





Article

An Efficient Motion Planning Method with a Lazy Demonstration Graph for Repetitive Pick-and-Place

Guoyu Zuo ^{1,2} , Mi Li ^{1,2}, Jianjun Yu ^{1,2}, Chun Wu ^{1,2} and Gao Huang ^{1,2,3,*} ¹ Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China² Beijing Key Laboratory of Computing Intelligence and Intelligent Systems, Beijing 100124, China³ Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing Institute of Technology, Beijing 100081, China

* Correspondence: huanggao@bjut.edu.cn

Abstract: Robotic systems frequently need to plan consecutive similar manipulation in some scenarios (e.g., pick-and-place tasks), leading to similar motion plans. Moreover, the workspace of a robot changes with the difference in operation actions, which affects subsequent tasks. Therefore, it is significant to reuse information from previous solutions for new motion planning instances to adapt to workplace changes. This paper proposes the Lazy Demonstration Graph (LDG) planner, a novel motion planner that exploits successful and high-quality planning cases as prior knowledge. In addition, a Gaussian Mixture Model (GMM) is established by learning the distribution of samples in the planning cases. Through the trained GMM, more samples are placed in a promising location related to the planning tasks for achieving the purpose of adaptive sampling. This adaptive sampling strategy is combined with the Lazy Probabilistic Roadmap (LazyPRM) algorithm; in the subsequent planning tasks, this paper uses the multi-query property of a road map to solve motion planning problems without planning from scratch. The lazy collision detection of the LazyPRM algorithm helps overcome changes in the workplace by searching candidate paths. Our method also improves the quality and success rate of the path planning of LazyPRM. Compared with other state-of-the-art motion planning algorithms, our method achieved better performance in the planning time and path quality. In the repetitive motion planning experiment of the manipulator for pick-and-place tasks, we designed two different experimental scenarios in the simulation environment. The physical experiments are also carried out in AUBO-i5 robot arm. Accordingly, the experimental results verified our method's validity and robustness.

Keywords: manipulation planning; motion and path planning; learning sampling distribution; autonomous robot



Citation: Zuo, G.; Li, M.; Yu, J.; Wu, C.; Huang, G. An Efficient Motion Planning Method with a Lazy Demonstration Graph for Repetitive Pick-and-Place. *Biomimetics* **2022**, *7*, 210. <https://doi.org/10.3390/biomimetics7040210>

Academic Editors: Shuai Li and Jinyou Shao

Received: 4 October 2022

Accepted: 16 November 2022

Published: 21 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Robot manipulators are widely used for performing continuous manipulation tasks such as parts assembly and material sorting in fields such as manufacturing [1]. In structured scenarios, robotic arms often only need to repeat the demonstration trajectory to perform repetitive tasks. However, in a semi-structured scene, the appearance of new obstacles will change the workspace, so that the denomination trajectory may collide with obstacles and affect the execution of operation tasks [2,3]. Furthermore, in some scenarios, the robotic arm often needs to perform multiple picking and placing tasks, the nature of repetitive pick-and-place tasks suggests that the solutions for all motion planning instances are similar to some extent [4]. In this case, we can collect successful planning cases during the offline phase, followed by reusing the knowledge online [5]. Importantly, two aspects should be considered in this case: (1) how to characterize and generalize the prior knowledge and use it to solve new motion planning problems; (2) how to ensure the stability of the motion planning process when the prior knowledge becomes invalid with changes of the environment.

In the past few decades, several methods have been proposed to solve motion planning problems. Among them, intelligent optimization algorithms, which iteratively solve trajectories that satisfy the task requirements, are popular. Tan et al. proposed a path planning method based on an improved ant colony algorithm [6], which improved the efficiency of planning by imperatively selecting the initial parameters of the ant colony algorithm through a particle swarm optimization algorithm. Evolutionary algorithms are used in robot trajectory planning to obtain a safe path that satisfies the kinematic constraints by iteratively optimizing the travel time and actuator effort [7]. However, intelligent optimization algorithms tend to fall into local extremes easily and converge slowly, which is determined by the nature of the optimization algorithm itself, while our method samples directly in the joint space, which is faster for planning and less likely to fall into local solutions.

Sampling-based motion planning (SBMP) algorithms can effectively solve motion planning problems without modeling the environment [8]; this type of method has improved robustness, therefore gradually becoming mainstream in this field. The Rapidly-exploring Random Tree (RRT) [9] uses a tree data structure to obtain connectivity information in the configuration space (C-space) of the robotic arm by random sampling. The Probabilistic Roadmap (PRM) [10] builds a roadmap for multi-query planning in C-space, so the roadmap can be reused for subsequent planning. However, in a semi-structured environment, before the roadmap is reused, it is necessary to perform collision detection on the nodes and edges of the roadmap again, otherwise the planned path may collide with obstacles. Some asymptotically optimal planning algorithms aim to minimize the cost function, such as RRT* and PRM* [11]. The path converges to the optimum with a sufficient number of samples. However, such algorithms often take a long time to converge because of the random sampling nature. Xu et al. proposed an RRTConnect algorithm based on a sparse expansion strategy and dead point saved strategy [12], which can effectively reduce the number of collision detections and accelerate the convergence speed. In contrast, our approach leverages past planning experience and can further reduce the likelihood of redundant sampling.

In light of the above, how to use past experience to speed up the solution of motion planning problems has attracted a lot of attention in the robot motion planning community. Roadmap-based approaches construct a graph data structure in configuration space to store connectivity information that has been collected in previous planning queries, while in the repetitive manipulation tasks of the robot arm, the position of the obstacle changes in workspace. An improved version of the original PRM algorithm, LazyPRM [13], was proposed to adapt to the changing environment by delaying collision detection. Rosell et al. collected demonstration data as a priori knowledge and used Principal Component Analysis (PCA) to learn the demonstration data to improve the path planning efficiency of the robotic arm during the grasping operation [14]. Similarly, PCA is used to learn demonstration data and combined with the RRT algorithm to achieve fast biased sampling [15], but this paper has only validated this in a 2D environment. Roveda et al. proposed an algorithm called HMM that is able to select reliable task trajectories from demonstration data [16], but when the workspace changes, some of the trajectories might conflict with the environment and thus planning fails. The road map constructed by GMM mixture distribution can quickly plan the path of grasping objects in a narrow passage [17]. Furthermore, the adaptive sampling method is combined with LazyPRM to solve the continuous operation motion planning of the manipulator. The past trajectories can be stored in the path library, which invoke a suitable path to solve the new motion planning queries. In the Lightning framework [18], the authors selected previous paths in a probabilistic way and the RRTConnect algorithm to modulate it. Similarly, the authors of [19] used previous experience to create a sparse roadmap, and then found valid path by A* [20], a shortest path search algorithm. On this basis, this method adopts a guaranteed mechanism so that when the A* search algorithm cannot find a feasible path, the RRTConnect algorithm can be used to repair from the disconnected states. However, these path library-based methods

occupy a large amount of storage space, and in a semi-structured environment, some of the stored paths will fail due to changes.

Learning-based approaches are often used to analyze the potential characteristics of a task from a set of task-related expert demonstration trajectories by deep reinforcement learning [21], which are used to guide the generation of task-related motion plans. The conditional variational auto-encoder, a generative model, allows for coding the environmental features and sampling from the latent space in complex environments [22]. Motion planning networks [23,24] are a novel class of planners that attempt to solve motion planning problems by deep learning. These methods first encode the environmental point cloud data and then exploit neural networks to fit the expert's demonstration trajectories, but these methods depend on the quality of the dataset and are subject to error accumulation when the neural networks are forward-propagated to generate samples. Reinforcement learning approaches treat the motion planning problem as a Markov process [25], where the intelligence learns the planning strategy through continuous trial and error, yet the manipulation skills learned by the robot in the simulation environment are difficult to deploy on real robots. Deep reinforcement learning-based planning algorithms have lots of model parameters and it is difficult to deploy on robotic arms.

