Hindawi
Complexity
Volume 2018, Article ID 4358747, 14 pages
https://doi.org/10.1155/2018/4358747

Research Article

Robot Motion Planning Method Based on Incremental High-Dimensional Mixture Probabilistic Model

Fusheng Zha,[1] Yizhou Liu,[1] Xin Wang,[2] Fei Chen,[3] Jingxuan Li,[1] and Wei Guo[1]

[1]*State key Laboratory of Robotics and Systems, Harbin Institute of Technology, Harbin, China*
[2]*Shenzhen Academy of Aerospace Technology, Shenzhen, China*
[3]*Istituto Italiano di Tecnologia, Via Morego 30, Genova, Italy*

Correspondence should be addressed to Xin Wang; xinwanghit07s@gmail.com and Fei Chen; fei.chen@iit.it

Received 11 June 2018; Revised 18 August 2018; Accepted 19 September 2018; Published 1 November 2018

Guest Editor: Andy Annamalai

The sampling-based motion planner is the mainstream method to solve the motion planning problem in high-dimensional space. In the process of exploring robot configuration space, this type of algorithm needs to perform collision query on a large number of samples, which greatly limits their planning efficiency. Therefore, this paper uses machine learning methods to establish a probabilistic model of the obstacle region in configuration space by learning a large number of labeled samples. Based on this, the high-dimensional samples' rapid collision query is realized. The influence of number of Gaussian components on the fitting accuracy is analyzed in detail, and a self-adaptive model training method based on Greedy expectation-maximization (EM) algorithm is proposed. At the same time, this method has the capability of online updating and can eliminate model fitting errors due to environmental changes. Finally, the model is combined with a variety of sampling-based motion planners and is validated in multiple sets of simulations and real world experiments. The results show that, compared with traditional methods, the proposed method has significantly improved the planning efficiency.

1. Introduction

In recent years, as robots play an increasingly important role in industrial production and daily life, the issue of motion planning has received extensive attention. Although modern robots have significant differences in configuration, size, perception, driving methods, and application scenes, autonomous navigation and planning in complex environments are common problems faced by almost all robots [1].

The motion planning problem refers to, given the related description of robot and environment, initial state and goal region, seeking a series of control inputs to drive the robot to complete the movement from the initial state to the goal region while satisfying the environmental constraints (without colliding with the obstacles). However, for some robots with high planning dimensions, their configuration space's obstacle region cannot be explicitly described. In response to this problem, sampling-based motion planning algorithms have developed rapidly and received widespread attention [2–4]. This type of methods do not explicitly describe obstacles. Instead, they rely on the collision query module to provide feasibility information of the candidate trajectories and connect a series of collision-free samples to generate a feasible path from the initial state to the goal region. The collision query module is generally implemented by the robot kinematics calculation and the space bounding box principle [5], which requires a large computational overhead to make the module a major bottleneck for limiting the efficiency of the sampling-based motion planning algorithms [6]. Sampling-based motion planning algorithms can generate a large number of labeled samples with spatial collision information in planning instances, which provides a necessary condition for the implementation of machine learning methods. If the robots can learn from past planning experience to guide the future planning tasks, more efficient motion planning can be achieved. Therefore, how to use machine learning methods to break the efficiency bottleneck of motion planning algorithms has become a research focus in this field in recent years.

A large amount of research work is devoted to performing adaptive sampling and guiding the planning algorithm to explore certain areas in the configuration space with machine learning methods. Dalibard et al. [7] use the principal component analysis (PCA) method for online analysis of samples to estimate the direction and position of the local narrow passage in collision-free space and increase the sampling density in the direction of the passage axis. Similar methods include bridge test [8] and retraction strategy [9, 10]. These methods can significantly improve the efficiency of the planning algorithm at the local narrow passage. Brock and Burns [11] propose an exploration strategy based on entropy guidance, which can guide the planning algorithm to sample the areas that maximize information gain. However, because of the high computational cost of this method, it is more suitable for the off-line path graph construction of multiqueue query algorithms like PRM. Arslan and Tsiotras [12] use the kernel function to learn the feasibility and heuristics of samples generated in the previous planning instances and increase the sampling density of the areas that may make current path cost lower. This method can improve the convergence rate of asymptotically optimum motion planning algorithms RRT$^{\#}$ [13]. Bialkowski et al. [14] propose to store empirically observed estimates of collision-free space in a point proximity data structure and then use kd-tree algorithm to generate future valid samples.

In addition, by learning the past experience of motion planning, the prediction of the new samples' feasibility can be achieved. Burns et al. [15] present a model-based motion planning method. This method utilizes the local weighted regression to incrementally learn the approximate model of the configuration space, and the classification of new samples is implemented. Compared with the traditional collision query module, the method has a smaller computational complexity. Yang et al. [16] propose a neural networks framework to achieve robot automatic collision avoidance. By exploiting the joint space redundancy, the human operator would be able to only concentrate on motion of robots end effector without concern over possible collision. Pan et al. [17] report that the collision query results produced in previous motion planning tasks are stored in a data set. When the new sample is generated, the KNN algorithm is used to search the k nearest neighbors of the sample in the data set, and the probability of its feasibility is estimated according to the neighbors' collision information. Yang et al. [18] combine the flexibility of the Gauss process (GP) with the efficiency of the RRT algorithm and establish a motion model to predict the movement of obstacles, so that the robot can find a safe path in the dynamic environment constraints. Huh et al. [19] propose that the Gaussian mixture model can be used to fit the probability distribution of the high-dimensional space obstacle region, so as to quickly predict the feasibility of the new samples. However, the influence of the number of Gaussian components on the model prediction accuracy has not been analyzed and considered. Besides, some other learning methods like conditional variational autoencoder (CVAE) [20], neural learning [21], experience graphs (E-Graphs) [22], and dynamic movement primitives (DMPs) [23] are also used in robot motion planning problem.

The above methods improve the efficiency of motion planning to some extent, but there are the following problems:

(1) Some lazy learning methods such as kd-tree and KNN require data sets with large sample size. In the motion planning problem of high-dimensional space, this will lead to great spatial complexity which is hard to realize.

(2) Models like PCA, CVAE, E-Graphs, and local weighted regression have poor flexibility. Even if the environment or the base of robot changes slightly, the model needs to be retrained, which greatly increases the computational overhead.

Therefore, in this article, we propose to use Gaussian mixture model (GMM) as a prior model of robot configuration space to improve the efficiency of robot motion planning. GMM can represent the unknown model by the linear combination of Gaussian probability density functions. It has the ability to fit the continuous probability density distribution in any dimensional space with small spatial complexity. On the other hand, the Greedy EM algorithm utilized in this paper can realize incremental training of GMM, so that the model can be updated according to the environment changes without retraining.

In general, the main contributions of this paper include the following: (1) The influence of number of components on the prediction accuracy is analyzed, and a method for adaptively training GMM of collision region in robot high-dimensional configuration space based on the convergence of log-likelihood function is proposed. (2) The Greedy EM algorithm is used to update the GMM online with real-time 3D map to adapt to environmental changes. (3) The above method is applied to a variety of sampling-based motion planning algorithms, and the planning efficiency is significantly improved.

