Received June 25, 2021, accepted July 12, 2021, date of publication July 19, 2021, date of current version July 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3098044

Research on the Dynamic Path Planning of Manipulators Based on a Grid-Local Probability Road Map Method

YOUYU LIU, BO CHEN, XUYOU ZHANG, AND RENJUN LI
Key Laboratory of Advanced Perception and Intelligent Control of High-End Equipment, Ministry of Education, Wuhu 241000, China
School of Mechanical Engineering, Anhui Polytechnic University, Wuhu 241000, China

Corresponding author: Bo Chen (2190110112@stu.ahpu.edu.cn)

This work was supported in part by the Special Fund for Collaborative Innovation of Anhui Polytechnic University and Jiujiang District under Grant 2021CYXTB3, and in part by the Natural Science Research Project of Higher Education of Anhui Province of China under Grant KJ2020A0357.

ABSTRACT A Grid-Local Probability Road Map (PRM) method was proposed for the path planning of manipulators in dynamic environments. Based on the idea of boundary discretization, a double-grid model was built to obtain a mapping from dynamic obstacles to configuration space. The collision detection was simplified as a data indexing process to improve its efficiency. Times of collision detections were reduced by employing local programming strategies and the stratified sampling method. Moreover, the validity of sampling was increased. Taking the PUMA560 manipulator as a research object, the simulation experiments show that the time consumption of the proposed simplified collision-detection algorithm is about 14% of that of the standard one, and the stratified sampling is beneficial to the generation of probability maps compared with simple random sampling method. The simulation experiment of the static path planning shows that the proposed algorithm consumes an average of 10ms, which is superior to the comparison algorithm and has high efficiency and real-time performance. The simulation experiment of the dynamic path planning shows that the proposed algorithm consumes an average of 7ms per step, which is better than the comparison algorithm. The proposed algorithm can adjust the global path in real time to avoid obstacles as the environment changes. The algorithm mentioned has been proved to be efficient.

INDEX TERMS Dynamic environment, double grid model, probability road map, path planning.

I. INTRODUCTION

Robot path planning [1] is to plan a path from the beginning to the ending, and can safely avoid the obstacles in the environment. Compared with static environments, path planning in dynamic environments requires higher real-time performance of algorithms [2]. For the dynamic path planning of multi-degree-of-freedom (DOF) manipulators, larger workspace and a large number of online collision detection become a bottleneck that limit the efficiency of the planning algorithms [3]. Consequently, scholars have done a lot of researches on spatial description methods and planning algorithms. As for the spatial description method, Lozano-Perez proposed a method to construct a mapping of obstacles in the configuration space by solving the limit of joint angle of multi-DOF manipulators [4]. Due to the complexity of mapping function, this method is difficult to be applied in practice. Newman proposed an analytical method, which regarded obstacles as a combination of features such as points, lines and planes, mapped these features to the configuration space through the analytical method, and obtained the mapping of obstacles in the configuration space through union [5]. However, only the mapping method of 2-DOF manipulators is discussed in the article, which is difficult to be applied to multi-DOF manipulators. As for planning algorithms, PRM algorithm [6], which was generated in the 1990s, is not limited by the spatial dimension and is widely used to solve the path planning problem of multi-DOF manipulators. However, it is difficult to apply in the dynamic environment since probability road maps take a long time to construct and cannot be reused. Leven proposed an extended PRM algorithm, which constructed the probability map without obstacles firstly, then

The associate editor coordinating the review of this manuscript and approving it for publication was Luigi Biagiotti.

added obstacles and adjusted the probability map to obtain a non-collision path in the query stage [7]. Bohlin proposed a planning algorithm suitable for dynamic environments, which constructed a probability map in the environment containing only static obstacles, and quickly updated the probability map combined with the lazy evaluation mechanism [8] to obtain a new path after the introduction of dynamic obstacles. However, these two methods still describe the dynamic environments in a narrow sense and do not involve the movement of obstacles or the transient occlusion of the starting and ending point. To sum up, firstly, a mapping method from obstacles to configuration space is complex and difficult to be applied to multi-DOF manipulators. Secondly, it is difficult for PRM algorithm to be used in dynamic environments and the object of some improved PRM algorithms proposed is still a narrow dynamic environment.

Based on the idea of boundary discretization [9], a grid mapping model was built to realize a rapid update of environmental maps. Introducing local programming idea [10] into PRM algorithm, a local PRM algorithm was proposed combining with the simplified collision-detection algorithm, which can break through the limitation that the standard PRM algorithm can only be used in a static space, and realize its path planning in a dynamic space. In the Section II, a mapping method from obstacles to configuration space is described, and a derived simplified collision-detection algorithm is proposed. In the Section III, the obstacle description, the sampling method and the proposed local PRM algorithm are described. In the Section IV, the mapping model, the collision-detection algorithm, the sampling method and the Grid-Local PRM algorithm are verified respectively by simulation experiments.

II. GRID MAPPING MODEL

A. DOUBLE GRID MODEL

Due to obstacles moving in real time, the changing spatial position cannot be quickly calculated by a complex continuous model, which makes online computing unavailable [11]. Consequently, instead of the continuous modeling method, we discretize the task space and configuration space, and propose a dynamic modeling method based on boundary points.

The task space is uniformly discretized, and the index is denoted as i. The i-th grid center vector $\boldsymbol{t}_i^{\text{center}}$ and its isotropic equivalent radius $\boldsymbol{t}_i^{\text{radius}}$ are as follows.

$$\begin{cases} \boldsymbol{t}_i^{\text{center}} = \left(\boldsymbol{t}_{\min} + (i - \frac{1}{2})\dfrac{(\boldsymbol{t}_{\max} - \boldsymbol{t}_{\min})}{\boldsymbol{n}_{\text{t}}}\right) \\ \boldsymbol{t}_i^{\text{radius}} = \dfrac{1}{2}\dfrac{(\boldsymbol{t}_{\max} - \boldsymbol{t}_{\min})}{\boldsymbol{n}_{\text{t}}} \end{cases} \tag{1}$$

where $\boldsymbol{n}_{\text{t}}$ is the coefficient matrix about the division of task space, and $\boldsymbol{t}_{\min}$, $\boldsymbol{t}_{\max}$ are the boundary matrixes for the task space.

Similarly, the configuration space is uniformly discretized, and the index is denoted as j. The j-th grid center vector $\boldsymbol{c}_j^{\text{center}}$

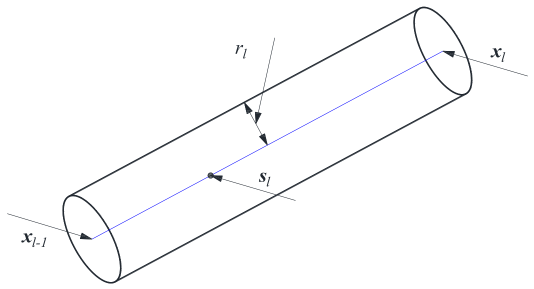

FIGURE 1. Diagram of a linkage.

and its isotropic equivalent radius $\boldsymbol{c}_j^{\text{radius}}$ are as follows.

$$\begin{cases} \boldsymbol{c}_j^{\text{center}} = \left(\boldsymbol{c}_{\min} + (j - \frac{1}{2})\dfrac{(\boldsymbol{c}_{\max} - \boldsymbol{c}_{\min})}{\boldsymbol{n}_{\text{c}}}\right) \\ \boldsymbol{c}_j^{\text{radius}} = \dfrac{1}{2}\dfrac{(\boldsymbol{c}_{\max} - \boldsymbol{c}_{\min})}{\boldsymbol{n}_{\text{c}}} \end{cases} \tag{2}$$

where $\boldsymbol{n}_{\text{c}}$ is the coefficient matrix about the division of configuration space, and $\boldsymbol{c}_{\min}, \boldsymbol{c}_{\max}$ are the boundary matrixes for the configuration space.

