RESEARCH ON ROBOT MANIPULATOR PATH PLANNING BASED ON DELAUNAY MAP

Jiali PI[1]*, Weiming ZHANG[1], Dongmei ZHANG[1], Xueqing LI[2], Binbin YU[3]
[1]*Army Logistics University of PLA, Chongqing, 401331, China*
[2]*Sichuan International Studies University, Chongqing, 400031, China*
[3]*91943 troops of PLA, Handan 056000, China*
E-mail: pijiali1991@163.com

***ABSTRACT**: This paper aims to find an optimal obstacle-free path for robot manipulator. Therefore, the global planning and the local planning were combined into an organic whole. First, the Monte-Carlo method was adopted to construct the Delaunay tetrahedral grid map. Then, the Floyd algorithm was employed to find the path of the collision-free global shortest path of the Cartesian space in the Delaunay map, that is, develop an obstacle avoidance path planning algorithm was developed for robot manipulator. Numerical simulations and experiments show that the path planning algorithm can effectively optimize the path of the robot manipulator in the Cartesian space and the joint angular space. Suffice it to say that the algorithm has great application potential in the robot manipulator control.*

***KEYWORDS**: robot manipulator, space path planning, map construction*

1 INTRODUCTION

The manufacturing assembly is a resource-intensive operation. It consumes lots of manpower and time. A possible way to improve the situation lies in the development automatic robot assembly. Nevertheless, it is still difficult to apply robot assembly technology in actual production, due to the complex operation path and the lack of accurate and efficient path planning ability of existing robot manipulators. This calls for the design of a new path planning algorithm that can enhance the rationality, timeliness and adaptability of robot manipulator path planning.

In the movement space of a robot manipulator, the path planning problem can be described as finding the optimal or close-to-optimal path from the initial state to the target state, provided that the robot manipulator will not be stuck by any obstacle. There are mainly three kinds of methods for robot manipulator path planning:

First, the algorithms based on artificial intelligence algorithms, namely ant colony algorithm (Shi et al., 2014), fuzzy algorithm (Chen and Zhu, 2011), fish swarm algorithm (Liang et al., 2016; Xu and Zhu, 2012) and genetic algorithm (Keshtkar, 2017; Qi et al., 2014; Gómez-Bravo et al., 2012; Wang and Xie, 2016).

Second, the algorithms based on objective function optimization, including gradient projection method (Fang and Zhao, 2010), artificial potential field method (Wang et al., 2015), sequential quadratic programming (nonlinear optimization) (Rubio et al., 2016), and parameter manipulability optimization (Lee and Song, 2016).

The above path planning algorithms are featured by heavy computation, high complexity and poor portability. In particular, the first type of algorithms need to correct their parameters based on samples, and cannot avoid the trap of local optimum.

Third, the algorithms based on grid maps and moving rules, such as Floyd algorithm, A~* algorithm, and Q learning algorithm (Qian et al., 2015). Among them, the Floyd algorithm is a global optimization algorithm with global optimal features. The advantage is that the shortest path obtained by the algorithm must be the shortest path of the Cartesian space in the whole grid (Zhang et al., 2017).

In this paper, the global planning and the local planning were combined into an organic whole. First, the Monte-Carlo method was adopted to construct the Delaunay tetrahedral grid map (the Delaunay map). Then, the Floyd algorithm was employed to find the path of the collision-free global shortest path of the Cartesian space in the Delaunay map.

2 MAPPING RELATIONSHIP BETWEEN THE CARTESIAN SPACE AND THE JOINT ANGULAR SPACE OF THE ROBOT MANIPULATOR

The Delaunay map was built on both the Cartesian space and the joint angular space by forward kinematic equation ${}_0^4T$ and inverse kinematics equation *inverse* (*x*, *y*, *z*). The robot

manipulator has 5 degrees-of-freedom (DOF). The D-H coordinates and parameters of the manipulator are shown in Figure 1 and Table 1, respectively.