This paper proposes a new motion planning algorithm, the lazy demonstration graph (LDG), which solves the repetitive motion planning of a manipulator. First, the high-quality trajectory solved by the expert planner is used as a priori knowledge to train the GMM, a generative model in machine learning, and the trained distribution is used to realize adaptive sampling so as to sample in the specific area containing the optimal paths, planning speed, and path quality. Second, this sampling method is combined with the LazyPRM algorithm. The introduction of lazy collision detection does not need to evaluate the overall validity of the graph to accelerate the construction of a roadmap. More importantly, it can adapt to environmental changes by querying candidate paths. Finally, we analyzed and modeled the continuous operation task of the manipulator, designed two different manipulator operation scenarios in the simulation environment, and achieved the continuous motion planning of the manipulator for the pick-and-place tasks by reusing the constructed LDG, without planning from scratch. In summary, the contributions of this paper are as follows:

1. In this paper, we propose an experience guided sampling method. The advantage of this method is that the distribution of demonstration data is learned via GMM, which are used to generate samples at the task-related location, so as to improve the sampling efficiency.
2. A new algorithm called Lazy Demonstration Graph (LDG), combined the above-mentioned sampling method with LazyPRM algorithm is proposed. The advantages of new algorithm are: First, the multi query property of graph is used to solve the continuous motion planning problem of the manipulator. Second, the introduction of lazy collision detection can improve planning efficiency and allows for adaptation to changes in the workspace to some extent.
3. The repetitive pick-and-place tasks are modeled, and the continuous planning problem of the robotic arm can be solved effectively by the proposed method and have been verified in the simulation experiments, at the same time, a physical experiment was designed to verify the effectiveness of our method.

The rest of this paper is organized as follows. Section 2 introduces the design process of our method in detail. Section 3 shows the results of the experiment. Section 4 provides a summary and future research plans.

2. Materials and Methods

2.1. Problem Definition

In this section, we focus on the definition of repetitive motion planning problems for robotic arms in semi-structured scenarios. Consider a d Degrees Of Freedom (DOF) robot, where the configuration space (C-space) is the d -dimensional space consisting of all

possible configurations of the robot, and each possible configuration is a point in C-space, which represents the d joint angles of the robot.

Let $X \subseteq \mathbb{R}^d$ be the C-space. We assume that X_{obs} is the region occupied by obstacles in C-space; therefore, $X_{free} = X \setminus X_{obs}$ represents the collision-free region. Let x_{start} denote the initial configuration in the collision-free C-space, and x_{goal} is the target configuration. A query for motion planning is to find a continuous trajectory $\tau : [0, 1] \rightarrow X_{free}$ for a given x_{start}, x_{goal} such that each point of the trajectory lies in the collision-free region, where $\tau(0) = x_{start}, \tau(1) = x_{goal}$ and they represent the start and end of the trajectory, respectively.

Robotic arms are usually required to move objects to specific locations in the workspace, for example, shelf stacking (see Figure 1). We assume that an instance of the robotic arm manipulation task T is represented by a set $T = \{t_1, t_2, \dots, t_i\}$, where $t_i(o_i, p_{start}, p_{goal})$ is one of the subtasks, which consists of the manipulated object o_i , start pose of the robot arm end-effector p_{start} and target pose p_{goal} . Let function f_x denote the mapping relationship from the pose of robot arm end-effector to a joint configuration of the robotic arm such that

$$\begin{aligned} f_x(p_{start}) &= x_{start} \\ f_x(p_{goal}) &= x_{goal} \end{aligned} \quad (1)$$

where f_x can be derived from the inverse kinematics of the robot. On this basis, each subtask $t_i(o_i, x_{start}, p_{goal})$ corresponds to a new motion planning query that leads to changes in the objects' position and in the underlying C-space of the robot indirectly. Furthermore, the repetitive manipulation task T is equivalent to multiple queries of motion planning for a finite time in semi-structured scenarios.

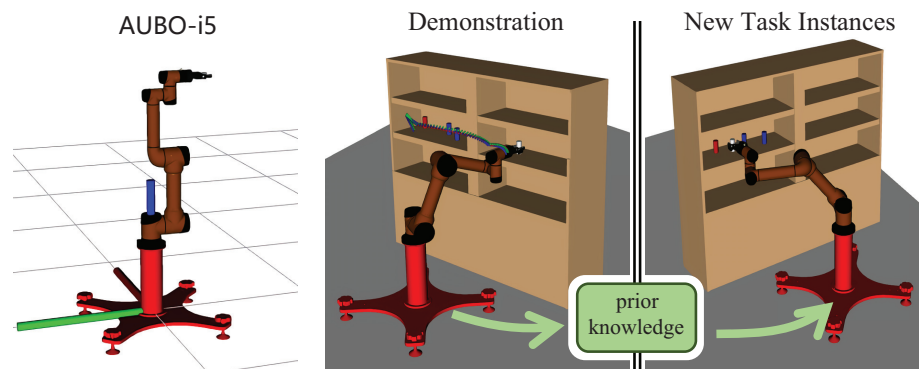


Figure 1. Our planner can generalize and leverage the high-quality paths solved by the expert planner to solve new task instances. The expert planner plans the paths by searching the manipulation tasks and collecting and using these paths as priori knowledge to solve new motion planning problems quickly and consistently to grab all the cylinders on the left and place them on the right side of the shelf, thus avoiding collisions.

2.2. Learning Sampling Distribution

One of the main ideas of the LDG is to introduce the previously solved path information into the sampling distribution to effectively solve similar task instances. This distribution focuses on placing samples in parts of the configuration space that are more relevant to the task to reduce useless search. Limiting the sampling space not only improves the planning efficiency of the planner but also overcomes the problem that it is difficult to place samples in a complex environment.

Owing to the complex configuration space topology of the robotic arm, we use a Gaussian Mixture Model (GMM) to model the distribution of collision-free trajectories in the robot C-space, using a probabilistic approach to extract the key configurations of the demonstration trajectories. We iteratively estimate the parameters of the GMM by an expectation maximization (EM) algorithm [26] that maximizes the likelihood that past solution configurations are sampled from this distribution [27]. To speed up the training

process as well as avoid local optimization, we initialize the parameters of the EM algorithm using the bisecting K-Means [28] clustering algorithm, while the k values are selected by the Bayesian Information Criterion (BIC) [29]. Sampling from the trained distribution can then create samples at task-relevant locations.

We use expert's demonstration trajectories as priori experience. These high-quality solutions can be obtained by asymptotically optimal motion planners for specific operation tasks. We discretize these trajectories to obtain the key configurations, which contain most of the information of the previous solutions, and these key configurations constitute the training set q :

$$q = [q_1 \dots q_i \dots q_n]^T \quad (2)$$

where q_i represents the path points contained in the i th demonstration path. We use this dataset to train a GMM, which can be considered as a superposition of multiple Gaussian models:

$$p(q) = \sum_{k=1}^K \pi_k N(q; \mu_k, \sigma_k) \quad (3)$$

where $p(q)$ represents the probability density function, K is the number of Gaussian components, and each Gaussian distribution N is composed of the mean μ_k and covariance matrix σ_k , and π_k is the weight of the Gaussian distribution.

Parameter estimation for the Gaussian mixture distribution is more complex than that for the Gaussian distribution. Thus, we employ the expectation and maximization steps to make the distribution fit the data in the dataset. In the expectation step, we introduce the hidden variable $\gamma_{i,k}$, which represents the probability that the i th data come from the k th Gaussian component.

$$\gamma_{i,k} = \frac{\pi_k N(q_i; \mu_k, \sigma_k)}{\sum_{k=1}^K \pi_k N(q_i; \mu_k, \sigma_k)} \quad (4)$$

In the maximization step, we use the updated probability value $\gamma_{i,k}$ to update the parameters of the GMM. First, we calculate the new mean of each Gaussian component:

$$\mu_k = \frac{\sum_{i=1}^N \gamma_{i,k} q_i}{\sum_{i=1}^N \gamma_{i,k}} \quad (5)$$

With the updated means, we can update the covariance matrix for each component:

$$\sigma_k = \frac{\sum_{i=1}^N \gamma_{i,k} (q_i - \mu_k)(q_i - \mu_k)^T}{\sum_{i=1}^N \gamma_{i,k}} \quad (6)$$

Finally, new weights can be calculated for each component:

$$\pi_k = \frac{\sum_{i=1}^N \gamma_{i,k}}{N} \quad (7)$$

We terminate the iteration when the log-likelihood function reaches a local optimum:

$$\ln p(Q) = \sum_{i=1}^N \ln \left(\sum_{k=1}^K \gamma_{i,k} N(q_i; \mu_k, \sigma_k) \right) \quad (8)$$

Once the model converges, we can use the distribution to generate samples, and the complete process is shown in Figure 2.