B. MATHEMATICAL MAPPING MODEL

To solve the mapping relationship between the grid t_i and the grid c_j, starting from the kinematics of the manipulators, a relative transformation matrix [12] of each linkage of the manipulators in the standard D-H coordinates can be expressed as follows.

$$ {}_{n}^{n-1}\boldsymbol{T} = \begin{bmatrix} c\theta_n & -s\theta_n & 0 & a_{n-1} \\ s\theta_n c\alpha_{n-1} & c\theta_n c\alpha_{n-1} & -s\alpha_{n-1} & -s\alpha_{n-1}d_n \\ s\theta_n s\alpha_{n-1} & c\theta_n s\alpha_{n-1} & c\alpha_{n-1} & c\alpha_{n-1}d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where a is a coefficient representing the translation of the x-axis, α is a coefficient representing the rotation of the x-axis, d is a coefficient representing the translation of the z-axis, θ is a coefficient representing the rotation of the z-axis, s stands for short form for cosine function, and c stands for short form for sine function.

For a n-DOF manipulator with given a joint configuration, some coefficients such as a, α and d being all given, the transformation matrix (see Eq. (3)) can be expressed by the matrix function of the joint angle, as shown in Eq. (4).

$$ {}_{n}^{n-1}\boldsymbol{T} = {}_{n}^{n-1}f(\theta_1, \theta_2, \cdots, \theta_n) \tag{4}$$

As for a linkage l shown in Fig. 1, giving any configuration $(\theta_1, \theta_2, \ldots, \theta_n)$, from Eq. (4), a recursive relation between the left endpoint $\boldsymbol{x}_{l-1}$ and the right endpoint $\boldsymbol{x}_l$ is as follows.

$$\boldsymbol{x}_l = {}_{l}^{l-1}\boldsymbol{T} \cdot \boldsymbol{x}_{l-1} = {}_{l}^{l-1}f(\theta_1, \theta_2, \cdots, \theta_l) \cdot \boldsymbol{x}_{l-1} \tag{5}$$

Suppose that the origin of the base coordinate system is $\boldsymbol{x}_0 = (0, 0, 0, 1)^{\text{T}}$. According to Eq. (5), the relationship between origins of each joint coordinate system and origin of the base coordinate system can be obtained, as shown in Eq. (6).

$$\boldsymbol{x}_l = {}_{l}^{l-1}f(\theta_1, \theta_2, \cdots, \theta_l) \cdot \boldsymbol{x}_{l-1}$$

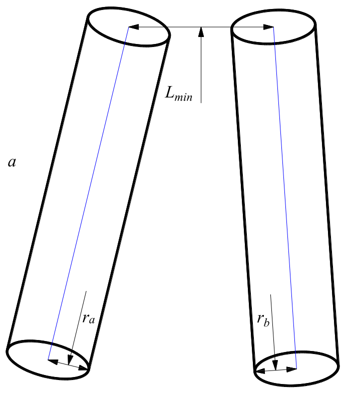

FIGURE 2. **Diagram of two-linkage interference.**

$$\begin{aligned}
&= {}_{l}^{l-1}f(\theta_1,\theta_2,\cdots,\theta_l)\cdot{}_{l-1}^{l-2}f(\theta_1,\theta_2,\cdots,\theta_{l-1})\cdot\boldsymbol{x}_{l-2}\\
&=\cdots\\
&=\prod_{k=1}^{l}{}_{k}^{k-1}f(\theta_1,\theta_2,\cdots,\theta_k)\cdot\boldsymbol{x}_0
\end{aligned}\tag{6}$$

According to the theorem of segment with fixed ratio [13], any point $\boldsymbol{s}_l$ in the linkage l can be expressed by the left endpoint $\boldsymbol{x}_{l-1}$ and the right endpoint $\boldsymbol{x}_l$, as shown in Eq. (7).

$$\boldsymbol{s}_l=\lambda_l(\boldsymbol{x}_l-\boldsymbol{x}_{l-1})+\boldsymbol{x}_{l-1}\tag{7}$$

From Eq. (6) and Eq. (7), we can infer the following.

$$\begin{aligned}
\boldsymbol{s}_l=&\prod_{k=1}^{l-1}{}_{k}^{k-1}f(\theta_1,\theta_2,\cdots,\theta_k)\cdot\boldsymbol{x}_0\\
&+\lambda_l\prod_{k=1}^{l}{}_{k}^{k-1}f(\theta_1,\theta_2,\cdots,\theta_k)\cdot\boldsymbol{x}_0\\
&-\lambda_l\prod_{k=1}^{l-1}{}_{k}^{k-1}f(\theta_1,\theta_2,\cdots,\theta_k)\cdot\boldsymbol{x}_0
\end{aligned}\tag{8}$$

where λ is a coefficient with interval (0,1), representing the position of point in the linkage.

As shown in Fig. 2, in addition to the collision between the manipulator and grid elements, the self-interference of the manipulator should also be considered [14]. For two adjacent linkages, the interference can be prevented by controlling the limit range of joint angle. Therefore, we only need to study the interference between non-adjacent linkages about the self-interference of manipulator. Combined with the previous linkage model, the shortest distance between two linkages is as follows.

$$\min d_{a,b}=L_{\min}-(r_a+r_b)\tag{9}$$

where r_a is the envelope radius of the linkage a, r_b is that of the linkage b, and $L_{\min}$ is the shortest distance between the shafts of the manipulator in a configuration $\boldsymbol{\theta}$.

$\left|\boldsymbol{s}_l\boldsymbol{c}_j^{\text{center}}\right|$ is defined as the Euclidean distance from any point in the linkage l to the center of the grid c_j in a configuration space.

The collision detection between the grid t_i and the grid c_j can be described as a nonlinear optimization problem [15], as shown in Eq. (10).

$$\left.\begin{array}{c}
\min f(x)\\
x=(\lambda_1,\lambda_2,\cdots,\lambda_n,\theta_1,\theta_2,\cdots,\theta_n)\\
f(x)=\min(\left|\boldsymbol{s}_1\boldsymbol{c}_j^{\text{center}}\right|,\left|\boldsymbol{s}_2\boldsymbol{c}_j^{\text{center}}\right|,\cdots,\left|\boldsymbol{s}_n\boldsymbol{c}_j^{\text{center}}\right|)\\
s.t.\\
0\le\lambda_1,\lambda_2,\cdots,\lambda_n\le 1\\
\boldsymbol{c}_j^{\text{center}}-\boldsymbol{c}_j^{\text{radius}}<\theta_1,\theta_2,\cdots,\theta_n\le\boldsymbol{c}_j^{\text{center}}+\boldsymbol{c}_j^{\text{radius}}\\
\prod_{k=1}^{n}\min d_k^{k-1}>0
\end{array}\right\}\tag{10}$$

By using Differential Evolution (DE) algorithm [16] to achieve the optimal solution $\min f(x)$ of the problem shown in Eq. (10), the mapping function $col_{ji}(\theta_1,\theta_2,\cdots,\theta_n)$ between t_i and c_j can be constructed, as shown in Eq. (11), where 1 is the collision state and 0 is the non-collision state.

$$\begin{aligned}
&col_{ji}(\theta_1,\theta_2,\cdots,\theta_n)\\
&\quad=\begin{cases}1, & \min f(x)\le\max(\boldsymbol{t}_i^{\text{radius}})+r_l\\ 0, & else\end{cases}
\end{aligned}\tag{11}$$

where r_l is the envelope radius of the linkage l.