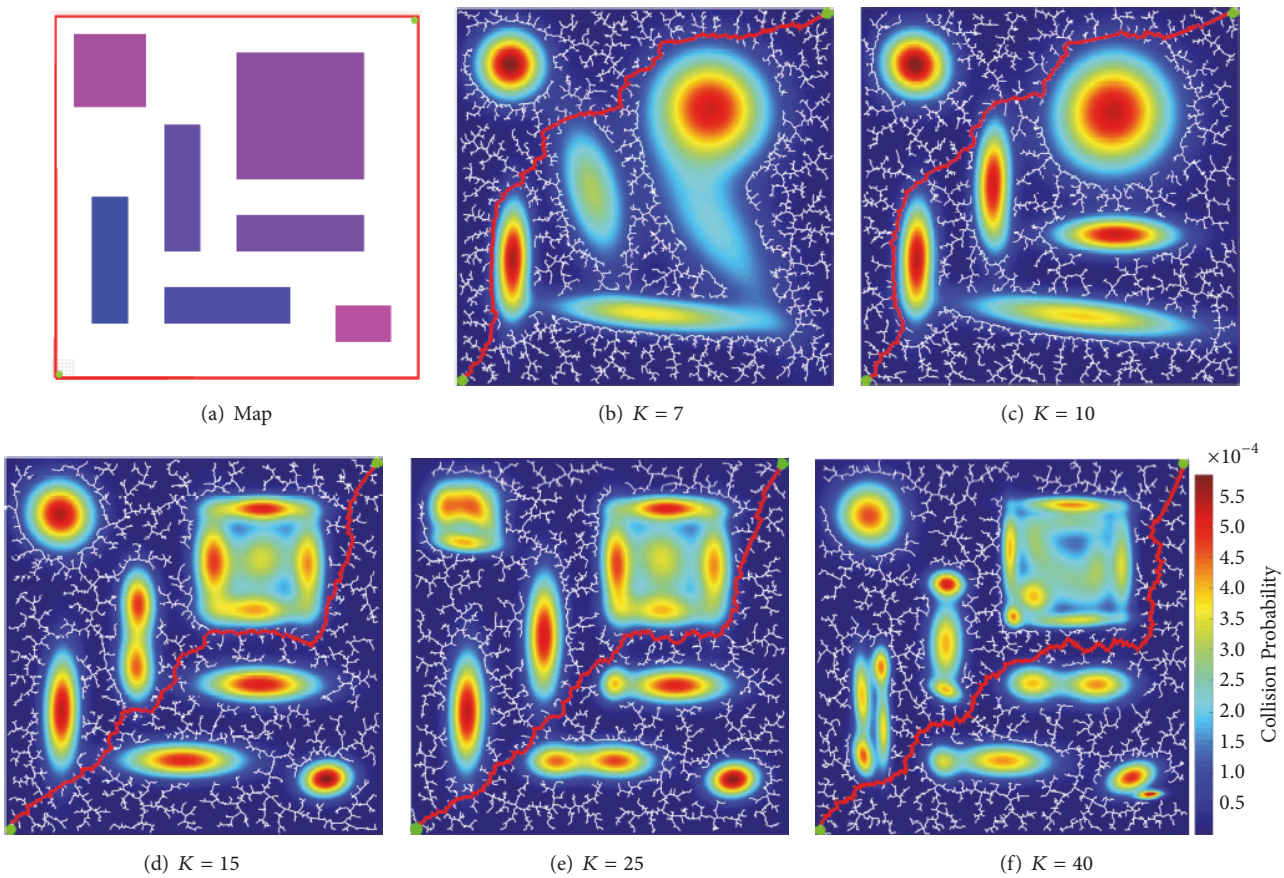
The paper is organized as follows: Section 2 introduces the main motivation of the research work; Section 3 introduces the incremental training process of GMM using Greedy EM clustering algorithm, and the integration with motion planning algorithms; Section 4 shows the simulations and the real world experiments' results; Section 5 summarizes the results and provides directions for future work.

2. Motivations and Problem Statement

The Gaussian mixture model [24] is a mixture probabilistic model that can fit the probability density distribution of arbitrary dimensions and arbitrary shapes by weighting the probability density functions of multiple Gaussian distributions.

$$P(x \mid \theta) = \sum_{k=1}^{K} \pi_k \phi(x \mid \theta_k) \tag{1}$$

where π_k is the component weights and satisfied $\pi_1 + \pi_2 + \cdots + \pi_K = 1, \pi_k \geq 0$. $\phi(x \mid \theta_k)$ is the probability density function of the kth Gaussian distribution, and θ_k is the parameter vector of the distribution:



(a) Map (b) $K = 7$ (c) $K = 10$

(d) $K = 15$ (e) $K = 25$ (f) $K = 40$

FIGURE 1: The Gaussian mixture model with different number of components is used to fit the probability distribution of the obstacle regions in the two-dimensional map (a), and the GMM-based collision query method is used to guide the RRT* algorithm to plan the path from the starting point to the end point. The white lines represent the feasible edges that RRT* algorithm explores, the green points are the starting point and the end point of the planning problem, and the red line represents the final path.

$$\phi(x \mid \theta_k) = (2\pi)^{-d/2} |S_k|^{-1/2} \cdot \exp\left[(-0.5)(x - m_k)^{\mathrm{T}} S_k^{-1} (x - m_k)\right] \tag{2}$$

where d is the dimension. m_k and S_k are the mean and covariance matrix of the Gaussian component, respectively.

By using the sample set $\{x_1, x_2, \ldots, x_N\}$ with sample size N, the parameters of the Gaussian mixture model can be estimated by the basic EM algorithm. The basic EM algorithm process is as follows:

Set the number of components K and the parameters' initial value $\theta^{(0)}$ properly, and start iterating.

Step E. According to the current model parameters, calculate the expected probability $P(k \mid x_j)$ of each Gaussian component to each observation data x_j.

$$P(k \mid x_j) = \frac{\pi_k \phi(x_j \mid \theta_k)}{\sum_{k=1}^{K} \pi_k \phi(x_j \mid \theta_k)}, \quad j = 1, 2, \ldots, N \tag{3}$$

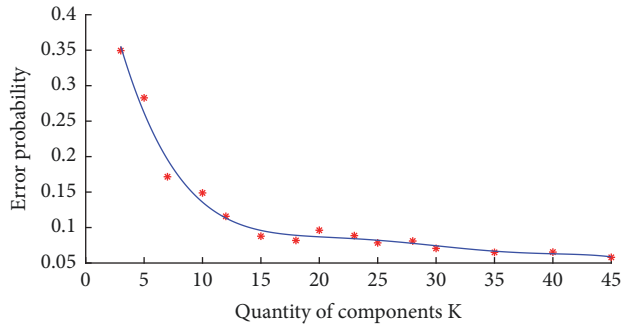
Step M. The weight of each Gaussian component π_k, the mean m_k, and the covariance matrix S_k are adjusted by maximum likelihood estimate method.

$$\pi_k = \frac{1}{N} \sum_{j=1}^{N} P(k \mid x_j)$$
$$m_k = \frac{\sum_{j=1}^{N} P(k \mid x_j) x_j}{\sum_{j=1}^{N} P(k \mid x_j)} \tag{4}$$
$$S_k = \frac{\sum_{j=1}^{N} P(k \mid x_j)(x_j - m_k)(x_j - m_k)^{\mathrm{T}}}{\sum_{j=1}^{N} P(k \mid x_j)}$$

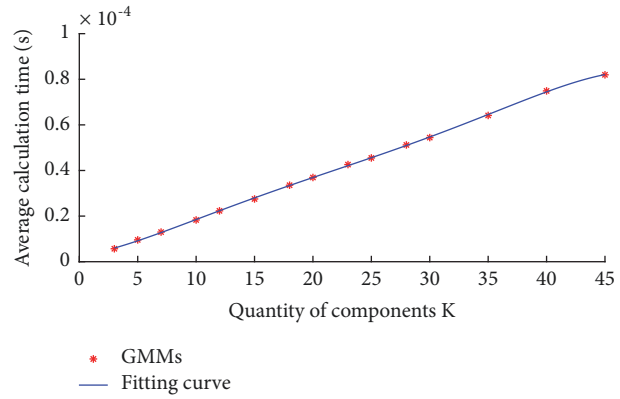
The E step and the M step are repeated to maximize the log-likelihood function $\mathscr{L}_K$ and improve the fitting ability of GMM to the sample set.

$$\mathscr{L}_K = \sum_{j=1}^{N} \log P(x_j \mid \theta) \tag{5}$$

For simplicity, first, the above basic EM clustering algorithm is used to train the GMMs with different number of components to fit the obstacle area in two-dimensional map (Figure 1(a)). This process will naturally extend later to



(a) Generalization error probability of GMMs



(b) The average calculation time required for GMMs to perform probabilistic calculations

FIGURE 2: Comparison of the GMMs' performance under different numbers of components.

multiple dimensions. The trained GMM is used to calculate the collision probability of the new samples, so as to guide the RRT* [1] (a RRT-like motion planning algorithm with asymptotically optimal) search for the feasible path from the starting point to the end point.

Figures 1(b)–1(f) show the influence of number of Gaussian components on the model fitting effect and the motion planning result. When $K = 7$, the fitting accuracy of the GMM is low, resulting in a large number of feasible regions in the two-dimensional map being blocked, and the planned path repeatedly passes through the obstacle region. Obviously, this kind of GMM cannot meet the needs of motion planning problems. With the increasing number of model components, the fitting accuracy of the GMM to the obstacle region is continuously improved. When $K = 15$, the contour of the high collision probability area is basically consistent with the obstacles in the map, which indicates that the GMM has the ability to perform collision query on new samples. If the number of Gaussian components continues to increase, the GMM's fitting accuracy is not significantly improved.