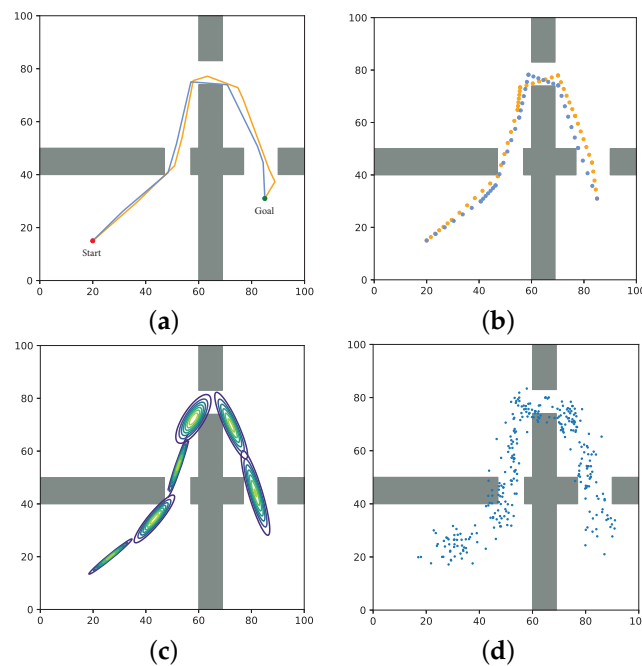


Figure 2. Overall process of the learning sample distribution. Collecting successful and high-quality planning cases (a), discretizing the trajectory, and constructing the training set (b), training the GMM with the collected data until the model converges (c), using the trained model for generating samples in task-related locations (d).

2.3. Lazy Demonstration Graph

In the previous section, we characterized the prior knowledge obtained by expert demonstrations with a GMM, and in this section, we generate sampling points and construct a probabilistic roadmap by Gaussian mixture distributions; that is, we apply previous solutions to solve specific motion planning problems. We first describe how the sampling-based motion planning algorithm constructs the roadmap, taking LazyPRM as an example, which is a classical roadmap-based motion planning algorithm on which our method is based.

As shown in Algorithm 1, the LazyPRM algorithm can be divided into two phases: constructing graph $G = (V, E) \subseteq X$ and querying. In the graph construction phase, LazyPRM iteratively samples a random configuration x_{rand} without checking its validity by collision detection. This random configuration is added directly to the set of vertices V (line 4 and 5). For each sample x_{rand} , a nearest neighbor search is performed (line 6–10); in this step, we usually have to set the value γ for distance constraint between two configurations. If the configuration x in the set of vertices V and random configuration x_{rand} meet the distance constraint ($\|x - x_{rand}\| < \gamma$), then the configuration x is considered as the nearest neighbor x_{near} of the random configuration x_{rand} . The random configuration x_{rand} is also directly connected to its nearest neighbor x_{near} as edge in the graph without collision detection, and this step often called local planning (line 9). When the graph is constructed, the shortest path can be found by the graph-search-based algorithm A* or Dijkstra's algorithm (line 13). Because the lazy data structure allows for invalid edges, the path obtained by the search algorithm may collide with obstacles; therefore, collision detection is needed for the solved path. If a collision occurs, the candidate path is searched until a collision-free effective path is obtained (line 14).

Algorithm 1: Lazy PRM

Input: x_{init}, x_{goal}
Output: *SolutionPath* τ_i^*

```

1  $V \leftarrow \{x_{init}, x_{goal}\}$ 
2  $E \leftarrow \emptyset$ 
3 while  $n < N$  do
4    $x_{rand} \leftarrow \text{Sample}()$ 
5    $V \leftarrow x_{rand}$ 
6   foreach  $x \in V$  do
7     if  $\|x - x_{rand}\| < \gamma$  then
8        $x_{near} \leftarrow x$ 
9        $E \leftarrow (x_{near}, x_{rand})$ 
10    end
11  end
12 end
13  $\tau \leftarrow \text{SearchPath}(G, x_{init}, x_{goal})$ 
14  $\tau_i^* \leftarrow \text{CheckValid}(\tau_i)$  // Collision Detection
15 return SolutionPath  $\tau_i^*$ 

```

In the offline phase, we first collect the high-quality trajectories related to the operation task, then discrete the trajectories to form the training set, and train the GMM until convergence, thus the distribution $GMM_Sample()$ can be used for sampling in task-related areas. The overall flow of the LDG is shown in Algorithm 2. The algorithm is divided into two phases, as in most roadmap-based algorithms. First, we construct an undirected graph structure through cyclic sampling, and then use it to query collision-free paths. The specific operation task T , the number of initial samples N and the distribution $GMM_Sample()$ are used as inputs to the algorithm. We first initialize the node set V and edge set E of the graph G . Consistent with what was performed in these papers, in order to balance exploration with exploitation, we set a threshold value t (line 3) to choose whether to use a Gaussian mixture distribution or uniform distribution to generate samples in a probabilistic way. Lines 4–10 represent the cyclic sampling phase, and when the random number k lies between 0 and t , we use the trained Gaussian mixture distribution to generate a sample by first selecting a Gaussian component and then drawing samples from this Gaussian distribution, with each Gaussian component being selected with a probability proportional to its probability of being in the mixture distribution. Accordingly, the samples are placed in a promising location, reducing useless exploration; otherwise, the sample is generated through a uniform distribution to achieve the purpose of exploring the C-space. Because we introduce the lazy collision detection strategy, we do not make a judgment here on whether the sample x_{rand} collides with the environment, but add it directly to the node set V (line 11). As in Algorithm 1, we use nearest the neighbor search to obtain the nearest neighbor node x_{near} of x_{rand} and construct connections between (x_{near}, x_{rand}) via a local planner, which can connect two nodes in a straight line to form an edge in the graph in lines 12–17.

Once the undirected graph data structure G is constructed, multiple queries can be performed by the persistent graph structure to solve continuous motion planning problems, as shown in Figure 3. The received task is processed in lines 20–24, and each subtask t_i corresponds to a request of motion planning. The function $\text{SearchPath}(\cdot)$ is used to process each planning request, and it contains three parameters. Here, p_{start} is the current pose of the end effort, p_{goal} is the target pose of the end effort, and function $f_x(\cdot)$ represents the mapping relationship from the end effort of robotic arm to the joint configuration, which is used to determine the joint configuration of the arm for object manipulation, the function $f_x(\cdot)$ can in turn be deduced according to the inverse kinematics of the manipulator. In this way, the path search function takes a pair of robotic arm configurations

as its input and searches for the shortest collision-free path by the graph G , with the help of priori knowledge. The graph search algorithm we use is the A* algorithm based on depth-first search.