III. DYNAMIC PATH PLANNING

A. DYNAMIC OBSTACLES

The motion of dynamic obstacles can be regarded as a rigid body motion, which can be decomposed into translational motion and rotational one. Let the rotation matrix be $\boldsymbol{M}$ and the translation vector be $\boldsymbol{v}$, as shown in Eq. (12) and Eq. (13). A point collected from the surface of dynamic obstacle is fixed with it and moves together with it. At time t, the position of the point is recorded as $\boldsymbol{p}=(p_1,p_2,p_3,\ldots,p_N)^{\mathrm{T}}$; At the $t+1$ moment after movement, the point position is recorded as $\boldsymbol{p}'=(p_1',p_2',p_3',\ldots,p_N')^{\mathrm{T}}$. Its calculation formula is shown in Eq. (14).

$$\boldsymbol{M}=\begin{pmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\\r_{31}&r_{32}&r_{33}\end{pmatrix}\tag{12}$$

$$\boldsymbol{v}=\begin{pmatrix}p_x\\p_y\\p_z\end{pmatrix}\tag{13}$$

$$\boldsymbol{p}'=\boldsymbol{M}\boldsymbol{p}+\boldsymbol{v}\tag{14}$$

B. COLLISION DETECTION

Based on the proposed obstacles model, the sequences of grids occupied in the task space can be obtained in real time, denoted as $\boldsymbol{List}_{\text{obs}}=(i_1,i_2,i_3,\ldots,i_N)$. Thus, the mapping state $occupy_{j,[i_1,i_2,i_3,\cdots,i_N]}$ of the grid j under the influence of $\boldsymbol{List}_{\text{obs}}$ can be retrieved, abbreviated as $occupy_j$, as shown in Eq. (15). The sequences of the grids occupied in the task space can be obtained by traversing each j.

$$occupy_j = \sum_{k=i_1}^{i_N} col_{jk} \tag{15}$$

Based on the proposed mapping relationship between configuration space and task space, a simplified collision-detection method is proposed.

For any given configuration $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)^{\mathrm{T}}$, using the proposed model, the high-dimensional index $\boldsymbol{Hid}$ of any grid occupied in configuration space is shown in Eq. (16). The single-dimensional index Oid of that is shown in Eq. (17).

$$\boldsymbol{Hid}_k = \frac{\boldsymbol{n}_c^k ceil(\boldsymbol{\theta}_k - \boldsymbol{c}_{\min}^k)}{\boldsymbol{c}_{\max}^k - c_{\min}^k} \tag{16}$$

$$Oid = \boldsymbol{Hid}_1 + \sum_{k=2}^{n} \left\{ \left(\boldsymbol{n}_c^{k-1}\right)^{k-1} (\boldsymbol{Hid}_k - 1) \right\} \tag{17}$$

The collision state between the configuration $\boldsymbol{\theta}$ and the obstacles can be represented by $isnonfree(\boldsymbol{\theta})$, shown in Eq. (18). That is, the collision detection is transformed into looking up database.

$$isnonfree(\boldsymbol{\theta}) = \begin{cases} 1, & occupy_{Oid} > 0 \\ 0, & else \end{cases} \tag{18}$$

C. LOCAL PRM ALGORITHM

Note that $\boldsymbol{x}_{\text{start}}$ is the global starting point, $\boldsymbol{x}_{\text{goal}}$ is the global goal point, $\boldsymbol{x}_{\text{current}}$ is the current position, $\boldsymbol{x}_1 \to \boldsymbol{x}_2$ is the path segment connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and the back-end endpoint of each path segment is a sub-goal point, marked as $\boldsymbol{x}_{\text{sub}}$.

In global path planning, there are usually two assumptions [17]: (1) the starting point and the ending point cannot be blocked by obstacles; (2) the ending point cannot be blocked by obstacles. In a dynamic environment, if the starting and the ending points are blocked by obstacles, and the manipulator is not at these positions, then the algorithms should be solvable. It is clear that this assumption is invalid.

As shown in Fig. 3, a lazy collision-detection strategy [18] is introduced in this article. There is no collision detection in the construction of probability maps or in the path query phases. That is to say, both the undirected maps and the searched global path may be "illegal". In this way, the hypothesis mentioned above can be avoided.

Secondly, in order to ensure that the path is feasible, local subgoal points being introduced, the collision detection is only performed on the piece $\boldsymbol{x}_{\text{current}} \to \boldsymbol{x}_{\text{sub}}$ to ensure the feasibility from the current position to the global next standard. However, the global path is likely to be "illegal". All in all, the strategy is only for the local area, reducing the number of collision detections as much as possible, to ensure that PRM algorithm can be used in a dynamic environment.

Latin hypercube sampling [19] is employed in this article. It is a kind of stratified sampling [20], which can simulate the distribution of problems with fewer points.

Points obtained by the simple random sampling are easy to aggregate, and part of them may be invalid. However, points

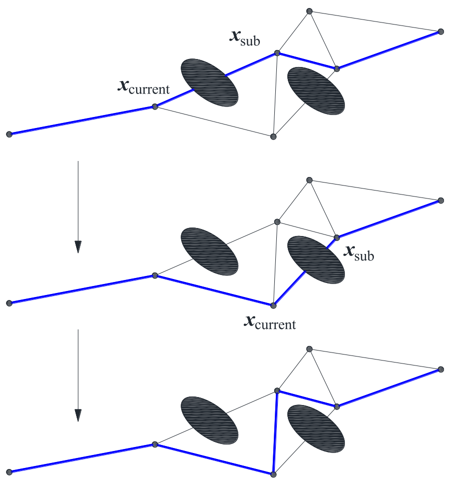

FIGURE 3. Diagram of Local PRM algorithm strategy.

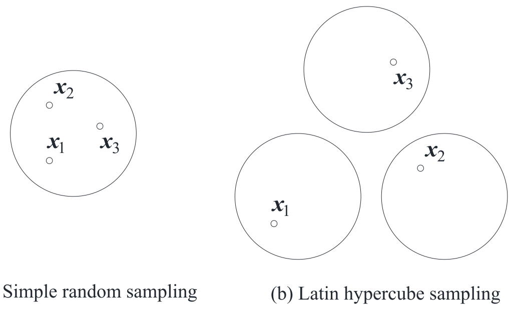

FIGURE 4. Simple random sampling and Latin hypercube sampling.

of the Latin hypercube sampling belong to different layers, and each sampling point can represent the characteristics of a local region. As shown in Fig. 4, using simple random sampling, it is easy to cause the sampling points to be located in a small area. Unlike it, each point obtained by Latin hypercube sampling belongs to a different small area.

A$*$ algorithm [21] is used to query the global path. The heuristic function $f(\boldsymbol{x})$ is shown in Eq. (19), and the distance is defined as an Euclidean distance [22] between two points.

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x}) \tag{19}$$

where $g(\boldsymbol{x})$ is the distance cost between the sampling point and the current point, and $h(\boldsymbol{x})$ is the distance cost between the sampling point and the goal point.

Steps of the algorithm are as follows, and the flowchart is shown in Fig. 5.

Step 1: Initialization. Including adjacency radius ρ, sample size N, index of path segment pid, global environment $world$, and current position $\boldsymbol{x}_{\text{current}}$.

Step 2: Sampling from the configuration space and the points obtained are denoted as $\boldsymbol{x}_{\text{sample}}^k$, in which the k is the index of the sampling points.

Step 3: The set of points $\boldsymbol{C}$ is constituted by x_{start}, x_{goal} and x_{sample}, and the set of edges $\boldsymbol{V}$ is calculated according to the adjacency radius.

