate samples to reduce the collision checks. In addition, the success rates of our approach are significantly higher than those of the GMMs-based learning method, which indicates that our approach has stronger representation ability than the GMMs-based method for learning the bias sampling distributions. In summary, the proposed methodology has a better planning performance than the GMMs-based learning method in the 12-DOFs dual-arm robot motion planning problem.

VI. Conclusion

In this paper we propose a bias sampling strategy based on GraphWAE to assist the sampling-based planners to further acquire the fast and high-quality motion planning path in complex environments. We implement three simulation experiments with several benchmark planners to validate the proposed methodology. The comparison results indicates that our approach is general and has a noticeable improvement in the planning performance for sampling-based motion planning algorithms. In addition, our approach also provides a new perspective on solving the high-dimensional optimal planning problem in the complex environment.

For the future work, we will continue promoting an in-depth study in the cross part between the graph neural networks and the robot motion planning. We will also carry out more complex experiments to further exploit the advantages of the GraphWAE. We hope that the learning-based method can make a significant contribution to the development of asymptotically optimal motion planning for robotic manipulation.

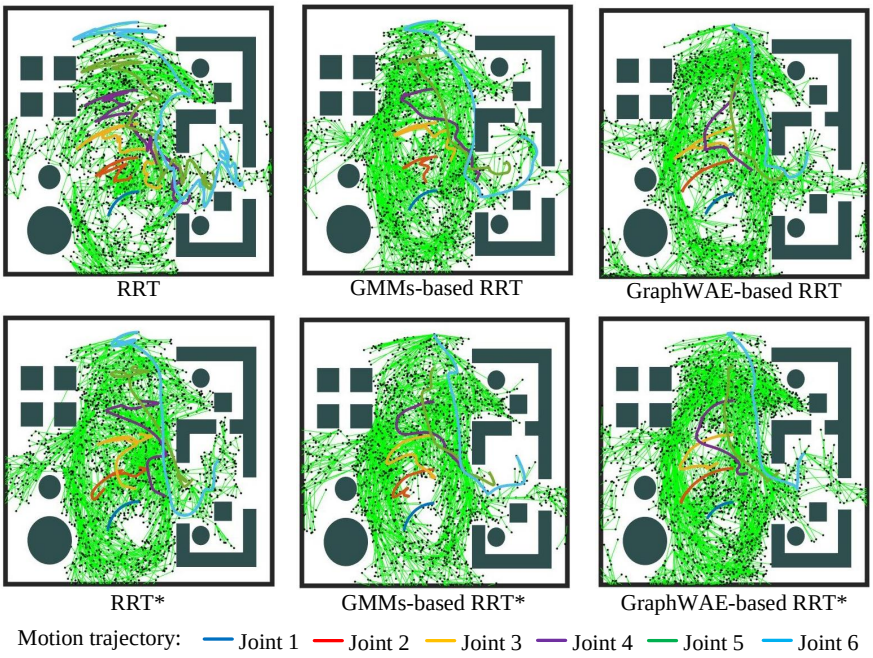

Fig. 13. Comparison of planning results using different methods at 6000 iterations.