Algorithm 2: Lazy Demonstration Graph

```

// Tasks, Maximum number of samples, Gaussian mixed model sampler
Input:  $T, N, GMM\_Sample()$ 
Output: SolutionPath  $\tau_i^*$ 
1  $V \leftarrow \emptyset$ 
2  $E \leftarrow \emptyset$ 
// Balance exploration and exploitation
3  $t \leftarrow ThresholdValue(0, 1)$ 
4 while  $n < N$  do
5    $k \leftarrow RandomValue(0, 1)$ 
6   if  $0 \leq k \leq t$  then
7      $x_{rand} \leftarrow GMM\_Sample()$ 
8   else
9      $x_{rand} \leftarrow Uniform\_Sample()$ 
10  end
11   $V \leftarrow x_{rand}$ 
12  foreach  $x \in V$  do
13    if  $\|x - x_{rand}\| < \gamma$  then
14       $x_{near} \leftarrow x$ 
15       $E \leftarrow (x_{near}, x_{rand})$ 
16    end
17  end
18 end
19  $G \leftarrow (V, E)$  // Lazy Demnostration Graph
20 foreach Subtask  $t_i \in T$  do
21    $\tau_i \leftarrow SearchPath(G, f_x(p_{start}), f_x(p_{goal}))$ 
22    $\tau_i^* \leftarrow CheckValid(\tau_i)$  // Collision Detection
23    $Rewire(\sigma_i^*)$  // Path Smooth
24   return  $\tau_i^*$ 
25 end

```

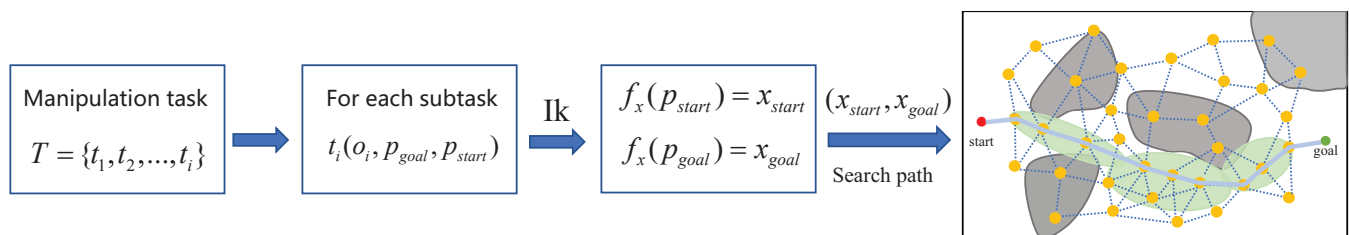


Figure 3. Overall process of solving the repetitive motion planning problem of robotic arm for manipulation tasks using LDG. The manipulation task T contains multiple pick-and-place subtasks and for each subtask t_i , the robotic arm is required to move the object from one position to another. the starting and target configurations of the arm can be obtained by inverse kinematics (IK) and then the shortest path is searched using the constructed LDG. the lazy data structure of LDG is kept in each task, which allows invalidating edges, it helps to improve the composition speed on the one hand and adapt the changes of the environment. In this figure, the green ellipse represents Gaussian mixture distribution obtained from prior knowledge training, which is used to place the samples in more promising locations.

Some valid nodes and edges of the graph constructed in C-space are invalidated because the manipulated object is attached to the manipulator. This results in the possibility of a collision between the manipulator and the environment increasing, as well as the position of object changing due to the operation of the robot. Therefore, in the construction phase of the graph, judging the validity of the nodes and edges of the graph is unnecessary, but collision detection on the searched paths must be performed σ_i through the function *CheckValid*(\cdot) (line 21). The collision detection function is implemented by the Flexible Collision Library (FCL); FCL is an open source project that includes various techniques for efficient collision detection and proximity computation [30]. If there is a collision on the path, the colliding nodes and edges are moved out of consideration, and then the candidate path σ_i^* is obtained by graph search. In this way, a lot of computation time can be saved without evaluating the overall effectiveness of nodes and edges in the graph, and we can also ensure the normal progress of subsequent tasks. Because the local planner connects the two nodes in a straight line, it leads to the redundancy of path points. Thus, a function *Rewire*(\cdot)—is introduced to remove redundant nodes in the path by evaluating whether the linear trajectories of two discontinuous nodes in the connection path have no collision (line 22). Similar to the reselection parent node algorithm used in RRT*, the lightweight implementation of this algorithm has little processing overhead, so it can be used without significantly increasing the path generation time, as shown in Figure 4.

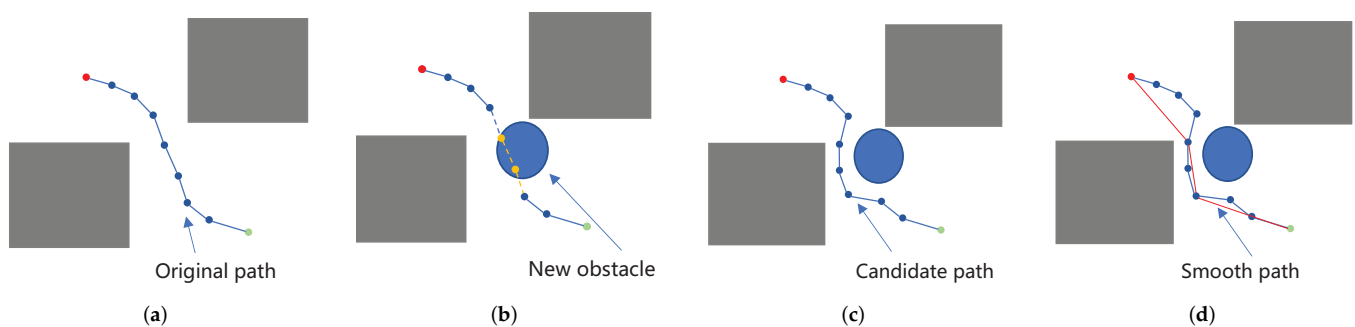


Figure 4. Given the start and goal points (shown in green and red dot), a path can be found through the graph search algorithm in a two-dimensional (2D) environment (a). When a new obstacle (blue ellipse) appears in the environment, the nodes and edges of the constructed graph structure will partially invalid (shown in yellow) (b). The planning problem is completed by searching the candidate path (ignoring the invalid nodes and edges when querying and searching path using other nodes and edges in the graph) (c). The path is further optimized by removing redundant states from the path (shown in red line) (d).

3. Experiments

To evaluate the effectiveness of our method, we designed a three-part experiment. First, a path planning test was conducted on a 2D grid map with a translational-only point robot, which illustrates the creation and uses of the LDG in a more visual way. Subsequent experiments were deployed on the ROS and MoveIt! In the simulation environment, two scenes of shelf stacking and picking from bin were built. Continuous pick-and-place manipulation experiments with a 6 DOF AUBO-i5 robotic arm showed that our method had a certain degree of efficiency and robustness. Finally, we use MoveIt! in ROS to plan the path of the physical manipulator, so as to realize the task of picking and placing. All tests were carried out on an Intel Core i5 with 2.40-GHz cores and 16 GB of RAM running Ubuntu 16.04.

3.1. 2D Gridworld

In this part of the experiment, we compared our method with the LazyPRM, RRT*, RRTConnect [31], and BiRRT* [32] algorithms. The LazyPRM algorithm constructs the road map with uniform sampling. The RRTConnect algorithm is the greedy variant of RRT,

which is rapidly extended by two trees, and has great speed advantage in solving motion planning problems. BiRRT* is the state-of-the-art single-query algorithm, which inherits the asymptotic optimization property of the RRT* algorithm and has a faster expansion speed. Our method sets the same maximum number of samples to 470 as in LazyPRM, and the connection distance of the nearest neighbor node is set to 45. We used BiRRT* as our expert tutorial planner to tutorialize the planning tasks and used the discrete path points as training data for the GMM. The number of iterations of the EM algorithm is set to 100, and the optimal number of components of the GMM is 12 according to the BIC criterion. probability of generating samples using GMM was set to 0.3. All code of this part of the experiment was implemented in matlab2017.

Figure 5 shows the planning visualization of our proposed method compared with three representative methods, i.e., LazyPRM, RRTConnect, and BiRRT*, in two different 2D scenes. Each 2D scene has a length and width of 500, and consists of several gray squares for obstacles. The planner needs to find collision-free paths given the start and end points within the limit time.