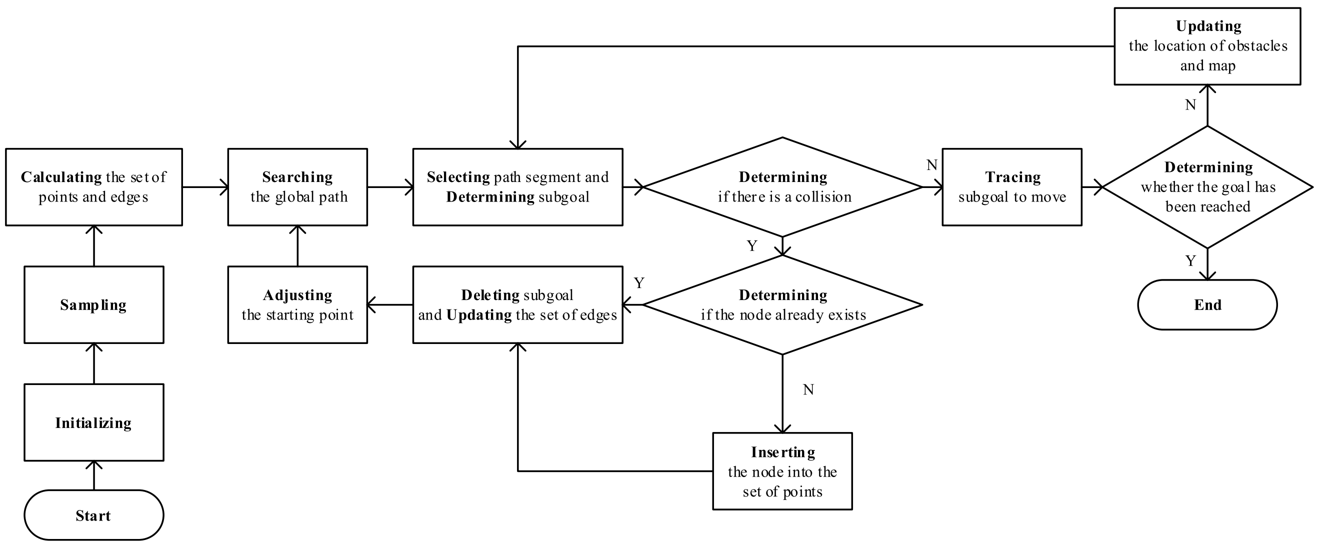
Start
Initializing
Sampling
Calculating the set of points and edges
Searching the global path
Selecting path segment and Determining subgoal
Determining if there is a collision
N
Tracing subgoal to move
Determining whether the goal has been reached
N
Updating the location of obstacles and map
Y
End
Y
Determining if the node already exists
Y
Deleting subgoal and Updating the set of edges
Adjusting the starting point
N
Inserting the node into the set of points

FIGURE 5. Flowchart of the grid-local PRM algorithm.

Step 4: Taking the edge set $\boldsymbol{V}$ as a search graph, A* algorithm is used to search the path ***path***.

Step 5: Let $\boldsymbol{x}_{\text{sub}}$ be $\boldsymbol{path}(pid)$, and judging whether $\boldsymbol{x}_{\text{current}} \rightarrow \boldsymbol{x}_{\text{sub}}$ piece collides with the obstacles.

Step 6: If there is no collision, $\boldsymbol{x}_{\text{current}} \rightarrow \boldsymbol{x}_{\text{sub}}$ trajectory is planned, thus $\boldsymbol{x}_{\text{current}}$ tracking the trajectory to move. If $\boldsymbol{x}_{\text{current}}$ reaches $\boldsymbol{x}_{\text{sub}}$, then $pid = pid + 1$.

Step 7: If there is a collision, determining whether $\boldsymbol{x}_{\text{current}}$ already exists in the set of points $\boldsymbol{C}$. If it is true, deleting subgoal point, adjusting the set of edges $\boldsymbol{V}$, initializing pid, and returning to step. 4. Otherwise, adding $\boldsymbol{x}_{\text{current}}$ to the set of points $\boldsymbol{C}$, recalculating the set of edges $\boldsymbol{V}$, delete sub-goal point, adjust the set of edges $\boldsymbol{V}$, initializing pid, and returning to step. 4.

Step 8: Repeating steps 5 to 7 until x_{current} reaches x_{goal}.

IV. SIMULATIONS AND VERTIFCATIONS

Taking a 6-DOF PUMA560 manipulator as an example, some simulation experiments by MATLAB and Robotics Toolbox [23] and theirs analysis are carried out. The D-H parameters of the PUMA560 manipulator are shown in Table. 1. Let $\boldsymbol{n}_{\text{t}} = 30 \cdot \text{ones}\,(6, 1)$, $\boldsymbol{n}_{\text{c}} = 30 \cdot \text{ones}\,(6, 1)$ and $\boldsymbol{t}_{\min} = -1 \cdot \text{ones}\,(3, 1)$, $\boldsymbol{t}_{\max} = 1 \cdot \text{ones}\,(3, 1)$; $\boldsymbol{c}_{\min}$ and $\boldsymbol{c}_{\max}$ is given by the D-H parameters [24]. The mapping from the task space to the configuration space can be calculated according to the method described in Section II.

Since the three posterior joints of the PUMA560 manipulator have no influence on the collision state between the whole manipulator and obstacles [25], its configuration space can be expressed explicitly by the method of dimensionality reduction.

Some grid indexes of obstacles are randomly selected to express theirs mapping to the configuration space explicitly. Fig. 6a is a single obstacle. Fig. 6b is combination obstacles. Fig. 6c and Fig. 6d are their corresponding mappings in the configuration space respectively. Moreover, the configuration

TABLE 1. D-H parameters of the PUMA560 manipulator.

θ	d (m)	a (m)	α (m)	range(°)
θ_1	0	0	1.5708	-160~160
θ_2	0	0.4318	0	-225~45
θ_3	0.1500	0.0203	-1.5708	-45~225
θ_4	0.4318	0	1.5708	-110~170
θ_5	0	0	-1.5708	-100~100
θ_6	0	0	0	-266~266

TABLE 2. Time consumption comparison among the two algorithms(s).

	avg	med	std
Standard one	8.1072e-05	8.1650e-05	3.9785e-05
Simplified one	1.0993e-05	1.0950e-05	3.4272e-06

space corresponding to the multiple grids is the union of those corresponding to the single grids.

In order to verify the performance of the proposed collision-detection algorithm, some obstacle sequences and manipulator joint configurations are randomly extracted from the task space and the configuration space respectively. The collision state between the obstacles and the manipulator are detected by employing the simplified collision-detection algorithm and the standard collision-detection algorithm [26] respectively, on which a total of 100 such calculations are performed. The statistical data of the experiment are shown in Table 2, which further supports the above conclusion. The average time consumption of the simplified collision-detection algorithm is only 13.56% of that of the standard collision-detection algorithm, and the median time consumption is only 13.41%. The standard deviation of the time consumption of the simplified collision-detection algorithm is also an order of magnitude higher than that of the standard collision-detection algorithm.

As shown in Fig. 7, the red dots in it are the time-consumption dots with the standard collision-detection algorithm, the blue dots are those with the simplified

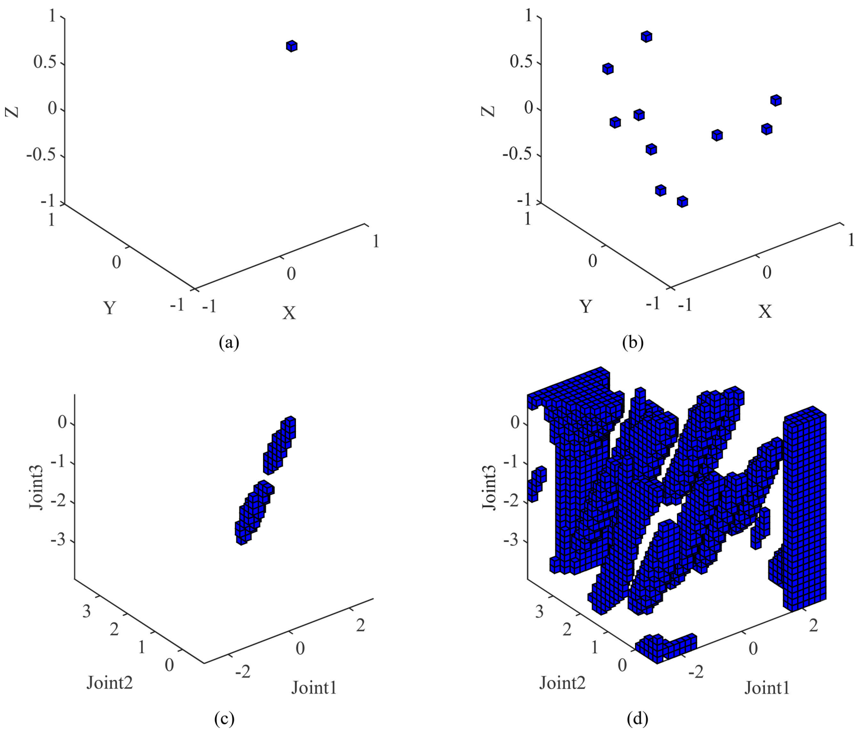

FIGURE 6. Mapping between the task space and the configuration space.

collision-detection algorithm, and the horizontal lines with corresponding colors are the average value lines.