Subsequently, the generalization error of GMM under different number of components is further calculated and analyzed. As shown in Figure 2(a), with the increasing number of Gaussian components, the probability of generalization error on new samples is reduced. After $K = 15$, if the number of components continues to increase, the performance of the GMM is not significantly improved, which is basically in accordance with the result shown in Figure 1.

On the other hand, the time to calculate collision probability of new samples under different number of components is calculated. As shown in Figure 2(b), the average calculation time is linearly positively correlated with the number of Gaussian components. This indicates that the increasing number of components will lead to greater computational overhead when performing probabilistic calculations on new samples and limit the efficiency of motion planning algorithms. Therefore, selecting the appropriate number of components K according to the actual environment can maximize the performance of GMM in the sampling-based motion planning algorithm.

Therefore, using the basic EM algorithm to train the GMM of the collision region has the following problems:

(1) The environment in which the robots are located is very different and requires different number of components, so it is impossible to artificially specify K to suit the needs of all environments.

(2) When the robot's environment changes, the GMM trained with the basic EM algorithm will fail and need to be retrained, which indicates that it does not have the ability to update online according to the environment.

3. Algorithms

To solve the above problems, the Greedy EM algorithm is used to perform incremental training on the GMM, and the K value is adaptively selected according to the convergence of the log-likelihood function. The model has the online learning ability to adapt to the environment changes. The GMM is used to perform collision query routine in a variety of sampling-based motion planning frameworks. The algorithm framework is shown in Figure 3.

3.1. The Training Method of the Probabilistic Model. As shown in Algorithm 1, X_{col} is the sample set of collision region in robot configuration space; G_{col} is the GMM to describe the distribution of the collision region; k is the number of components of the current GMM; $\mathbf{a}, \mathbf{m}, \mathbf{S}$ is an array of weights, mean values, and covariance matrices for all components of the current GMM; and $\mathscr{L}_k$ is the value of the overall log-likelihood function under the current number of Gaussian components k.

The incremental training process of GMM is divided into two parts: global EM iteration and binary EM iteration. The global EM iteration process is exactly the same as the basic EM algorithm. The maximum likelihood estimation is used to adjust the weights, mean values, and covariance matrices of all Gaussian components in the current GMM to improve fitting ability to the sample set X_{col}, as shown in (3), (4), and (5).

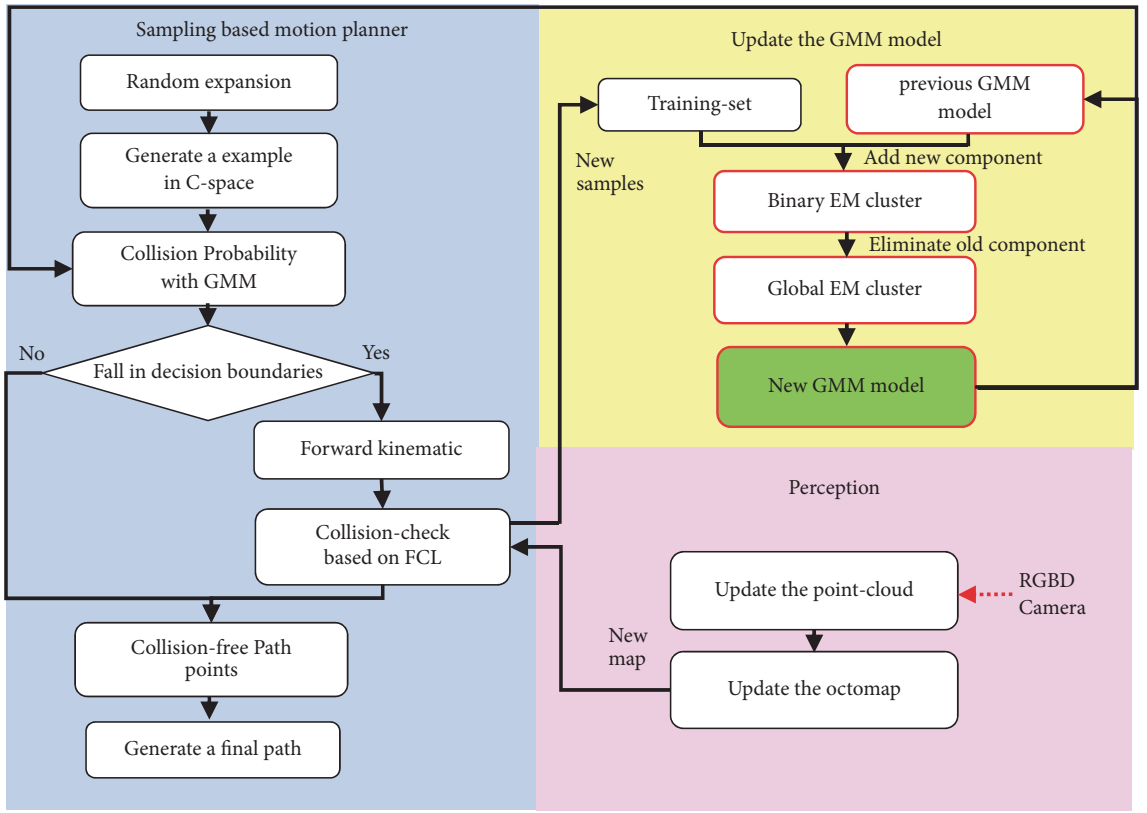


Figure 3: Algorithm framework: the main contributions are highlighted in red.

Binary EM iteration is the key step in incremental training. Add a new component $\phi(x \mid \theta)$ to the GMM $f_k(x)$ with k components.

$$f_{k+1}(x) = (1 - a_{new}) f_k(x) + a_{new}\phi(x \mid \theta), \quad a_{new} \in (0, 1) \tag{6}$$

The new component's weight a_{new} is initialized by the following equation.

$$a_{new} = \begin{cases} 0.5, & \text{if } k = 1 \\ \dfrac{2}{k+1}, & \text{if } k \geq 2 \end{cases} \tag{7}$$

The mean m_{new} is set to a randomly selected sample, and the covariance matrix S_{new} is initialized to $\sigma^2 I$. According to [24], σ depends on the dimension of the planning problem d, sample size N, and a fixed number β which is set to half of the maximum singular value of samples' covariance matrix (8).

$$\sigma = \beta \left[\frac{4}{d+2} N\right]^{1/(d+4)} \tag{8}$$

To make the new GMM better fit the sample set, the new component's weight and the parameters vector θ need to be adjusted. The optimization goal is to maximize the binary log-likelihood function.

$$\begin{aligned} \mathscr{L}_{k+1} &= \sum_{j=1}^{N} \log f_{k+1}(x_j) \\ &= \sum_{j=1}^{N} \log\left[(1 - a_{new}) f_k(x_j) + a_{new}\phi(x \mid \theta)\right] \end{aligned} \tag{9}$$

According to the maximum likelihood estimation, binary EM iterations are performed on the current GMM and the new Gaussian component. The binary EM iteration process is as follows.

Set the initial number of components k to 1, and initialize the parameters of the component according to the above method.