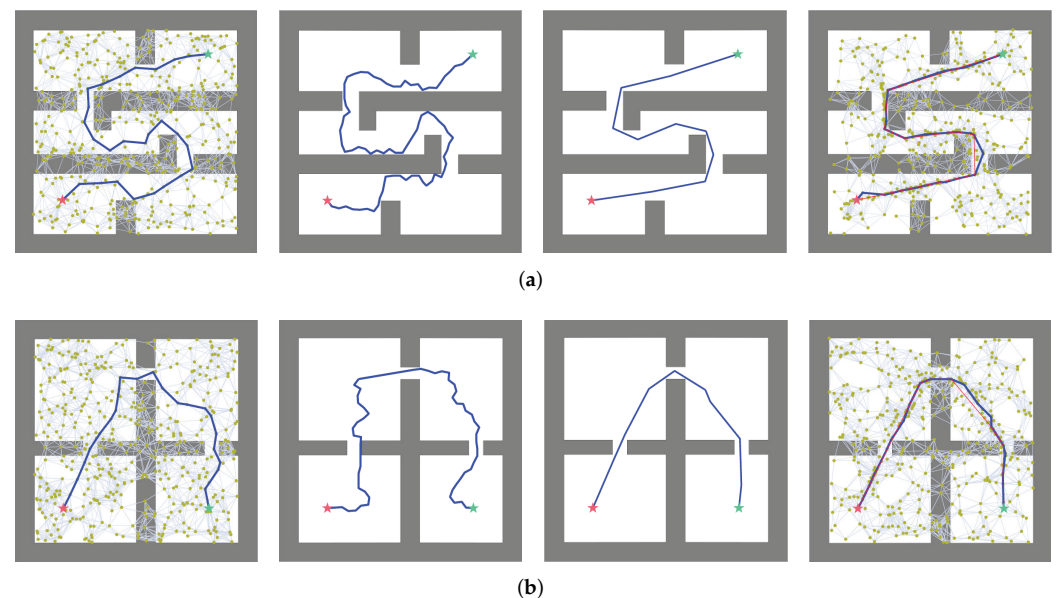


Figure 5. (a,b) are two different environments used in the 2D Gridworld experiments. From left to right, the planning results of LazyPRM, RRTConnect, BiRRT*, and LDG are shown. The green and red five-pointed stars represent the start and the goal, respectively; the blue line is the valid path generated by the planner; and the red line is the path of the LDG smoothed by the Rewire function, which can reduce the redundant state of the path.

We evaluated the performance of the algorithms from the three aspects of planning time, path cost, and success rate. Planning time measures the time spent by the planner to generate a path from the starting point to the goal point. Path cost reflects the ability of the planner to approach the optimal path. When all the states in the path are located outside the obstacles and the planner is not timed out, then the path is considered successful. Then, 30 tests were run in two scenes, and the experimental results were averaged; all the results of the four algorithms are shown in Tables 1 and 2.

As can be seen in Tables 1 and 2, LDG undergoes a greater improvement in the quality of the paths and success rate of planning compared to the LazyPRM algorithm, in the two scenarios, the length of the path decreases by 12.53% and 9.8%, respectively, and the planning success rate improved by 36.67% and 53.34%, so the efficiency has been improved. Quantitatively, the planning time of RRTConnect algorithm is slightly better than that of the LDG, but the path quality of our algorithm is improved by 28.9% and 36.9%, respectively, compared to RRTConnect. Compared with the BiRRT* algorithms, the LDG is slightly better than these two algorithms in terms of path quality, and has a significant advantage

in planning time, the planning time in each of the two scenarios was reduced by 85.3% and 88.7%. The above three evaluation indicators are better than RRT*. We also introduce the path smoothing strategy, which enables the quality of the path to be further improved and also guarantees that the quality of the path is good even with less prior knowledge.

Table 1. Speed, path cost and success rate of LDG benchmarked against other algorithms in scene 1.

| Algorithms | Planning Time | Path Cost | Success Rate |
|-------------|---------------|-----------|--------------|
| LazyPRM | 0.376 | 975.63 | 63.33 |
| RRT-Connect | 0.229 | 1200.25 | 100 |
| RRT* | 3.716 | 890.27 | 83.33 |
| BiRRT* | 2.637 | 869.33 | 100 |
| LDG | 0.387 | 853.41 | 100 |

Table 2. Speed, path cost and success rate of LDG benchmarked against other algorithms in scene 2.

| Algorithms | Planning Time | Path Cost | Success Rate |
|-------------|---------------|-----------|--------------|
| LazyPRM | 0.295 | 726.21 | 46.66 |
| RRT-Connect | 0.286 | 904.81 | 100 |
| RRT* | 4.841 | 678.47 | 76.66 |
| BiRRT* | 2.811 | 663.39 | 100 |
| LDG | 0.319 | 654.88 | 100 |

Overall, the performance of our algorithm is the best compared to the other four. In contrast, the LDG, such as the LazyPRM algorithm, delays the collision detection to the search phase of the path, which not only improves the speed of graph construction but also lets the LDG adapt to changes in the workspace to some degree. When the environment has partially changed, the LDG can also search the candidate path through the constructed graph. More importantly, for new planning problems, the tree-based method needs to explore the robotic arm configuration space again each time. The LDG algorithm, as a roadmap-based method, can save a large amount of information in the robotic arm configuration space in advance. In subsequent planning tasks, only directly calling the graph structure and performing the search for paths is needed. This approach also improves the planning efficiency. In the subsequent experiments, we used the multi-query nature of the LDG algorithm for continuous path planning for operation-oriented tasks.

3.2. Continuous Pick-and-Place of Manipulator

In this section, we design a robotic arm continuous motion planning experiment for an operational task in the Robot Operating System (ROS) environment. We first describe the experimental environment and experimental setup, and then compare the LDG with other robust planners in the same environment. The simulation environments included shelf stacking and picking from a deep bin, which are shown in Figure 6a,b separately. The robotic arm used is the AUBO-i5 with an attached Robotiq gripper. Taking the shelf stacking task as an example, this task required the robot arm to pick up all cylinders on the left side of the shelf and place them on its right side while avoiding obstacles in the workspace during the robot arm planning process. The operation task consists of four grasping and placing subtasks, for a total of eight motion paths.

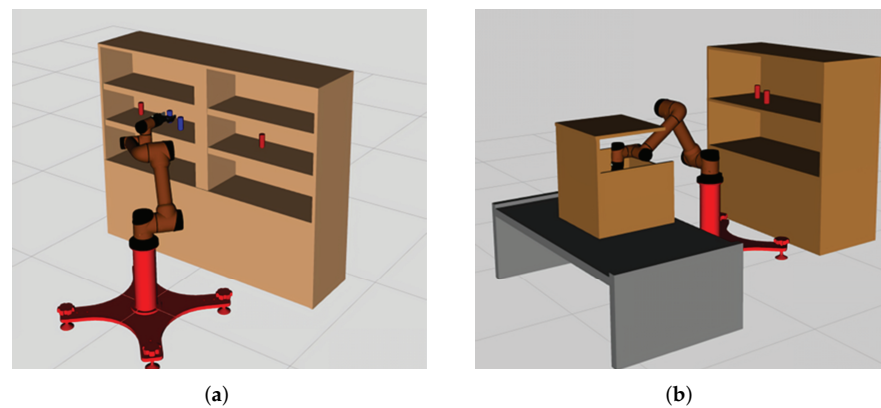


Figure 6. AUBO-i5 performs continuous pick-and-place tasks, shelf stacking (a), picking from deep bin (b).

We tested the LDG with the RRTConnect and BiRRT* algorithms in the same scenario and performed 10 iterations of the experiment for the shelf stacking task, with a total of 80 paths (picking and placing 40 times). As in the previous 2D experiments, we first used the BiRRT* algorithm to teach the operational task, and then used these demonstration paths as training data to train the GMM until convergence, the number of iterations of the EM algorithm is set to 100, the number of Gaussian components is 14 according to the BIC criterion, this step was performed offline. In practice, we do not teach all the picking and placing tasks by BiRRT*, because the paths planned in the same environment tend to have similar characteristics, a small number of paths are already representative. We also provide a certain percentage of uniform samples, which further improves the generalization ability of the planner. With the help of a trained Gaussian mixture distribution, the LDG can place more samples at task-relevant locations. The trajectory smoothing strategy was deployed for each planner, which could effectively improve the quality of the paths. All algorithms in this part of the experiment were implemented by C++ codes.

The experimental results are shown in Tables 3 and 4. Each subtask contained one picking and placing manipulation, and we measured the performance of the planner planning mainly from the two metrics of path cost and planning time. In the shelf stacking task and picking from a deep bin task, the initial number of nodes of the LDG was set to 3000 and 4500, and the ratio of Gaussian mixed sampling to uniform sampling was 1:2. We show the experimental data for the first two subtasks in the table, which shows the effectiveness of our proposed method, and the overall performance of the three planners in the four subtasks is compared more visually in box plots.