According to the distribution pattern of the dots, it is obvious that the dots with standard collision-detection algorithm are scattered, while the results of the simplified collision-detection algorithm are dense. Since the complexity of the environment selected for the experiments is random, it means that the detection time of the algorithm in this article is not sensitive to the complexity of the environment. According to the average value lines, the time consumption of the simplified collision-detection algorithm is less than that of the standard collision-detection algorithm.

The Latin hypercube sampling and the simple random sampling were used respectively to sample and build probability maps in two-dimensional graphs with the starting point of (0, 0) and the ending point of (1, 1), on which a total of 100 such calculations are performed.

As shown in Fig. 8 is a set of experimental results. Fig. 8. Fig. 8a and Fig. 8b that are the sampling point graphs using the Latin hypercube sampling and simple random sampling, respectively. Fig. 8c and Fig. 8d are the undirected graphs using the Latin hypercube sampling and simple random sampling, respectively. As shown in Table 3 is the average values of the aggregation degree and coverage area, where the aggregation degree is the average value of the reciprocal of the distance between any two points of effective connection, and the coverage area is the average values of the polygon area formed by the boundary of the undirected graphs.

Comparing with Fig. 8a and Fig. 8b, it can be seen that the sampling points using simple random sampling method

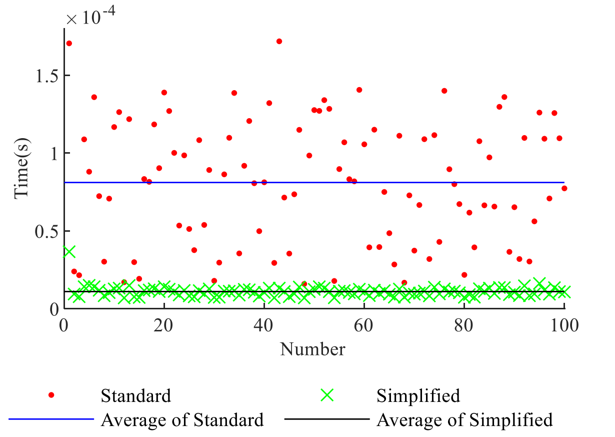

FIGURE 7. Graph of the time consumption of algorithms.

occupy less space than the Latin hypercube sampling method, and an aggregation phenomenon occurs in the upper left corner. Similarly, Comparing with Fig. 8c and Fig. 8d, it can be seen that the undirected maps of the simple random sampling have smaller coverage and fewer paths than that of the Latin hypercube sampling. Combined with Table 3, the aggregation degree of the simple random sampling is greater than that of the Latin hypercube sampling, but the coverage area is on the contrary.

The results show that the Latin hypercube sampling method is more beneficial to the construction of undirected maps.

In order to verify the performance of the proposed algorithm in a static environment, 6 groups of grid sequences are

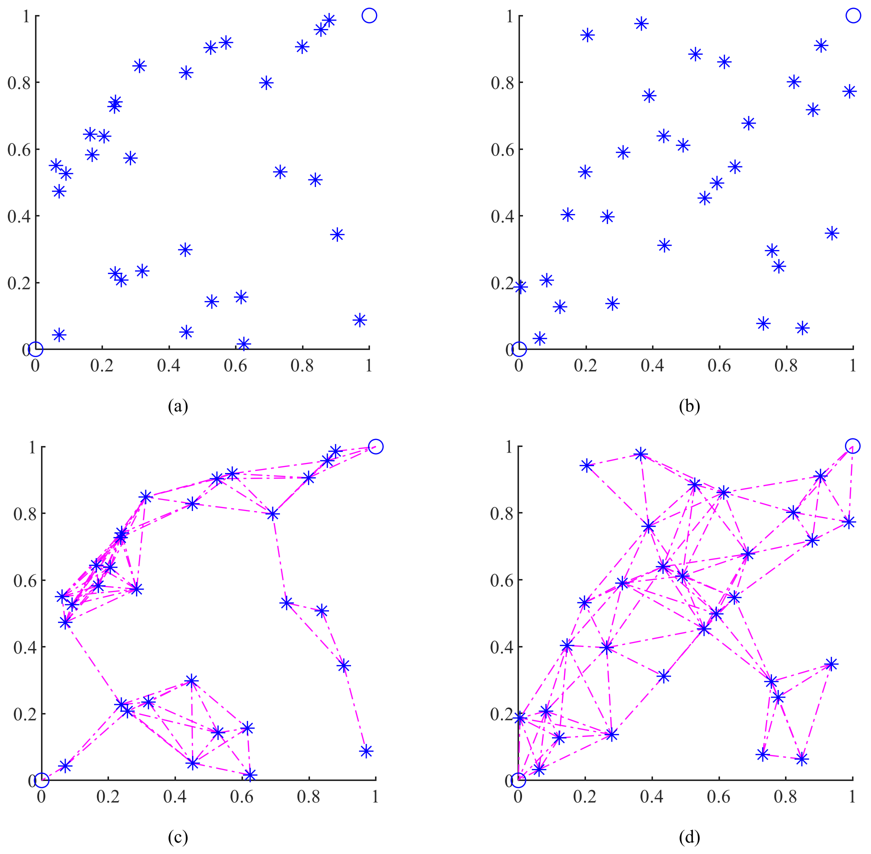

FIGURE 8. Sampling and constructing undirected maps in two-dimensional space.

TABLE 3. Performance comparison among the two methods.

	Aggregation degree	Coverage area
Simple random sampling	3.3761	0.4371
Latin hypercube sampling	2.9833	0.6793

randomly sampled in the task space to construct a static environment for the experiment. As shown in Fig. 9, the number of grids is 6, 9, 12, 15 and 21, and the environmental complexity of the 6 groups of the grid sequences increases successively. The random selection of grid sequences is on the premise that the selected grid sequences do not occupy the starting and ending points of the planning task. That is to say, there is a feasible path. The starting point of the task is set as ($\frac{\pi}{3}$, $-\pi$, $-\pi$, 0, 0, 0), and the ending point of the task as ($-\frac{2\pi}{3}$, $\frac{\pi}{6}$, $\frac{\pi}{6}$, $\frac{\pi}{2}$, $\frac{\pi}{6}$, π). For each group of grid sequences, the path is searched by the PRM [27], the Lazy PRM [28] and the Local PRM respectively. Each group of experiments is conducted for 100 times, and the average and median values are adopted. The statistics are shown in Table 4.

As shown in Table 4, with the increase of the environmental complexity, the time consumption of each algorithm presents an upward trend. However, the time consumption of the algorithm in this article is still the least among the three. The average time consumption of the proposed algorithm is 0.10% of PRM algorithm, and 20.55% of Lazy PRM algorithm. The median value is 0.08% of PRM algorithm, and

TABLE 4. Time consumption comparison among the three algorithms(s).