Step E. Initialize the new Gaussian component' parameters θ and calculate its expected probability to each observation sample x_j.

$$P(k+1 \mid x_j) = \frac{a_{new}\phi(x_j \mid \theta)}{(1 - a_{new}) f_k(x_j) + a_{new}\phi(x_j \mid \theta)} \tag{10}$$

1: **function** BUILD_GMM(X_{col})
2: $G_{col} \longleftarrow$**INITIAL_GMM**(X_{col})
3: **while** $\mathscr{L}_k/\mathscr{L}_{k-1} - 1 > 1e^{-3}$ **do**
4: $m_{new} \longleftarrow$RandomSelect(X_{col})
5: $S_{new} \longleftarrow \sigma^2 I$
6: $a_{new} \longleftarrow 2/(k+1)$
7: $(a_{new}, m_{new}, S_{new}) \longleftarrow$ **BINARY_EM**($G_{col}, a_{new}, m_{new}, S_{new}$)
8: $G_{col} \longleftarrow$ AddToGMM($G_{col}, a_{new}, m_{new}, S_{\text{new}}$)
9: $G_{col} \longleftarrow$**GLOBLE_EM**(G_{col})
10: **end while**
11: **return** G_{col}
12: **end function**
13: **function** INITIAL_GMM(X_{col})
14: $\mathbf{a}[1] \longleftarrow 1$
15: $\mathbf{m}[1] \longleftarrow \mathrm{E}(X_{col})$
16: $\mathbf{S}[1] \longleftarrow \mathrm{Cov}(X_{col})$
17: $G_{col} \longleftarrow$**GLOBLE_EM**(G_{col})
18: **return** G_{col}
19: **end function**
20: **function** ADDTOGMM ($G_{col}, a_{new}, m_{new}, S_{new}$)
21: **for** $j = 1 \longrightarrow k$ **do**
22: $a[j] \longleftarrow a[j]/(1 + a_{new})$
23: **end for**
24: $\mathbf{a}[k+1] \longleftarrow a_{new}$
25: $\mathbf{m}[k+1] \longleftarrow m_{new}$
26: $\mathbf{S}[k+1] \longleftarrow S_{new}$
27: **return** G_{col}
28: **end function**

Algorithm 1: The Greedy EM Algorithm.

Step M. Maximize the log-likelihood function (9) by adjusting the new component's weight a_{new}, mean m_{new}, and covariance matrix S_{new}.

$$a_{new} = \frac{1}{N}\sum_{j=1}^{N} P\left(k+1 \mid x_j\right)$$

$$m_{new} = \frac{\sum_{j=1}^{N} P\left(k+1 \mid x_j\right) x_j}{\sum_{j=1}^{N} P\left(k+1 \mid x_j\right)} \tag{11}$$

$$S_{new} = \frac{\sum_{j=1}^{N} P\left(k+1 \mid x_j\right)\left(x_j - m\right)\left(x_j - m\right)^{\mathrm{T}}}{\sum_{j=1}^{N} P\left(k+1 \mid x_j\right)}$$

Then, as shown in line 20 to line 28 of Algorithm 1, the weights of all components in the current GMM are diluted, and the new component trained by the binary EM algorithm is added to the GMM. Then the new GMM's parameters are adjusted through the global EM iterations until convergence.

As shown in line 3 to line 10 of Algorithm 1, the global EM iterations and binary EM iterations are alternated to implement the incremental training of GMM. When the overall log-likelihood function value $\mathscr{L}_k$ becomes stable (which means $\mathscr{L}_k/\mathscr{L}_{k-1} - 1 < 1e^{-3}$, as shown in line 3 of Algorithm 1), the addition of a new Gaussian component is stopped and the training is completed. In this way, according to the convergence of the overall log-likelihood function, the number of Gaussian components K can be adaptively adjusted without being specified in advance, so as to reasonably weigh the accuracy and computational efficiency of the GMM. Beside, compared with the basic EM algorithm, the GMM trained by Greedy EM algorithm has better fitting accuracy [25].

3.2. Online Updating of Gaussian Mixture Model. The actual working environment of the robot is dynamic and uncertain. The Gaussian mixture model trained off-line by the above method may cause errors due to the changes of environment or robot base, so the GMM is required to have the ability to update online. Through the Greedy EM algorithm, a new Gaussian component is added to fit the new collision region due to environmental changes, and some useless Gaussian components are eliminated.

As shown in line 2 to line 4 of Algorithm 2, the octree map $\mathscr{O}_{octree}$ is updated according to the point cloud data $\mathscr{P}_{pointcloud}$ acquired by the RGB-D camera. The sample set X_{col} of the high-dimensional collision region is updated according to the new octree map and robot kinematics model $\mathscr{K}_{robot}$. Then, according to (1) and (2), the probability that each sample belongs to the current GMM is calculated. If the probability is small, it indicates that the sample is a new collision sample point due to environmental changes. In this way, new samples are screened, and a sample is randomly selected as the initial mean value of the newly added Gaussian component. Finally, the Greedy EM algorithm is used to train the GMM, and the new component is added continuously until the log-likelihood function converges.

As shown in line 13 of Algorithm 2, G_{col} is the GMM to fit the collision region in configuration space. in order to prevent the infinite increase of the number of components and inefficiency of calculation, some useless Gaussian components in the new GMM need to be eliminated, such as the weights being very small, or the components with covariance matrix determinant close to 0, as shown in Figure 4. Other components obtain the weight of the removed components according to their respective weights. The sum of weights of all components is always 1. This process enables the GMM to adapt quickly and accurately to environment changes.

3.3. GMM-Based Motion Planner. After the GMM of the high-dimensional collision region is established by the Greedy EM algorithm, the probability of the new samples belonging to the GMM can be calculated by (1) and (2), thereby determining whether the sample is feasible. In this way, the forward kinematics calculation and precise collision query are avoided, and the total collision query time is greatly shortened.

As shown in line 7 to line 13 of Algorithm 3, if the probability of the new sample $\mathscr{P}(q_{new})$ belonging to the current GMM is less than the upper probabilistic boundary of the collision-free region $\delta_{col-free}$, the sample is considered feasible. If the probability of the new sample $\mathscr{P}(q_{new})$ belonging to the GMM is greater than the lower probabilistic boundary δ_{col} of the collision region, the sample is considered

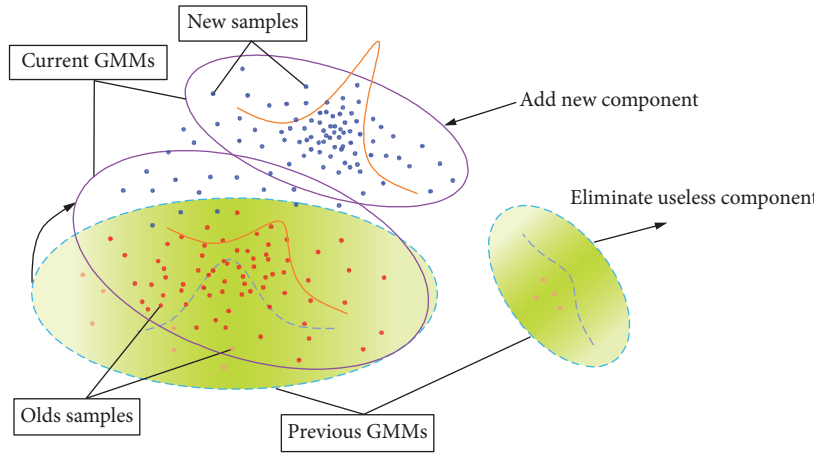


Figure 4: The schematic diagram of GMM's online updating process.

1: **function** UPDATE_GMM($G_{col}, X_{col}, \mathscr{P}_{pointcloud}$)
2: $\mathscr{O}_{octree}.update(\mathscr{P}_{pointcloud})$
3: $X_{col}.update(\mathscr{O}_{octree}, \mathscr{K}_{robot})$
4: $X_{col\_new} \longleftarrow$ ScreenSamples(X_{col}, G_{col})
5: **while** $\mathscr{L}_k/\mathscr{L}_{k-1} - 1 > 1e^{-3}$ **do**
6: $m_{new} \longleftarrow$RandomSelect(X_{col_new})
7: $S_{new} \longleftarrow \sigma^2 I$
8: $a_{new} \longleftarrow 2/(k+1)$
9: $(a_{new}, m_{new}, S_{new}) \longleftarrow$ **BINARY_EM**($G_{col}, a_{new}, m_{new}, S_{new}$)
10: $G_{col} \longleftarrow$AddToGMM($G_{col}, a_{new}, m_{new}, S_{new}$)
11: $G_{col} \longleftarrow$**GLOBLE_EM**(G_{col})
12: **end while**
13: $G_{col} \longleftarrow$Eliminate(G_{col})
14: **return** G_{col}
15: **end function**

Algorithm 2: GMM's Online Updating.

Require: X_{col} examplars
1: $G_{col} \longleftarrow$**BUILD_GMM**(X_{col})
2: $\mathscr{D}.init(q_{init})$
3: **while** Distrance(q_{goal}, q_{new}) < d_{min} **do**
4: $q_{rand} \longleftarrow$RandomSampling()
5: $q_{near} \longleftarrow$NodeSelection($\mathscr{D}, q_{rand}$)
6: $q_{new} \longleftarrow$NodeExpansion($\mathscr{D}, q_{rand}, q_{near}$)
7: **if** $\mathscr{P}(q_{new}) < \delta_{col-free}$ **then**
8: q_{new} is collision-free
9: **else if** $\mathscr{P}(q_{new}) > \delta_{col}$ **then**
10: q_{new} is collision
11: **else**
12: FCLCollisionQuery(q_{new})
13: **end if**
14: **end while**
15: $G_{col} \longleftarrow$**UPDATE_GMM**($G_{col}, X_{col}, \mathscr{P}_{pointcloud}$)

Algorithm 3: GMM-based Motion Planner.