TABLE I
COMPARISON OF EXPERIMENTAL RESULTS WITH DIFFERENT ITERATION TIMES

Iteration times		3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000
Planning time (s)	RRT	9.85	11.38	13.04	14.76	16.28	18.12	19.64	21.03	22.38	24.53	26.34
	GMMs-based RRT	8.11	9.55	11.04	12.85	14.13	15.08	15.98	16.41	17.08	17.85	18.78
	GraphWAE-based RRT	**6.04**	**6.95**	**8.11**	**9.89**	**11.04**	**11.92**	**12.83**	**14.06**	**15.18**	**15.98**	**17.23**
	RRT*	105.32	124.28	141.45	158.96	177.85	195.37	209.42	223.28	243.31	258.64	278.37
	GMMs-based RRT*	89.35	103.69	121.54	139.26	156.42	175.35	190.28	205.63	218.79	224.85	237.62
	GraphWAE-based RRT*	**61.58**	**76.28**	**91.93**	**105.16**	**119.27**	**132.84**	**148.55**	**161.81**	**177.34**	**189.58**	**195.75**
Cost	RRT	1058.46	989.14	954.72	912.28	855.65	834.23	837.49	831.52	828.74	831.27	833.65
	GMMs-based RRT	868.44	845.32	821.65	804.33	783.35	771.26	782.10	769.55	774.38	785.28	775.42
	GraphWAE-based RRT	**832.57**	**816.38**	**788.18**	**757.34**	**747.26**	**736.39**	**725.27**	**731.65**	**739.21**	**728.54**	**725.83**
	RRT*	538.26	513.73	486.55	471.62	455.25	431.58	410.86	388.79	361.05	345.76	326.84
	GMMs-based RRT*	495.37	472.19	443.76	418.75	385.58	362.15	342.82	325.17	323.35	321.46	322.75
	GraphWAE-based RRT*	**418.52**	**384.86**	**351.14**	**326.52**	**311.68**	**308.75**	**305.89**	**304.16**	**303.74**	**303.26**	**303.48**
Success rate	RRT	0.125	0.275	0.45	0.575	0.725	0.85	0.925	0.975	0.975	1.00	1.00
	GMMs-based RRT	0.225	0.425	0.55	0.675	0.75	0.875	0.925	1.00	1.00	1.00	1.00
	GraphWAE-based RRT	**0.475**	**0.65**	**0.725**	**0.875**	**0.95**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	RRT*	0.125	0.275	0.65	0.725	0.875	0.90	0.975	1.00	1.00	1.00	1.00
	GMMs-based RRT*	0.375	0.45	0.625	0.725	0.80	0.875	0.95	1.00	1.00	1.00	1.00
	GraphWAE-based RRT*	**0.50**	**0.825**	**0.95**	**0.975**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

TABLE II
COMPARISON OF EXPERIMENTAL RESULTS WITH DIFFERENT ITERATION TIMES

Methods	time (s)	success rate	cost(m)
RRT	5.42	0.20	3.26
GMMs-based RRT	3.57	0.475	3.14
GraphWAE-based RRT	**2.25**	**0.75**	**2.73**
RRT*	9.74	0.375	2.21
GMMs-based RRT*	6.83	0.525	2.05
GraphWAE-based RRT*	**4.36**	**0.80**	**1.86**

ACKNOWLEDGMENT

We would like to thank other members (Teoh Yee Seng, Yi Zhao, Vasanth Elangovan, Qi-Long Yuan, Ramamoorthy Luxman, Jin Wu and Han-Yu Song) from Robotics Research Centre (RRC) of NTU for helping us improve the research work. Additionally, the authors would like to thank experienced anonymous reviewers for their valuable and constructive suggestions for improving the overall quality of this paper.

REFERENCES

[1] S. Karaman, E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846-894, Jun. 2011.

[2] J. Schulman, Y. Duan, J. Ho, et al, "Motion planning with sequential convex optimization and convex collision