	PRM		Lazy PRM		Local PRM	
	avg	med	avg	med	avg	med
1	6.7927	6.7772	0.0264	0.0232	0.0108	0.0080
2	9.3077	9.3255	0.0790	0.0585	0.0093	0.0078
3	12.0766	11.9752	0.0316	0.0284	0.0032	0.0033
4	14.6775	14.6542	0.0283	0.0270	0.0045	0.0040
5	17.0907	17.0209	0.1179	0.1116	0.0287	0.0208
6	19.2776	19.1401	0.0940	0.0534	0.0210	0.0216
avg	13.2038	13.1489	0.0629	0.0504	0.0129	0.0109

21.68% of Lazy PRM algorithm. The results show that the proposed collision-detection strategy and the simplified collision detection can significantly reduce the time consumption of the algorithm.

Further, in order to verify whether the proposed algorithm is competent in dynamic environments and to investigate its performance in dynamic environments, the simulation environment is set as follows. The starting point of the task is ($-\frac{\pi}{3}$, $\frac{\pi}{6}$, $-\frac{7\pi}{3}$, 0, 0, 0), the ending point of the task is ($\frac{\pi}{3}$, $\frac{\pi}{6}$, $-\frac{2\pi}{3}$, 0, $-\frac{\pi}{2}$, 0), and the initial position of the dynamic obstacle with a radius of 0.0667 is ($\frac{3}{5}$, $-\frac{1}{3}$, $-\frac{2}{15}$), which can move along the Z-axis with a certain probability.

Due to the excessive time consumption of the PRM algorithm, the Lazy PRM algorithm and the Local PRM algorithm are used to carry out the simulation of path planning for a dynamic environment. A total of 10 groups are carried out.

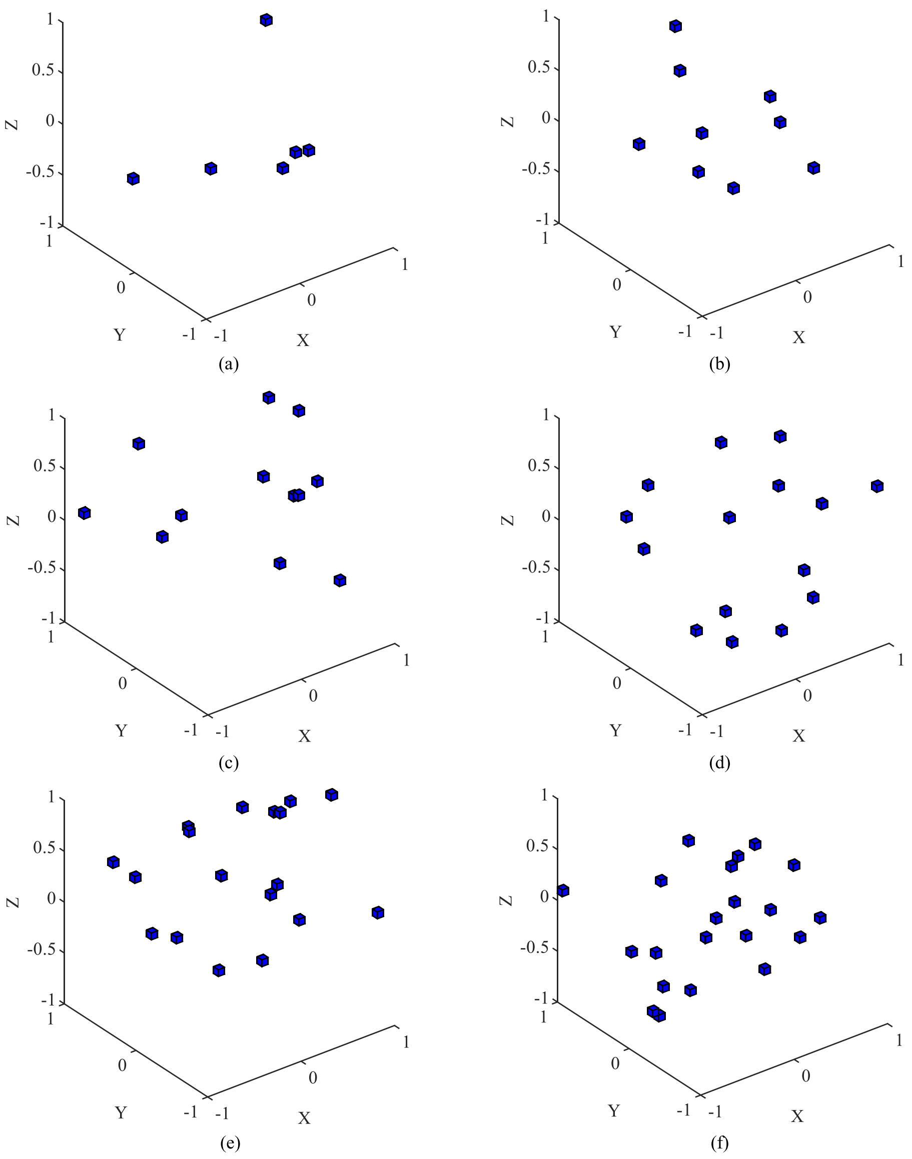

FIGURE 9. Obstacle grids in a static environment.

TABLE 5. Time consumption Comparison among the two algorithms(s).

	Lazy PRM			Local PRM		
	Period	Total Time	Single Time	Period	Total Time	Single Time
1	84	1.9600	0.0233	104	0.7921	0.0076
2	85	2.0615	0.0243	71	0.7068	0.0100
3	77	1.9471	0.0253	71	0.7306	0.0103
4	176	3.2763	0.0186	99	0.7233	0.0073
5	131	2.6857	0.0205	75	0.7068	0.0094
6	133	2.7317	0.0205	133	0.7664	0.0058
7	124	2.5711	0.0207	114	0.7366	0.0065
8	98	2.1993	0.0224	125	0.7584	0.0061
9	116	2.5040	0.0216	90	0.7155	0.0079
10	106	2.5369	0.0239	135	0.7571	0.0056

The statistical data are shown in Table 5, where "Period" is the number of planning periods.

As shown in Table 5, since the movement of obstacles is random, the difficulty of planning is different, which is shown in the table as the difference of Total Time. The more times the algorithm is reprogrammed, the greater the value will be.

However, in general, the time consumption of the algorithm in this article is less than that of the comparison algorithm. Single time is the average time consumption of the algorithm in a single period, and the average values of multiple groups are 0.0221s and 0.0076s respectively. The single-step time consumption of the proposed algorithm is 34.55% of that of the contrast algorithm. The results show that the dynamic performance of the proposed algorithm is better than that of the contrast algorithm, and the Local PRM algorithm can be used in dynamic environments.