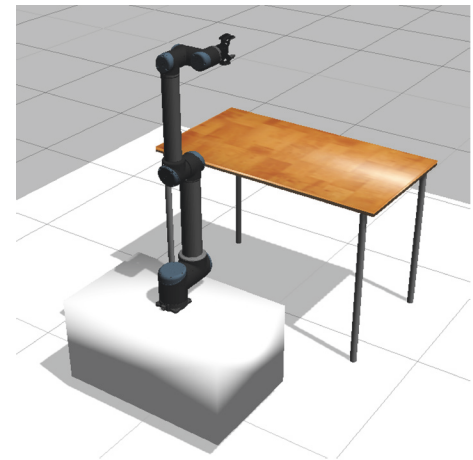


Figure 5: GMM's training scene.

unfeasible. If the probability is between the δ_{col} and $\delta_{col-free}$, the GMM's prediction result is ambiguous. Therefore, the precise collision query on these samples will be implemented by the kinematics and Flexible Collision Library (FCL) [26].

δ_{col} and $\delta_{col-free}$ can be determined by cross-validation. Cross-validation is a method commonly used in machine learning to evaluate the prediction effect of a model with a small sample set. 1000 samples in the collision region and collision-free region are collected, respectively, to form two cross-validation sets. Through multiple tests, the probability of GMM' generalization error for two cross-validation sets under different decision boundaries δ can be obtained. For robot motion planning, false positive means that the collision-free samples are classified as collision samples, which may cause the motion planner to fail to find a feasible solution. False negative means that the collision samples are classified as collision-free samples and may cause the final path to pass through collision region.

For example, in the environment of Figure 5, the GMM is trained by Greedy EM algorithm to fit collision region in 6-DOF robot's configuration space. Then, the above-mentioned

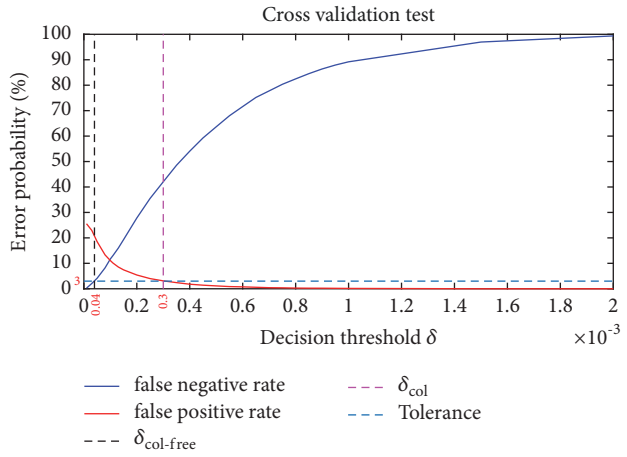


Figure 6: Cross-validation results.

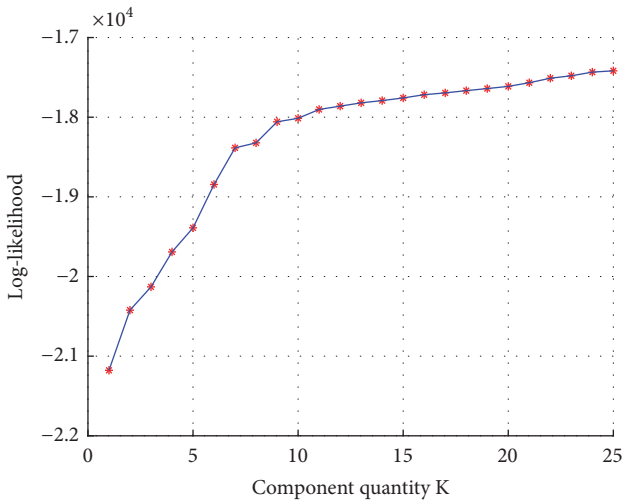


Figure 7: The learning curve of Greedy EM algorithm.

cross-validation is performed on the six-dimensional GMM to obtain the generalization error curves as shown in Figure 6. In order to guarantee the safety and success rate of motion planning, the tolerable generalization error probability is set to 3.0%, which is the empirical value obtained by many simulation experiments. The intersections with two generalization error curves are δ_{col} and $\delta_{col-free}$.

4. Simulations and Experiments

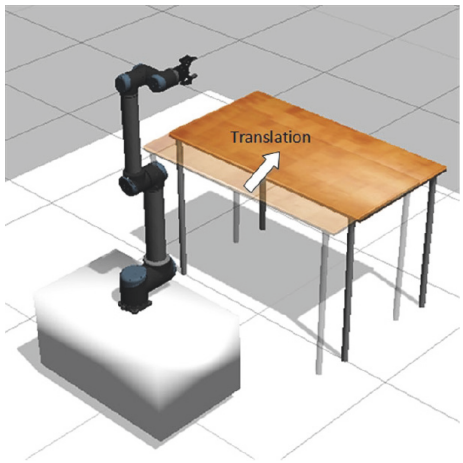
This section provides some simulations and real world experiments' results to support the method proposed in this paper. All experiments are conducted on an Intel Core i5-4590 3.3GHz personal computer. ROS and Gazebo are used to build a simulation platform. FCL is used to acquire the sample set of the collision region in robot configuration space and perform accurate collision query when the GMM-based collision query results are ambiguous. OMPL [27] (Open Motion Planning Library) provides a variety of sampling-based motion planning algorithms for comparison experiments.

4.1. Learning and Online Updating. In order to prove the effectiveness of the algorithm, the 6-DOF UR10 robot in Figure 5 is used for the simulation test. First, according to Algorithm 1, a six-dimensional Gaussian mixture model under the environment is trained. By alternately using the global EM iterations and binary EM iterations, the number of Gaussian components K gradually increases. When $K = 25$, the log-likelihood function $\mathscr{L}_k$ converges, meaning that the GMM has completed training and stopped adding new Gaussian components, as shown in Figure 7.

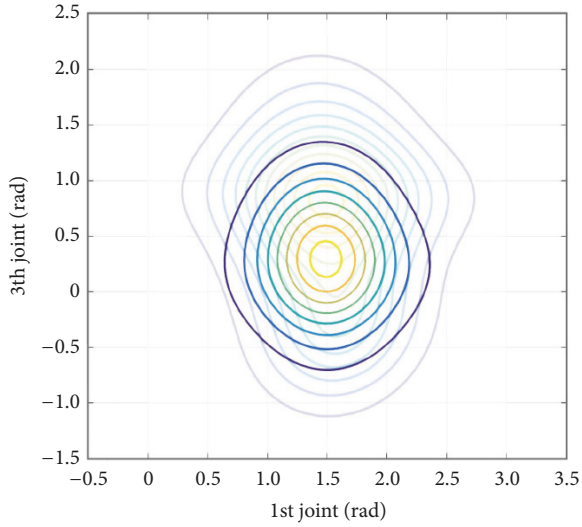
After the model training is completed, the incremental GMM's online updating capability is tested by changing the environment. As shown in Figure 8, the table in Figure 5 is rotated and translated to change the environment. Besides, We have also carried out simulations to change the environment through moving the robot. Algorithm 2 is used to update the GMM online. The new Gaussian components are added and trained according to the binary EM iterations from (7) to (11), and the basic EM algorithm from (3) to (5) is used to train the GMM with new components added. This will allow the GMM to fit the new collision sample set. After the training is completed, the Gaussian components that cannot help to fit the new collision sample set are eliminated to reduce the computational overhead of the algorithm. As shown in Figures 8(b), 8(d), 8(f), and 8(h), the projections of the GMM on the first joint and the third joint subspace change according to the changes of environment and robot base. Therefore, GMM has the ability to respond to the environmental changes online.