Table 3. Numerical results of AUBO-i5 repetitive motion planning in shelf stacking task.

| Algorithms | Tasks | Sequence | Planning Time | Path Cost |
|-------------|-------|----------|---------------|-----------|
| LDG | pick | 1st | 1.86 | 5.16 |
| | | 2nd | 0.95 | 4.26 |
| | place | 1st | 1.19 | 5.34 |
| | | 2nd | 0.93 | 4.32 |
| RRT-Connect | pick | 1st | 1.76 | 10.14 |
| | | 2nd | 1.63 | 9.49 |
| | place | 1st | 2.43 | 10.73 |
| | | 2nd | 2.27 | 10.01 |
| BiRRT* | pick | 1st | 6.67 | 5.98 |
| | | 2nd | 5.04 | 5.73 |
| | place | 1st | 7.28 | 6.6 |
| | | 2nd | 5.66 | 5.82 |

Table 4. Numerical results of AUBO–i5 repetitive motion planning in picking from bin task.

| Algorithms | Tasks | Sequence | Planning Time | Path Cost |
|-------------|-------|----------|---------------|-----------|
| LDG | pick | 1st | 4.54 | 6.47 |
| | | 2nd | 2.14 | 6.76 |
| | place | 1st | 2.07 | 6.69 |
| | | 2nd | 2.3 | 7.04 |
| RRT-Connect | pick | 1st | 3.16 | 9.24 |
| | | 2nd | 3.82 | 10.66 |
| | place | 1st | 4.06 | 11.31 |
| | | 2nd | 5.91 | 12.66 |
| BiRRT* | pick | 1st | 9.96 | 7.57 |
| | | 2nd | 10.49 | 7.98 |
| | place | 1st | 12.51 | 8.66 |
| | | 2nd | 14.69 | 7.69 |

As can be seen from the data in Tables 3 and 4, the LDG has a higher time cost than RRTConnect for the first picking operation, as the roadmap-based motion planning algorithm needs to construct a graph and save it. However, in the subsequent planning process, the LDG can reuse the saved graph structure in subsequent subtasks to achieve the effect of increasing the solution speed. Because RRTConnect and BiRRT* are tree-based motion planning algorithms, each query requires expanding the entire tree from scratch, which does not save computation time. Furthermore, because of the introduction of prior knowledge, the LDG can generate a large number of samples directly at these narrow passages in the workspace, so the average planning time of the LDG is smaller than the other two algorithms. In summary, when performing two picking and placing tasks, our method reduces the planning time by 39.1% and 34.1% compared to the RRTConnect algorithm in the two experimental scenarios, and the length of the path is reduced by 53.2% and 38.5%, respectively. Compared to the BiRRT* algorithm, our method reduces the planning time by 80% and 76.8% in the two experimental scenarios, and the length of the path is reduced by 20.9% and 15.5%, respectively.

Figure 7 visually shows the performance comparison of the three planning algorithms used by the AUBO–i5 manipulator to perform shelf stacking tasks in the form of box plots. The top plot corresponds to the planning time, and the bottom plot corresponds to the length of the path. Task 1 and Task 4 are shown from left to right, and each task contains a pick-and-place operation, represented by blue and orange boxes, respectively. It can be seen from the box plot that the LDG has the best performance in general, except the time spent in the pick operation of task 1 is higher than that of RRTConnect. Second, the algorithm has fewer outliers, which also reflects that the LDG algorithm has better robustness. These experimental results show that the LDG constructs a lazy probabilistic road map by introducing prior knowledge, which is suitable for repetitive operation motion planning in similar environments.

Figure 8 shows composite pictures of AUBO–i5 performing a shelf stacking task. As can be seen from task 1, the trajectory generated by the LDG algorithm is close to the optimal trajectory. When the pick-and-place manipulation has completed, we added a new obstacle, which invalidated some edges in the constructed roadmap, and the LDG was able to complete subsequent manipulation by querying candidate paths. This approach ensured the stability of the motion planning process, which benefits from the lazy data structure of the LDG in each task.

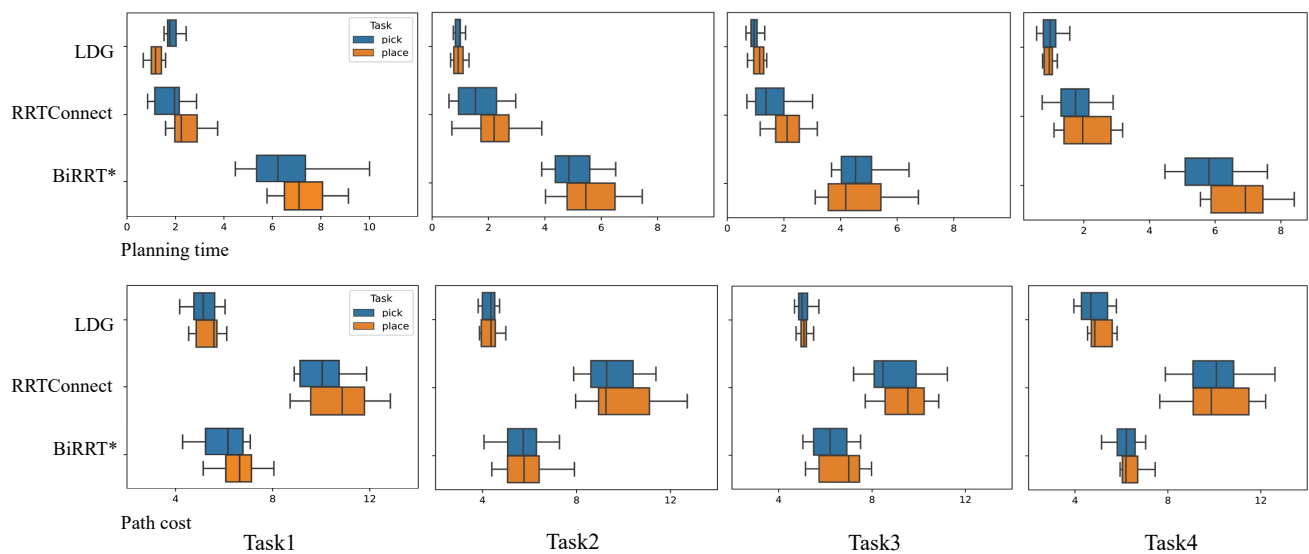


Figure 7. Experimental data obtained from the AUBO–i5 shelf stack tasks, which includes 4 times pick-and-place subtasks. Blue boxes and orange boxes are for pick-and-place tasks, respectively. To complete the manipulation task, each planner must plan a collision-free path to pick up the object and place it at the target location. The top boxplots represent the planning time, and the bottom boxplots represent the length of the planned path.

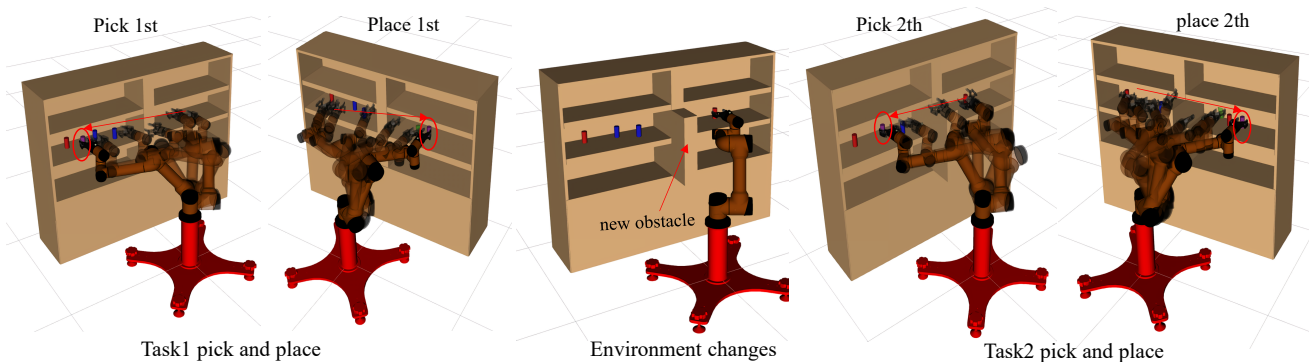


Figure 8. The composite images of the AUBO–i5 in shelf stacking task. These pictures show the robot executing a series of joint commands generated by LDG, picking up objects from the shelf and placing them on the other side. Although the roadmap in LDG algorithm is built for specific scenes, when the environment changes (the position of the operated object changes or new obstacles appear), the algorithm can still ensure the completion of subsequent tasks by querying candidate paths.