The 10th group of data is selected to illustrate the planning process of the algorithm in this article. There are 135 planning cycles in the path planning process, which generate a path composed of 6 straight lines. As shown in Fig.10, Fig. 10a is

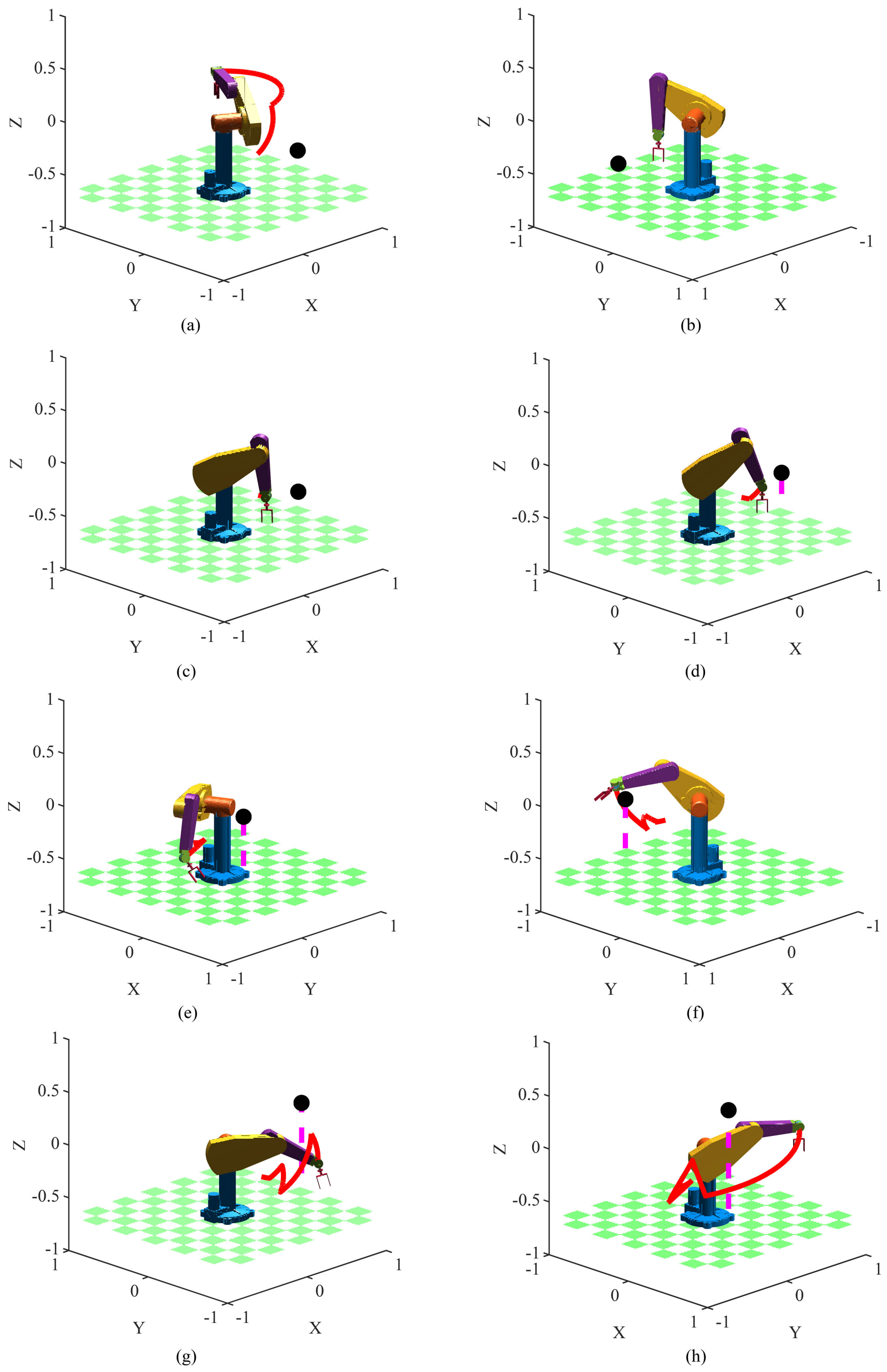

FIGURE 10. Path planning of the manipulator.

the path of the fixed obstacle; Fig. 10b, Fig. 10c, Fig. 10d, Fig. 10e, Fig. 10f, Fig. 10g and Fig. 10h are the key positions of the path of the obstacle, in which the pink track is the path of the moving obstacle, and the red track is the trajectory of the end-of-arm tools (EOAT). The positions of key points at the EOAT are shown in Table 6. The changes of the position of obstacles in the process are shown in Table 7, which lists the planning period for the change of the position of obstacles. For the period not listed, the position of obstacles remains unchanged.

During the period 1 to 5, it is judged that the changes of the positions of the obstacles will not collide with the manipulator, which moves to the key point 5. During the period 6 to 16, it is judged that the changes of the positions of the obstacles will also not collide with the manipulator, which moves to the key point 16. During the period 17, it is judged

TABLE 6. Positions of key points in the path of the manipulator(rad).

	1	2	3	4	5	6
1	-1.0472	0.5236	-3.6652	0	0	0
5	-0.9971	0.4818	-3.5001	-0.0049	-0.0913	0.0120
16	-0.7933	0.6645	-3.4107	-0.1543	-0.3947	-0.0254
48	-1.1433	0.1951	-2.9688	0.9714	-0.7764	0.6591
66	-0.7637	0.7024	-2.5169	1.3239	-0.7544	0.3769
97	-0.6197	0.1668	-2.2557	-0.0418	-0.7797	0.1021
135	1.0472	0.5236	-2.0944	0	1.5708	0

TABLE 7. Changes of the positions of obstacle(m).

	Z		Z		Z		Z
1	-0.3333	17	-0.0667	67	0.2000	122	0.4667
6	-0.2667	21	0.0000	84	0.2667	126	0.5333
13	-0.2000	23	0.0667	87	0.3333	129	0.6000
15	-0.1333	46	0.1333	109	0.4000	135	0.6000

that the changes of the positions of the obstacles will collide with the scheduled path of the manipulator. The algorithm should reprogram a path and the manipulator moves to the key point 48, during which the changes of the positions of the obstacles have no effect on the scheduled path. During the period 49 to 66, it is judged that the changes of the positions of the obstacles will not collide with the manipulator, which moves to the key point 66. During the period 67, it is judged that the changes of the positions of the obstacles will collide with the scheduled path of the manipulator. The algorithm should reprogram a path and the manipulator moves to the key point 97, during which the changes of the positions of the obstacles have no effect on the scheduled path. During the period 98 to 135, it is judged that the changes of the positions of the obstacles will not collide with the manipulator, which moves to the key point 135. And so on until the manipulator reaches the goal. In other words, in the planning period 17 and 67, the algorithm implements avoidance behaviors twice for the moving obstacles, and realizes the dynamic path planning.

By comparing the differences of the path planning between the static environment and the dynamic environment, it can be seen that the proposed method can adjust the global path in real time according to the changes of environment, to obtain non-collision path. Obviously, the PRM algorithm can be used in dynamic environments.

V. CONCLUSION

A Grid-Local PRM algorithm is proposed which using mapping model, sampling strategies, lazy collision detection and single local detection method. The simulation results of the collision detection show that the performance of the proposed simplified collision-detection algorithm is much better than that of the standard collision-detection algorithm, and the stability is far better than that of the standard collision-detection algorithm. The sampling experiment in two-dimensional space also shows that the stratified sampling is beneficial to the generation of probability map. The simulation experiment of the path planning shows that the proposed algorithm consumes an average of 10ms in the static environment and 7ms in single step in the dynamic environment, which is better than the comparison algorithm. The proposed algorithm can adjust the global path in real time to avoid obstacles according to environmental changes, which have been verified to be feasible and effective. The proposed method can implement dynamic path planning and can be used for dynamic obstacle avoidance of manipulators. In this article, the problem of path planning in dynamic environment has been investigated, which has solved the problem that PRM algorithm is difficult to be applied in it to a certain extent. In the future, the dynamic trajectory planning and trajectory tracking controller design of the manipulator will be expanded.

REFERENCES

[1] X. Wang, B. Tang, and X. Gu, "Research on obstacle avoidance strategy for welding robot," *J. Mech. Eng.*, vol. 55, no. 17, pp. 77–84, Sep. 2019, doi: 10.3901/jme.2019.17.077.

[2] L. Xie and S. Liu, "Dynamic obstacle-avoiding motion planning for manipulator based on improved artificial potential filed," *Control Theory Appl.*, vol. 35, no. 9, pp. 1239–1249, 2018, doi: 10.7641/CTA.2018.70187.

[3] J. Zhao, Z. Zhang, Q. Zhang, D. Chen, and S. Gui, "Research status and development trend of robot safety," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 44, no. 7, pp. 1347–1358, 2018, doi: 10.13700/j.bh.1001-5965.2017.0568.