4.2. 6-DOF Robot Motion Planning. The 6-DOF UR10 robot is used to perform motion planning in the simulation environment of Figure 5. After the completion of GMM's incremental training, the false positive and false negative generalization error curves (Figure 6) are obtained using the cross-validation method in Section 3.3, and the decision boundaries δ_{col} and $\delta_{col-free}$ are determined accordingly. Finally, the GMM-based collision query strategy described above is applied to four sampling-based motion planning algorithms: RRT [3], RRT-Connect [28], BiEST [29], and KPIECE [30].

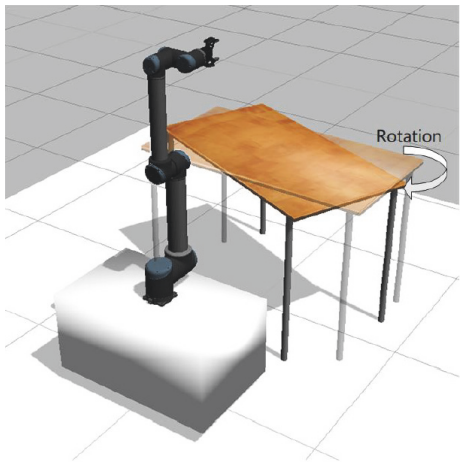
The initial state and the goal region of the planning problem are specified, and the above planners are used to solve the planning problem repeatedly to obtain the boxplots of the planning time (Figure 9). Besides, we provide the average planning time and the planning success rate, as shown in Table 1. The allowed planning time is set to 50s. It can be seen that the collision query strategy based on GMM is generally applicable to various sampling-based motion planning algorithms. Compared with the basic collision query methods based on forward kinematics and space geometry, the computational overhead is greatly reduced, and the overall motion planning efficiency is improved by 2-3 times.



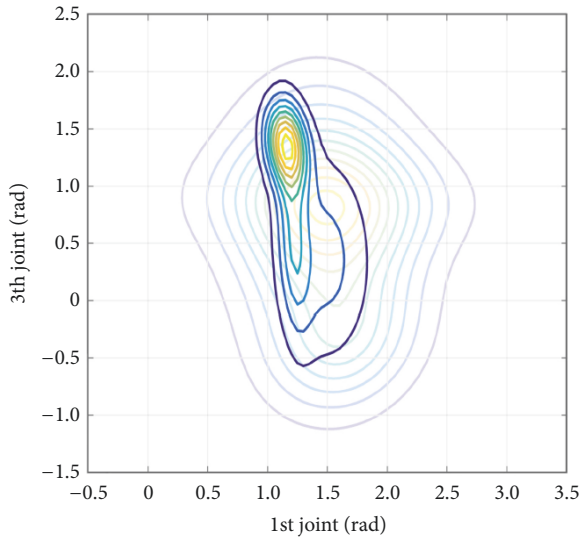
(a) Translating the table



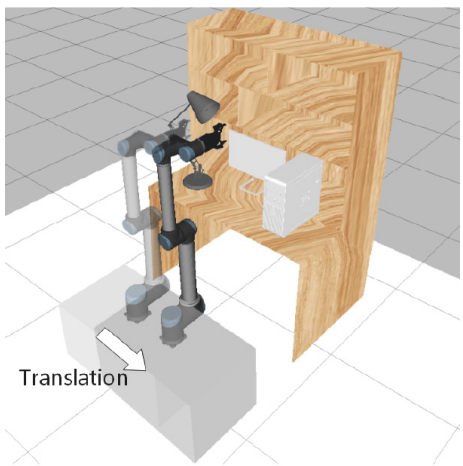
(b) Updating GMM for table' translation



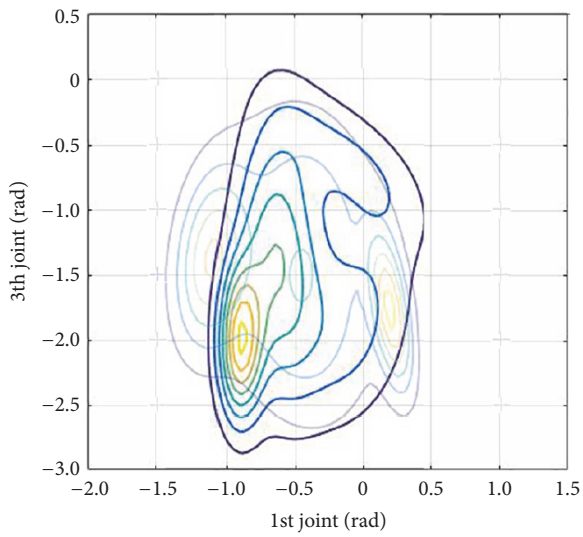
(c) Rotating the table



(d) Updating GMM for table' rotation

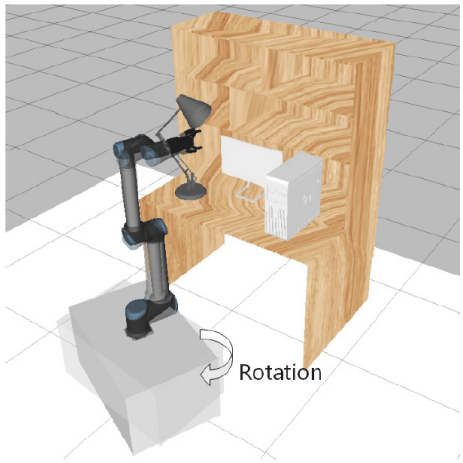


(e) Translating the robot

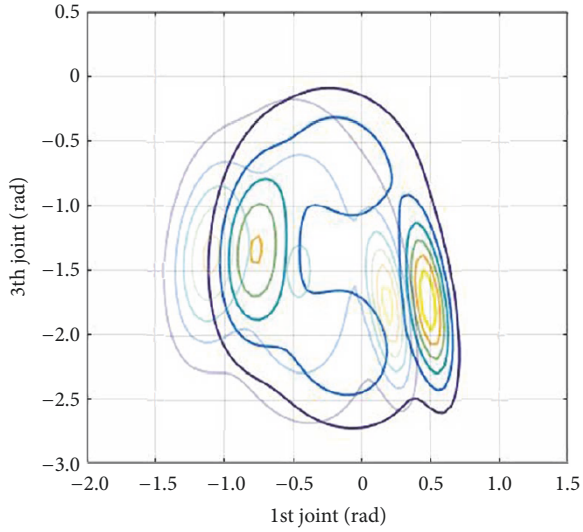


(f) Updating GMM for robot' translation

Figure 8: Continued.