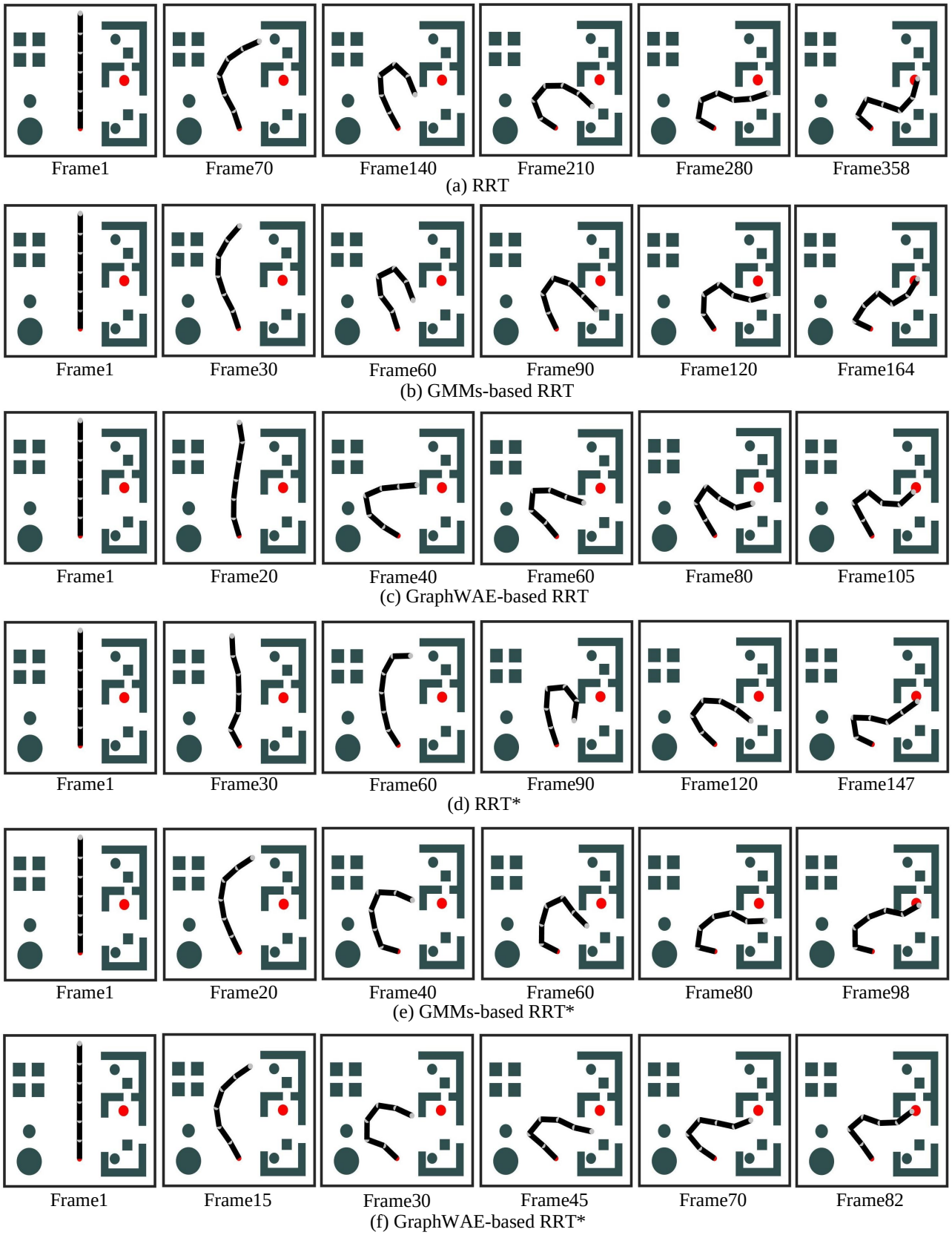

Fig. 14. Comparison of solutions to 6-DOFs robot motion planning problem using different methods at 6000 iterations.

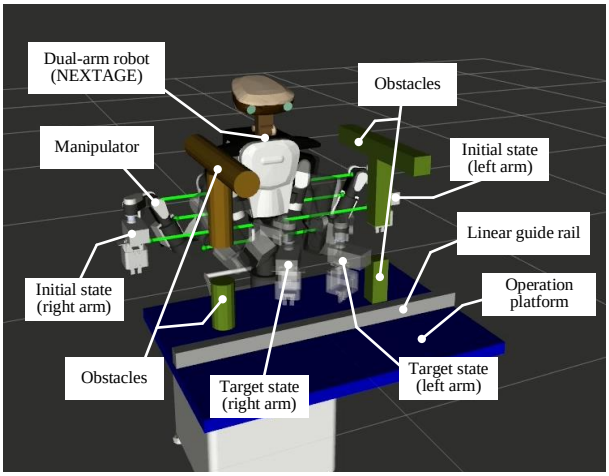

Fig. 15. The 12-DOFs dual-arm robot motion planning problem. The simulation scene is a simplified version of dual-arm collaborative manipulation project, which just focuses on the planning part. The size of the operation platform is 1.26 m $\times$ 1.24 m. The search space size of the whole planning process is 1.5m $\times$ 1.5m $\times$ 1.5m. We use the bottom center of robot waist as the origin of world coordinate system.