3.3. Physical Robot Arm Planning Experiment

In order to further verify the effectiveness of the method proposed in this paper, we use the AUBO–i5 physical manipulator to build a scene for picking and placing tasks, and use MoveIt! in ROS, the software platform builds a corresponding simulation environment according to the real-world scene, as is shown in Figure 9a,b.

In this section, we use ROS as the host computer to control the physical manipulator, so that the AUBO–i5 manipulator can pick objects from box1 and place them into box2, while avoiding obstacles in the workspace. We first used the BiRRT* algorithm to demonstrate the first pick-and-place task, and used the demonstration data as prior knowledge to construct an adaptive sampler by GMM. Then perform the second picking and placing operation, and use the current position as the starting point for the motion planning problem Figure 10, we use the proposed method to plan the path of the robotic arm, so that its gripper moves to box1 to grasp the object; the path sequence is shown in Figure 11.

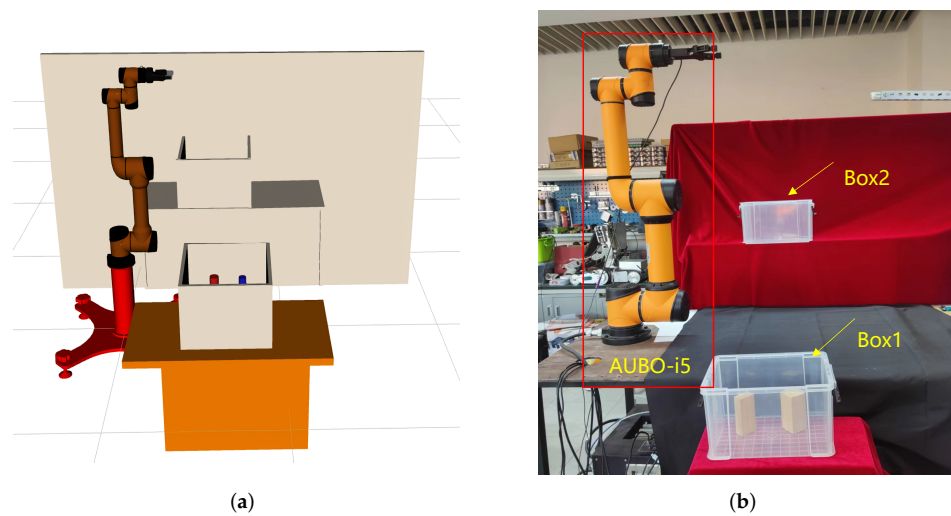


Figure 9. The scene for pick-and-place tasks in MoveIt! (a), AUBO–i5 physical manipulator (b).

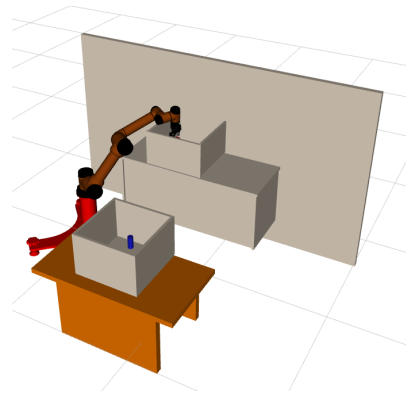


Figure 10. The starting position of AUBO–i5 when it is about to perform picking task.



Figure 11. The path sequence of AUBO–i5 when executing the picking task.

Similarly, we use the current position as the starting point for the placement task Figure 12, and to verify that our method can cope with changes in the workspace, we added a new obstacle between the two boxes, it can be seen that AUBO–i5 can still plan to the target position while avoiding obstacles in the workspace, realizing the placement of object; the path sequence is shown in Figure 13.

It can be seen from the experiments that our algorithm can be transferred from the simulation environment to the physical manipulator, and can better complete the motion planning for grasping and placing tasks, and at the same time, it can cope with changes in the workspace to a certain extent. The joint position change curve is shown in the following Figures 14 and 15.

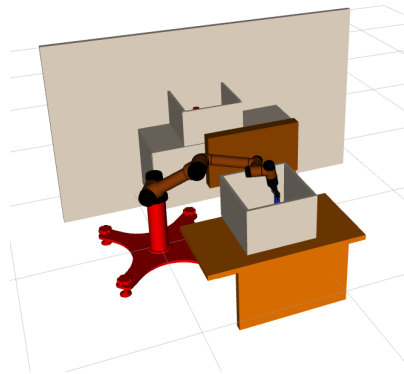


Figure 12. The starting position of AUBO–i5 when it is about to perform picking task.

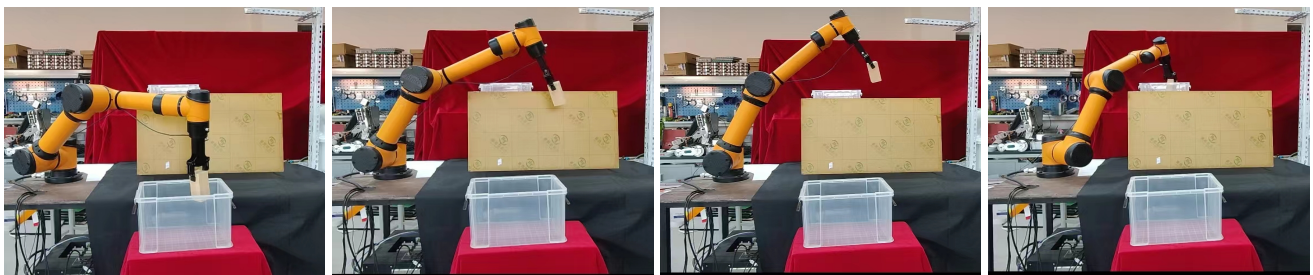


Figure 13. The path sequence of AUBO–i5 when it is about to perform placing task.

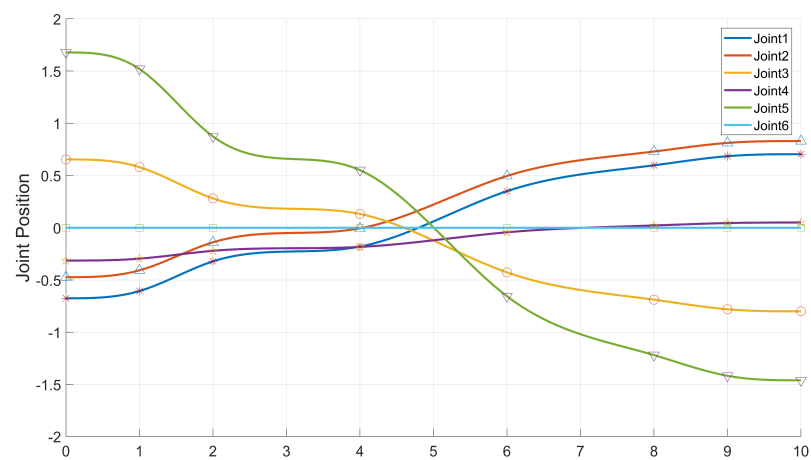


Figure 14. The joint position change curve of the AUBO–i5 when performing the picking task.

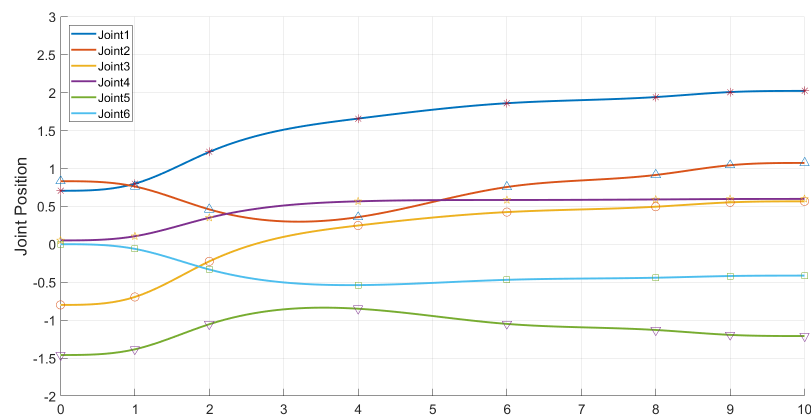


Figure 15. The joint position change curve of the AUBO–i5 when performing the placement task.