[4] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robot. Autom.*, vol. RA-3, no. 3, pp. 224–238, Jun. 1987, doi: 10.1109/JRA.1987.1087095.

[5] W. S. Newman and M. S. Branicky, "Real-time configuration space transforms for obstacle avoidance," *Int. J. Robot. Res.*, vol. 10, no. 6, p. 650, 1991, doi: 10.1177/027836499101000605.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439.

[7] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *Int. J. Robot. Res.*, vol. 21, no. 12, p. 999, 2002, doi: 10.1177/0278364902021012001.

[8] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. Millennium Conf., IEEE Int. Conf. Robot. Automat. Symp. (ICRA)*, vol. 1, Apr. 2000, pp. 521–528, doi: 10.1109/ROBOT.2000.844107.

[9] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *J. Intell. Robot. Syst.*, vol. 84, nos. 1–4, pp. 829–858, Dec. 2016, doi: 10.1007/s10846-016-0362-z.

[10] N. Wang and H. Xu, "Dynamics-constrained global-local hybrid path planning of an autonomous surface vehicle," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 6928–6942, Jul. 2020, doi: 10.1109/TVT.2020.2991220.

[11] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm," *Sensors*, vol. 18, no. 2, p. 571, 2018, doi: 10.3390/s18020571.

[12] A. Rahmani and S. Faroughi, "Application of a novel elimination algorithm with developed continuation method for nonlinear forward kinematics solution of modular hybrid manipulators," *Robotica*, vol. 38, no. 11, pp. 1963–1983, Nov. 2020, doi: 10.1017/S0263574719001747.

[13] S. Sanhan, W. Sanhan, and C. Mongkolkeha, "New existence of fixed point results in generalized pseudodistance functions with its application to differential equations," *Mathematics*, vol. 6, no. 12, p. 324, Dec. 2018, doi: 10.3390/math6120324.

[14] O. Sim, J. Oh, K. K. Lee, and J.-H. Oh, "Collision detection and safe reaction algorithm for non-backdrivable manipulator with single force/torque sensor," *J. Intell. Robot. Syst.*, vol. 91, nos. 3–4, pp. 403–412, Sep. 2018, doi: 10.1007/s10846-017-0695-2.

[15] S. Zhang and Y. Xia, "Solving nonlinear optimization problems of real functions in complex variables by complex-valued iterative methods," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 277–287, Jan. 2018, doi: 10.1109/TCYB.2016.2632159.

[16] H. Liu, G. Kang, and K. Li, "MIMO UWB antenna based on differential evolution algorithm," *J. South China Univ. Technol., Natural Sci. Ed.*, vol. 48, no. 3, pp. 24–31, 2020, doi: CNKI:SUN:HNLG.0.2020-03-004.

[17] J. Ma, Y. Wang, Y. He, K. Wang, and Y. Zhang, "Motion planning of citrus harvesting manipulator based on informed guidance point of configuration space," *Trans. Chin. Soc. Agricult. Eng.*, vol. 35, no. 8, pp. 100–108, 2019, doi: CNKI:SUN:NYGU.0.2019-08-012.

[18] A. L. Madsen, "Variations over the message computation algorithm of lazy propagation," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 36, no. 3, pp. 636–648, Jun. 2005, doi: 10.1109/TSMCB.2005.862488.

[19] S. S. Politis, Z. Zhang, Z. Han, J. J. Hasenbein, and M. Arellano, "Stochastic analysis of network-level bridge maintenance needs using Latin hypercube sampling," *ASCE-ASME J. Risk Uncertainty Eng. Syst., A, Civil Eng.*, vol. 7, no. 1, Mar. 2021, Art. no. 04020049, doi: 10.1061/AJRUA6.0001099.

[20] H. Samawi, A. Chatterjee, J. Yin, and H. Rochani, "On quantiles estimation based on different stratified sampling with optimal allocation," *Commun. Statist.-Theory Methods*, vol. 48, no. 6, pp. 1529–1544, Mar. 2019, doi: 10.1080/03610926.2018.1433856.

[21] H. Wang, P. Yin, W. Zhang, H. Wang, and J. Zuo, "Mobile robot path planning based on improved A* algorithm and dynamic window method," *Robot*, vol. 42, no. 3, pp. 346–353, 2020, doi: 10.13973/j.cnki.robot.190305.

[22] D. Han, H. Nie, J. Chen, and M. Chen, "Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection," *Robot. Comput.-Integr. Manuf.*, vol. 49, pp. 98–104, Feb. 2018, doi: 10.1016/j.rcim.2017.05.013.

[23] P. Corke, "MATLAB toolboxes: Robotics and vision for students and teachers," *IEEE Robot. Autom. Mag.*, vol. 14, no. 4, pp. 16–17, Dec. 2007, doi: 10.1109/M-RA.2007.912004.

[24] L. Huashan, Z. Wuneng, L. Xiaobo, and Z. Shiqiang, "An efficient inverse kinematic algorithm for a PUMA560-structured robot manipulator," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 5, pp. 1–5, May 2013, doi: 10.5772/56403.

[25] A. Izadbakhsh and P. Kheirkhahan, "On the voltage-based control of robot manipulators revisited," *Int. J. Control, Autom. Syst.*, vol. 16, no. 4, pp. 1887–1894, Aug. 2018, doi: 10.1007/s12555-017-0035-0.

[26] H. Touzani, H. Hadj-Abdelkader, N. Seguy, and S. Bouchafa, "Multi-robot task sequencing & automatic path planning for cycle time optimization: Application for car production line," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1335–1342, Apr. 2021, doi: 10.1109/lra.2021.3057011.

[27] J.-J. Park, H.-S. Kim, and J.-B. Song, "Collision-free path planning for a redundant manipulator based on PRM and potential field methods," *J. Inst. Control, Robot. Syst.*, vol. 17, no. 4, pp. 362–367, Apr. 2011, doi: 10.5302/J.ICROS.2011.17.4.362.

[28] H. Akbaripour and E. Masehian, "Semi-lazy probabilistic roadmap: A parameter-tuned, resilient and robust path planning method for manipulator robots," *Int. J. Adv. Manuf. Technol.*, vol. 89, nos. 5–8, pp. 1401–1430, Mar. 2017, doi: 10.1007/s00170-016-9074-6.

YOUYU LIU was born in Tongcheng, Anhui, China, in 1976. He received the B.S. and M.S. degrees from the Lanzhou University of Technology, Lanzhou, China, in 2004, and the Ph.D. degree in mechanical engineering from the Hefei University of Technology, Hefei, China, in 2015. Since June 2004, he has been working with Anhui Polytechnic University, Wuhu, China. Since 2018, he has been a Professor. He has published more than 50 articles. His current research interests include path planning of manipulators, robot safety, and smart manufacturing systems.

BO CHEN was born in Huanggang, Lu'an, Anhui, China, in 1996. He received the bachelor's degree in mechanical design, manufacturing and automation from Suzhou University, in 2019. He is currently pursuing the master's degree in mechanical engineering with Anhui Polytechnic University. His research interests include robot and robotics static and dynamic complex environment obstacle avoidance process and its applications.

XUYOU ZHANG was born in Hefei, Anhui, China, in 1995. He received the B.S. degree in mechanical engineering from Hefei University, in 2018, and the master's degree from Anhui Polytechnic University. His research interests include perception of robots and obstacle avoidance control of robots.

RENJUN LI was born in Dali, Shanxi, China, in 1974. He received the B.S. degree from the China University of Petroleum (East China), Dongying, China, in 1996, and the M.S. and Ph.D. degrees in mechanical engineering from the Xi'an University of Technology, Xi'an, China, in 2009. Since July 2009, he has been working with Anhui Polytechnic University, Wuhu, China. Since 2012, he has been an Associate Professor. He has published more than 20 articles. His current research interests include robot mechanism and robot safety.