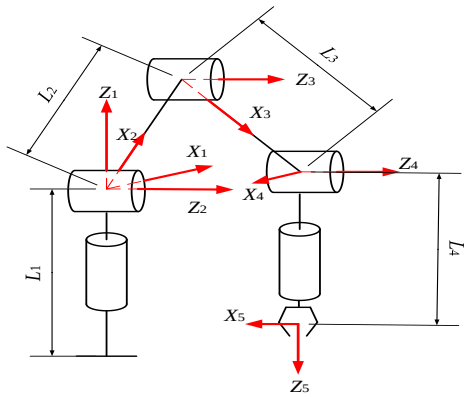

Figure 1. Kinematic scheme of 5 DOF manipulator

Table 1. Denavit-Hartenberg parameters of the 5 DOF manipulator in Figure 1.

i	Twist angle α_{i-1}	Link length a_{i-1}	Link offset d_i	Joint angle θ_i
1	0	0	L_1	θ_1
2	π/2	0	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	θ_4
5	-π/2	0	L_4	θ_5

The D-H transform equation is generally expressed as:

$$
{}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \cos\alpha_{i-1}\sin\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\alpha_{i-1}\sin\theta_i & \sin\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

The forward kinematics equation of the robot manipulator was obtained by multiplying the D-H equation of each link transformation:

$$
{}^{4}_{0}T = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 0 \end{bmatrix}.
$$

where:

$x = \cos(\theta_1)(L_3\cos(\theta_2+\theta_3) + L_2\cos(\theta_2) - L_4\cos(\theta_2+\theta_3+\theta_4))$,

$y = \sin(\theta_1)(L_3\cos(\theta_2+\theta_3) + L_2\cos(\theta_2) - L_4\cos(\theta_2+\theta_3+\theta_4))$, and

$z = L_1 + L_3\sin(\theta_2+\theta_3) + L_2\sin(\theta_2) - L_4\cos(\theta_2+\theta_3+\theta_4)$.

The forward kinematics equation represents the mapping relationship between the joint angular space and the Cartesian space. Then, the inverse kinematic equations were derived from the forward kinetic equation ${}^{4}_{0}T$. The link transformation matrix L_1 was moved to the left of the forward kinematics equation, ${}^{4}_{0}T = {}^{1}_{0}T\,{}^{2}_{1}T\,{}^{3}_{2}T\,{}^{4}_{3}T$. Since the (2, 4) elements on both sides of the equation are equal, $0 = (p_y c_1 - p_x s_1)/(2c_1^2 - 1)$.

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) \quad (1)$$

Next, the link transformation matrices L_2 and L_3 were moved to the left side of the forward kinematics equation. Sixteen elements of equal values were found by comparing the left and right sides of the equation: $\cos(\theta_i)$ and $\sin(\theta_i)$ are written in c_i and s_i, $\cos(\theta_i+\theta_j)$ and $\sin(\theta_i+\theta_j)$ are written in c_{ij} and s_{ij}, and so on. Hence, the right side of the equation can be written as:

$$
\begin{bmatrix} c_{234} & s_{234} & 0 & L_3c_{23} + L_2c_2 \\ 0 & 0 & -1 & 0 \\ s_{234} & c_{234} & 0 & L_3s_{23} + L_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

The left side of the equation was divided into two sub-matrices:

$$
\begin{bmatrix} r_{11}c_1c_5 - r_{12}c_1s_5 + r_{21}c_5s_1 - r_{22}s_1s_5 & r_{13}c_1 + r_{23}s_1 \\ r_{21}c_1c_5 - r_{11}c_5s_1 - r_{22}c_1s_5 + r_{12}s_1s_5 & r_{23}c_1 - r_{13}s_1 \\ r_{31}c_5 - r_{32}s_5 & r_{33} \\ 0 & 0 \end{bmatrix}
$$

$$
\begin{bmatrix} -r_{12}c_1c_5 - r_{12}c_1s_5 - r_{22}c_5s_1 - r_{21}s_1s_5 & p_xc_1 + p_ys_1 - L_4r_{23}s_1 - L_4r_{13}c_1 \\ r_{12}c_5s_1 - r_{22}c_1c_5 - r_{21}c_1s_5 + r_{11}s_1s_5 & p_yc_1 - p_xs_1 + L_4r_{13}s_1 - L_4r_{23}c_1 \\ -r_{32}c_5 - r_{31}s_5 & p_z - L_1 - L_4r_{33} \\ 0 & 1 \end{bmatrix}
$$

The following equations can be obtained by equalizing (3, 3) elements on both sides of the matrix.