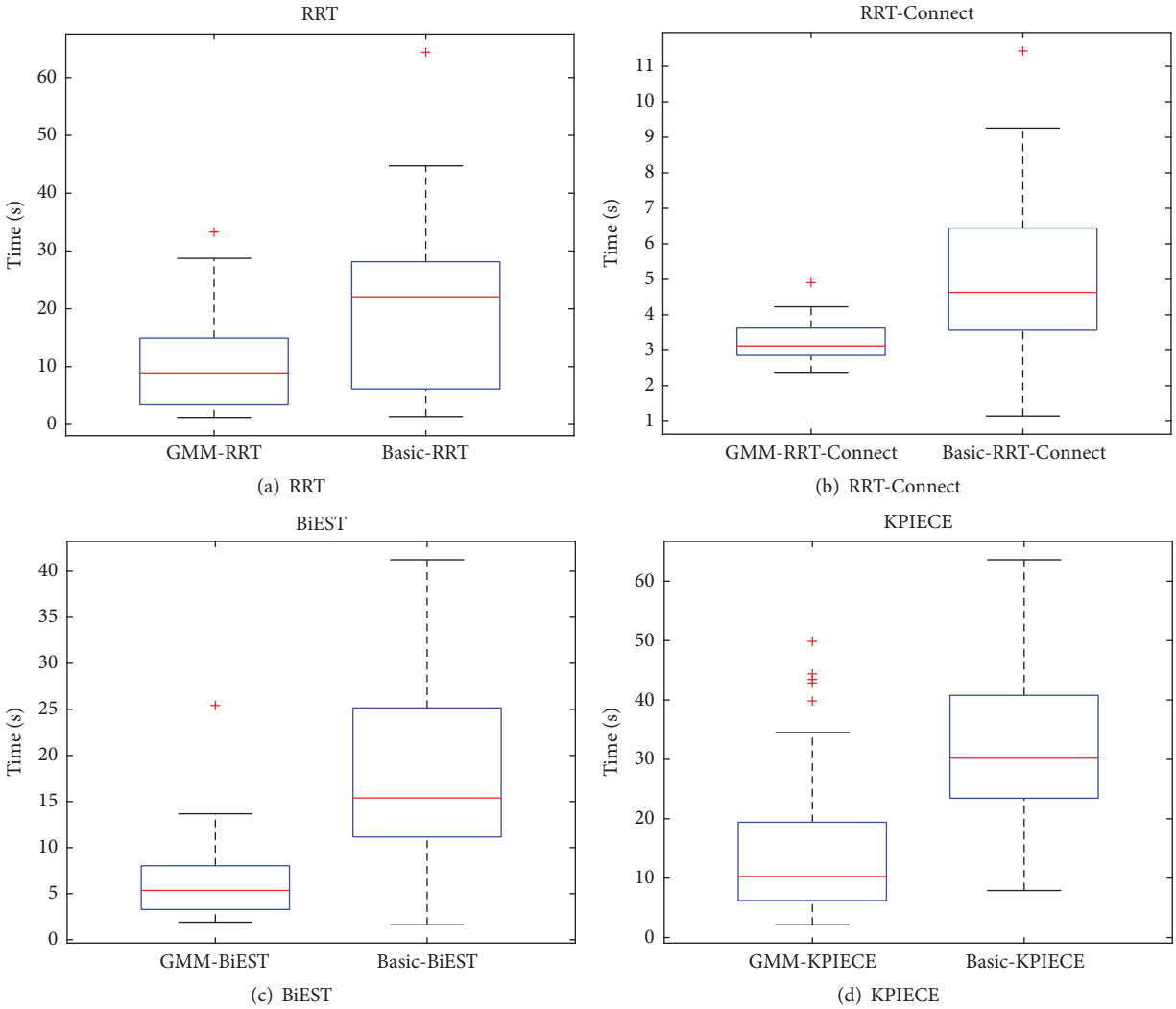


(g) Rotating the robot



(h) Updating GMM for robot' rotation

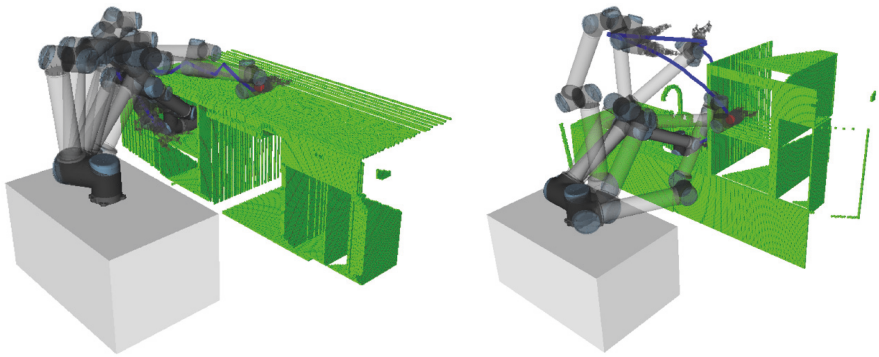
Figure 8: GMM's online updating.



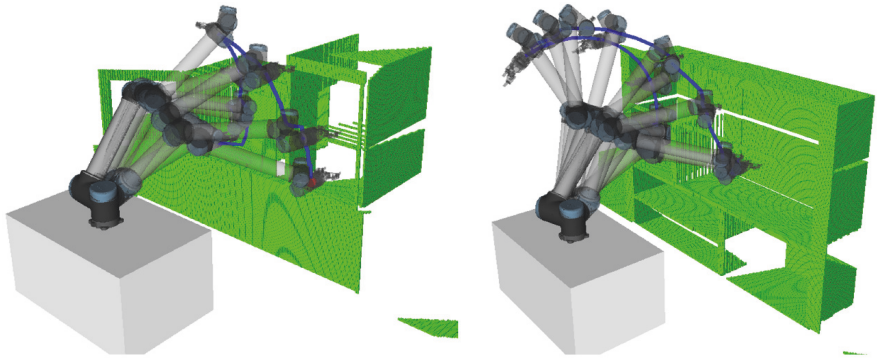
(a) RRT (b) RRT-Connect

(c) BiEST (d) KPIECE

Figure 9: The boxplots of various motion planning algorithms' planning times.



(a) Scene 1: desk (b) Scene 2: kitchen



(c) Scene 3: microwave oven (d) Scene 4: cabinet

Figure 10: A 6-DOF UR10 robot is used to pick and place a cup in various home environments, and the universality of the proposed method is verified. The green grid represents occupied space stored in the octree data structure, and the blue line represents the trajectory of the end effector when the robot performs the pick and place task.

Table 1: Planning performance of the motion planning algorithms.

Planning Methods	RRT		RRT-Connect		BiEST		KPIECE	
	Basic	GMM	Basic	GMM	Basic	GMM	Basic	GMM
Average Planning Time(s)	22.04	10.63	5.122	2.239	18.05	5.200	34.34	15.19
Successful Rate(%)	67.4	95.3	100	100	96.3	100	46.2	93.1

In order to further test the effectiveness of the proposed method in various environments, four common scenes (Figure 10) are built in a simulation environment. The RGB-D camera is used to acquire the point cloud data of each scene, and a three-dimensional octree map is created based on the OctoMap library [31]. The Greedy EM algorithm is used to train the GMMs of the robot configuration space in the four scenes, respectively, and the GMM-based and the basic collision query methods are applied to the above four kinds of motion planners, respectively. In the four planning scenes, the task of picking and placing a cup is accomplished by motion planning of a robot, repeatedly testing this planning task and obtaining the average planning time and successful rate with the GMM-based and basic collision query methods, as shown in Table 2. The allowed planning time is set to 50s.

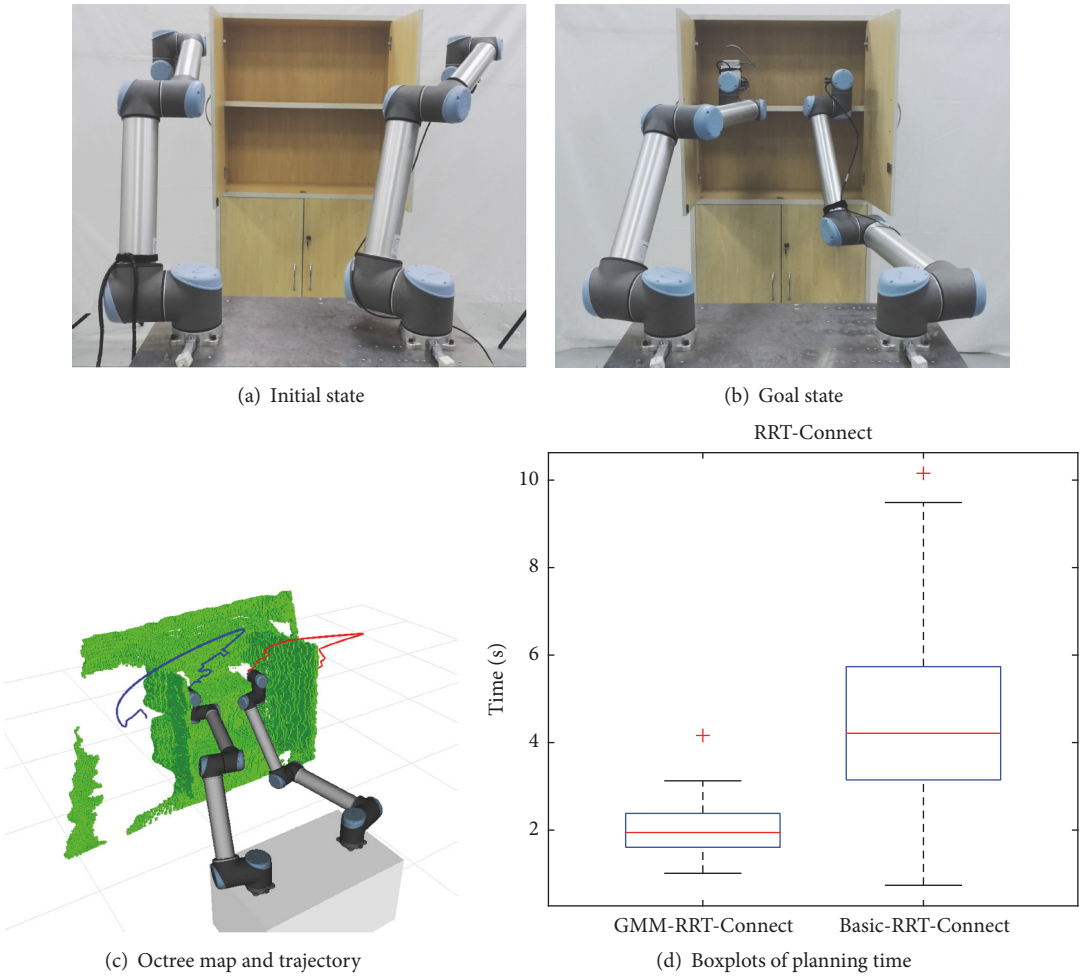
From Table 2, it can be seen that the actual performance of the proposed method varies due to different complexity of the planning tasks and number of Gaussian components, the latter of which is caused by different planning scenes. However, the average planning time of the GMM-based motion planning method is reduced by 2-4 times compared with the basic collision query method.