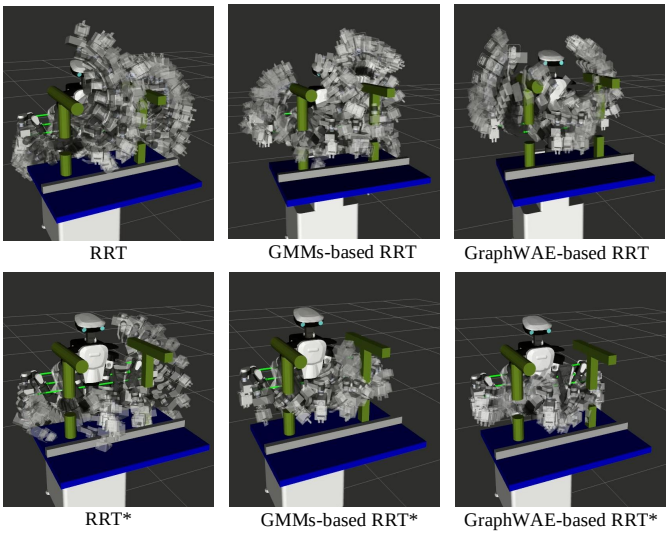

Fig. 16. The planning results with different planning methods in MoveIt!.

checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp.1251-1270, Aug. 2014.

[3] S. Karaman, M.R. Walter, A. Perez, E. Frazzoli, S. Teller, "Anytime motion planning using the RRT," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 1478-1483, May 2011.

[4] J. D. Marble, K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Trans. Robot.*, vol.29, no. 2, pp. 432-444, Jan. 2013.

[5] G. Sánchez, J. C. Latombe, "On delaying collision checking in PRM planning: Application to multi robot coordination," *Int. J. Robot. Res.*, vol. 21, no.1, pp. 5-26, Jan. 2002.

[7] J. P.Van den Berg, M. H. Overmars, "Using workspace

[6] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," *Proc. IEEE Int. Conf. Robot. Autom.*, pp. 2951-2957, May 2015.
information as a guide to non-uniform sampling in probabilistic roadmap planners," *Int. J. Robot. Res.*, vol. 24, no. 12, pp. 1055-1071, Dec. 2005.

[8] J. D. Gammell, S. S. Srinivasa, T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 3067-3074, May 2015.

[9] J. Huh, D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 63-69, May 2016.

[10] J. Huh, B. Lee, D. D. Lee, "Adaptive motion planning with high-dimensional mixture models," *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, PP. 3740-3747, may 2017.

[11] B. Ichter, J. Harrison, M. Pavone, "Learning sampling distributions for robot motion planning," *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 7087-7094, May 2018.

[12] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.,* vol. 20, no. 1, pp. 61-80, Jan. 2008.

[13] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw.*, 2020.

[14] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint*, arXiv: 1711.01558, 2017.

[15] Z. Zhang, M. Wang, A. Nehorai, "Optimal transport in reproducing kernel hilbert spaces: Theory and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1741-1754, Jul. 2020.

[16] X. Jia, M. Zhao, Y. Di, Q. Yang, J. Lee, "Assessment of data suitability for machine prognosis using maximum mean discrepancy," *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5872-5881, Jul. 2018.

[17] S. M. LaValle, J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp.378-400, May 2001.

[18] L. Janson, E. Schmerling, A. Clark, M. Pavone, "Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions," *Int. J. Robot. Res.*, vol. 34, no.7, pp. 883-921, Jun. 2015.

[19] D. Kingma, J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] D. Coleman, I. Sucan, S. Chitta, N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint*, arXiv:1404.3785, 2014.

[21] I. A. Sucan, M. Moll and L. E. Kavraki, "The open motion planning library", *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72-82, Dec. 2012.