4. Conclusions

In this paper, we proposed LDG, a new motion planner for the pick-and-place tasks of a robotic arm. This method is a roadmap-based motion planning algorithm that uses the solutions of previous problems to speed up the solution of similar problems. We first analyzed and modeled the operation task-oriented planning problem of robotic arm, and then collected the high-quality trajectories planned by the expert planner as a priori knowledge. Notably, GMM fit an arbitrary distribution, therefore being able to learn the distribution of collected trajectory samples and build an adaptive sampler. Thus, GMM achieved sampling in specific regions of the optimal path solution to improve the speed and quality of motion planning. At the same time, we integrated this adaptive sampler into the LazyPRM algorithm, which enabled LDG to adapt to environmental changes as in a tree-based planning algorithm by delaying collision detection and guaranteeing the valid execution of subsequent tasks. For the new planning problem, the tree-based motion planning algorithm needed to plan from scratch, while we took advantage of the multi-query nature of the graph by saving the constructed road map at the end of the first planning task. In the follow-up planning problem, the persistent graph structure was called directly for the subsequent problem, thus further improving the speed of planning. We applied the proposed method to the continuous pick-and-place task of the AUBO-i5 manipulator, and both the simulation experiment and the physical experiment verified the effectiveness of our method. In the future, we will explore the combination of task planning with our proposed motion planning algorithm to solve complex long-term robot planning problems.

Author Contributions: Conceptualization, G.Z.; methodology, G.Z.; software, G.Z. and M.L.; validation, G.Z., M.L. and G.H.; formal analysis, G.Z.; investigation, G.Z. and M.L.; resources, G.Z. and M.L.; data curation, G.Z., M.L., J.Y., C.W. and G.H.; writing—original draft preparation, G.Z.; writing—review and editing, G.Z. and M.L.; visualization, M.L.; supervision, G.H.; project administration, G.Z.; funding acquisition, G.Z. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Nature Science Foundation of China grant number: 61873008 and 62103053, in part by the National Key Research and Development Program of China grant number: 2018YFB1307004.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank the biomimetics reviewers for their critical comments and suggestions for improving the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Thakar, S.; Rajendran, P.; Kabir, A.M.; Gupta, S.K. Manipulator motion planning for part pickup and transport operations from a moving base. *IEEE Trans. Autom. Sci. Eng.* **2020**, *19*, 191–206. [\[CrossRef\]](#)
2. Lai, T.; Ramos, F. Rapid replanning in consecutive pick-and-place tasks with lazy experience graph. *arXiv* **2021**, arXiv:2109.10209.
3. Farajtabar, M.; Daniali, H.M.; Varedi, S.M. Pick and place trajectory planning of planar 3-RRR parallel manipulator in the presence of joint clearance. *Robotica* **2017**, *35*, 241–253. [\[CrossRef\]](#)
4. Pairet, È.; Chamzas, C.; Petillot, Y.; Kavraki, L.E. Path planning for manipulation using experience-driven random trees. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3295–3302. [\[CrossRef\]](#)
5. Fisher, R.; Rosman, B.; Ivan, V. Real-time motion planning in changing environments using topology-based encoding of past knowledge. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018.
6. Tan, Y.; Ouyang, J.; Zhang, Z.; Lao, Y.; Wen, P. Path planning for spot welding robots based on improved ant colony algorithm. *Robotica* **2022**, 1–13. [\[CrossRef\]](#)
7. Sathiya, V.; Chinnadurai, M. Evolutionary algorithms-based multi-objective optimal mobile robot trajectory planning. *Robotica* **2019**, *37*, 1363–1382. [\[CrossRef\]](#)
8. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.

9. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Technical Report; Computer Science Department, Iowa State University: Ames, IA, USA, October 1998.
10. Kavraki, L.E.; Kolountzakis, M.N.; Latombe, J.C. Analysis of probabilistic roadmaps for path planning. *IEEE Trans. Robot. Autom.* **1998**, *14*, 166–171. [\[CrossRef\]](#)
11. Bekris, K.E.; Shome, R. Asymptotically optimal sampling-based planners. *arXiv* **2019**, arXiv:1911.04044.
12. Xu, J.; Wang, J. Effective motion planning of manipulator based on SDPS-RRTConnect. *Robotica* **2022**, *40*, 1855–1867. [\[CrossRef\]](#)
13. Bohlin, R.; Kavraki, L.E. Path planning using lazy PRM. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000.
14. Rosell, J.; Suárez, R.; Pérez, A. Path planning for grasping operations using an adaptive PCA-based sampling method. *Auton. Robot.* **2013**, *35*, 27–36. [\[CrossRef\]](#)
15. García, N.; Rosell, J.; Suárez, R. Motion planning using first-order synergies. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany, 28 September–2 October 2015; pp. 2058–2063.
16. Roveda, L.; Magni, M.; Cantoni, M.; Piga, D.; Bucca, G. Human-Robot Collaboration in Sensorless Assembly Task Learning Enhanced by Uncertainties Adaptation via Bayesian Optimization. *Robot. Auton. Syst.* **2020**, *136*, 103711. [\[CrossRef\]](#)
17. Qiu, Q.; Cao, Q. Motion planning in semistructured environments with teaching roadmaps. *Intell. Serv. Robot.* **2020**, *13*, 1855–1867. [\[CrossRef\]](#)
18. Berenson, D.; Abbeel, P.; Goldberg, K. Experience-based planning with sparse roadmap spanners. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia, 26–30 May 2012.
19. Coleman, D.; Şucan, I.A.; Moll, M.; Okada, K.; Correll, N. Experience-based planning with sparse roadmap spanners. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 25–30 May 2015.
20. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Man Cybern. Syst.* **1968**, *4*, 100–107. [\[CrossRef\]](#)
21. Tsukamoto, H.; Chung, S.J. Learning-based robust motion planning with guaranteed stability: A contraction theory approach. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6164–6171 [\[CrossRef\]](#)
22. Ichter, B.; Pavone, M. Robot motion planning in learned latent spaces. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2407–2414. [\[CrossRef\]](#)
23. Qureshi, A.H.; Simeonov, A.; Bency, M.J.; Yip, M.C. Motion planning networks. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–22 May 2019.
24. Qureshi, A.H.; Miao, Y.; Simeonov, A.; Yip, M.C. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Trans. Robot.* **2020**, *37*, 48–66. [\[CrossRef\]](#)
25. Strudel, R.; Garcia, R.; Carpentier, J.; Laumond, J.P.; Laptev, I.; Schmid, C. Learning obstacle representations for neural motion planning. *arXiv* **2020**, arXiv:2008.11174.
26. Biernacki, C.; Celeux, G.; Govaert, G. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Comput. Stat. Data Anal.* **2003**, *41*, 561–575. [\[CrossRef\]](#)
27. Fujimoto, Y.; Murata, N. A modified EM algorithm for mixture models based on Bregman divergence. *Ann. Inst. Stat. Math.* **2007**, *59*, 3–25. [\[CrossRef\]](#)
28. Kashef, R.; Kamel, M.S. Enhanced bisecting k-means clustering using intermediate cooperation. *Pattern Recognit.* **2009**, *42*, 2557–2569. [\[CrossRef\]](#)
29. Burnham, K.P.; Anderson, D.R. Multimodel inference: Understanding AIC and BIC in model selection. *Sociol. Methods Res.* **2004**, *33*, 261–304. [\[CrossRef\]](#)
30. Pan, J.; Chitta, S.; Manocha, D. FCL: A general purpose library for collision and proximity queries. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia, 21–25 May 2012.
31. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000.
32. Klemm, S.; Oberländer, J.; Hermann, A.; Roennau, A.; Schamm, T.; Zollner, J.M.; Dillmann, R. Rrt-connect: Faster, asymptotically optimal motion planning. In Proceedings of the IEEE international conference on robotics and biomimetics, Zhuhai, China, 6–9 December 2015.