4.3. 12-DOF Robot Motion Planning. In order to verify the application effect of this algorithm in the actual scene, two UR10 robots and RealSense camera are used to build a 12-DOF dual-arm robot experimental platform, as shown in Figures 11(a) and 11(b). In order to prevent the robot obstruct camera's view, the "Eye in Hand" arrangement is adopted.

Due to the high planning dimensions of dual-arm robots, the successful rate under allowed planning time (50s) of most motion planning algorithms is too low, and it is difficult to obtain large amounts of data through multiple experiments. Therefore, only the RRT-Connect algorithm is used to perform multiple experiments, and the planning times based on the GMM method and the basic method are, respectively, obtained (Figure 11(d)). The success rate of both methods is 100%.

Table 2: Planning performance of the motion planning algorithms in four scenes.

Average Planning time(s)/Successful Rate(%)	RRT		RRT-Connect		BiEST		KPIECE	
	Basic	GMM	Basic	GMM	Basic	GMM	Basic	GMM
Scene 1	32.27/31.2	27.47/72.0	0.971/100	0.530/100	3.425/86.0	1.724/100	4.680/82.3	1.458/100
Scene 2	17.55/90.2	9.759/98.7	0.310/100	0.178/100	1.929/100	0.779/100	1.446/100	0.632/100
Scene 3	16.32/90.3	4.176/100	0.607/100	0.372/100	6.792/70.0	2.056/100	4.128/100	1.605/100
Scene 4	33.20/68.6	7.416/100	0.510/100	0.271/100	3.737/100	0.734/100	3.348/100	1.826/100



(a) Initial state (b) Goal state

(c) Octree map and trajectory (d) Boxplots of planning time

FIGURE 11: 12-DOF dual-arm robot experiment.(a), (b) The initial and goal state of the planning problem, respectively. (c) The environmental model represented by octree, and the lines represent the trajectory of end effectors. (d) The boxplots of planning time based on GMM and basic collision query method.

Experiments show that the method proposed in this paper is also applicable to higher-dimensional motion planning problems.

5. Conclusion

In order to improve the planning efficiency of the sampling-based motion planning algorithm, this paper studies the influence of the number of components on the fitting accuracy and proposes a self-adapting model training method, which achieves a rapid collision query in the robot high-dimensional configuration space. Using the machine learning method, the robot learns the collision query results generated during the previous motion planning instances, and the Gaussian mixture model is incrementally established to quickly estimate the new high-dimensional samples' collision probability. In order to eliminate the model fitting errors caused by environmental changes, the Greedy EM algorithm is used to adaptively select number of components of the GMM-based on the convergence of the log-likelihood function and achieved online response to the changes of external environment. This method is applied to several kinds of sampling-based motion planning algorithms. The simulations and real world experiments are performed on the 6-DOF and 12-DOF robots in various scenes. Compared with the basic methods, the planning time is greatly shortened. The effectiveness of the proposed algorithm is proved.

In the future work, we will continue to focus on improving the practical performance of this method in higher-dimensional robot motion planning problems, such as Humanoid robots and quadruped robots. The GMM will be used for guided sampling to further narrow the scope of the search algorithm.

Data Availability

All data generated or analyzed during this study are included in this published article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. U1713222, No. 61773139, and No. 61473015), Natural Science Foundation of Heilongjiang Province, China (Grant No. F2015008), and Shenzhen Peacock Plan (No. KQTD2016112515134654).

References

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[2] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, vol. 1994, Unknown Publisher, 1994.

[3] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[4] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[5] G. v. Bergen, "Efficient Collision Detection of Complex Deformable Models using AABB Trees," *Journal of Graphics Tools*, vol. 2, no. 4, pp. 1–13, 1997.

[6] G. Sánchez and J.-C. Latombe, "On delaying collision checking in PRM planning: Application to multi-robot coordination," *International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[7] S. Dalibard and J.-P. Laumond, "Linear dimensionality reduction in random motion planning," *International Journal of Robotics Research*, vol. 30, no. 12, pp. 1461–1476, 2011.

[8] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pp. 4420–4426, Taiwan, September 2003.

[9] L. Zhang and D. Manocha, "An efficient retraction-based RRT planner," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation, ICRA 2008*, pp. 3743–3750, USA, May 2008.

[10] M. Saha, J.-C. Latombe, Y.-C. Chang, and F. Prinz, "Finding narrow passages with probabilistic roadmaps: The small-step retraction method," *Autonomous Robots*, vol. 19, no. 3, pp. 301–319, 2005.

[11] B. Burns and O. Brock, "Information theoretic construction of probabilistic roadmaps," in *Proceedings of the Intelligent Robots and Systems, 2003.(IROS 2003) 2003 IEEE/RSJ International Conference*, vol. 1, pp. 650–655, 2003.

[12] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, pp. 2646–2652, Germany, October 2015.

[13] O. Arslan and P. Tsiotras, *The role of vertex consistency in sampling-based algorithms for optimal motion planning*, 2012, arXiv preprint arXiv:1204.6453.

[14] J. Bialkowski, M. Otte, and E. Frazzoli, "Free-configuration biased sampling for motion planning," in *Proceedings of the 2013 26th IEEE/RSJ International Conference on Intelligent Robots and Systems: New Horizon, IROS 2013*, pp. 1272–1279, Japan, November 2013.

[15] B. Burns and O. Brock, *Model-based motion planning*, vol. 51, Computer Science Department Faculty Publication Series, 2004.

[16] C. Yang, X. Wang, L. Cheng, and H. Ma, "Neural-Learning-Based Telerobot Control with Guaranteed Performance," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3148–3159, 2017.

[17] J. Pan and D. Manocha, "Fast and robust motion planning with noisy data using machine learning," in *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[18] K. Yang, S. Keat Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[19] J. Huh and D. D. Lee, "Learning high-dimensional Mixture Models for fast collision detection in Rapidly-Exploring Random Trees," in *Proceedings of the Robotics and Automation (ICRA), 2016 IEEE International Conference*, pp. 63–69, 2016.

[20] B. Ichter, J. Harrison, and M. Pavone, *Learning sampling distributions for robot motion planning*, 2017, arXiv preprint arXiv:1709.05448.

[21] C. Yang, G. Peng, Y. Li, R. Cui, L. Cheng, and Z. Li, "Neural networks enhanced adaptive admittance control of optimized robot-environment interaction," *IEEE Transactions on Cybernetics*, vol. 99, p. 12, 2018.

[22] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-Graphs: Bootstrapping Planning with Experience Graphs," *Robotics: Science and Systems*, vol. 5, no. 1, 2012.

[23] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, "Robot Learning System Based on Adaptive Neural Control and Dynamic Movement Primitives," in *Proceedings of the EEE transactions on neural networks and learning systems*, vol. 99, pp. 1–11, 2018.

[24] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," *Neural Processing Letters*, vol. 15, no. 1, pp. 77–87, 2002.

[25] J. Q. Li and A. R. Barron, "Mixture density estimation," *Advances in neural infor- mation processing systems*, pp. 279–285, 2000.

[26] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," pp. 3859–3866.

[27] I. A. Şucan, M. Moll, and L. Kavraki, "The open motion planning library," *IEEE Robotics and Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[28] J. J. Kuffner Jr. and S. M. la Valle, "RRT-connect: an efficient approach to single-query path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 995–1001, April 2000.

[29] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation, ICRA. Part 3 (of 4)*, pp. 2719–2726, April 1997.

[30] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII*, pp. 449–464, Springer, Berlin, Heidelberg, 2009.

[31] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.