Solving $0 = -r_{32}c_5 - r_{31}s_5$ yields

$$\theta_5 = -\tan^{-1}\frac{r_{32}}{r_{31}} \quad (2)$$

According to the equation on the left and right sides of the matrix elements in the fourth column, two equations were sorted out as:

$$L_3c_{23} + L_2c_2 = p_xc_1 + p_ys_1 - L_4r_{23}s_1 - L_4r_{13}c_1 \quad (3)$$

$L_3 s_{23} + L_2 s_2 = p_z - L_1 - L_4 r_{33}$
(4)

With the sum of two equations' square, it is found that θ_3 can be solved. Then, θ_3 was substituted into any equation to yield θ_2. θ_3 and θ_2 were substituted into to yield θ_4. At this point, 5 inverse kinematics equations (1-5) $inverse(x, y, z)$ were derived to depict the mapping relationship of robot manipulator between the Cartesian space and the joint angular space.

$c_{234} = r_{33}$
(5)

3 OBSTACLE COLLISION DETECTION

Following the concept of "arm plane" proposed by Kreutz-Delgado et al., the arm plane was defined with such three points as the shoulder joint, the elbow joint and the wrist joint (Jiang et al., 2013) (Figures 2~3). In this way, the collision detection problem was transformed into an obstacle intersection problem in the arm plane. The intersection of the robot manipulator and the obstacle is essentially the polyline intersection of 2 planes (Dong et al., 2017)

Figure 3 shows the polylines in two planes, i.e. the obstacle plane and the arm plane. If the polylines of the 2 planes intersect with each other, it means the robot manipulator has collided into an obstacle. Then, a four-step process was executed to calculate the intersection line between the two planes, determine if an intersection occurs by rapid rejection test and cross test, and identify the intersection position by collision point calculation.

(1) Determination of polylines on the obstacle plane

The obstacle surface was approximated by a number of planes to find the polylines $q_1 q_2$according to the simultaneous equations of the obstacle plane and the robot arm plane. Similarly, the polylines on the robot arm plane $p_1 p_2$ were also obtained.

(2) Rapid rejection test

Let the rectangle with $p_1 p_2$ as the diagonal line be R_1, and the rectangle with $q_1 q_2$ as the diagonal line be R_2. If there is no intersection between $R_1$and R_2, the two lines will not have an intersection point (Figure 4).

(3) Cross test

As shown in

Figure 4, the intersection means $p_1 p_2$ and $q_1 q_2$ must cross each other. The vector cross product determines whether the two lines cross by its geometric meaning. If $p_1 p_2$ crosses $q_1 q_2$, the vectors $\overrightarrow{q_1 p_1}$ and $\overrightarrow{q_1 p_2}$ are on both sides of vector $\overrightarrow{q_1 q_2}$, namely: $\overrightarrow{q_1 p_1} \times \overrightarrow{q_1 q_2} \cdot \overrightarrow{q_2 p_1} \times \overrightarrow{q_1 q_2} < 0$

If$\overrightarrow{q_1 p_1} \times \overrightarrow{q_1 q_2} = 0$, $p_1 p_2$ and $q_1 q_2$ are collinear. Similarly, the intersection of $p_1 p_2$ and $q_1 q_2$ can be judged by: $\overrightarrow{p_1 q_1} \times \overrightarrow{p_1 p_2} \cdot \overrightarrow{q_2 p_1} \times \overrightarrow{p_1 p_2} < 0$

(1) Collision point calculation

Let us denote the plane determined by $p_1 p_2$ and $q_1 q_2$ as plane Π. Among the planes that are perpendicular to plane Π, the one passing through $p_1 p_2$ is denoted as plane Π_1, and the one passing through $q_1 q_2$ is denoted as plane Π_2. If there is a common point for the three planes, it means the three planes intersect each other, and that $p_1 p_2$ crosses $q_1 q_2$. Then, the collision point can be obtained by the equations of these planes.

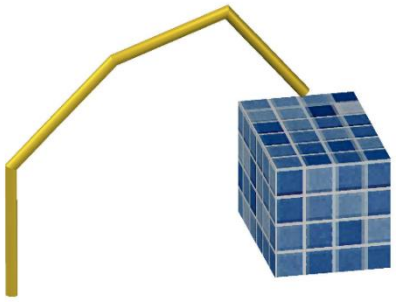

Figure 2. Three-dimensional model of manipulator obstacle

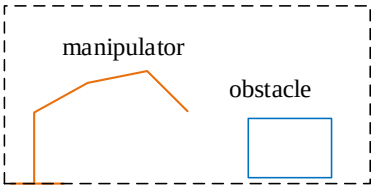

Figure 3. Projection Model of Robot manipulator and obstacle in Arm Plane

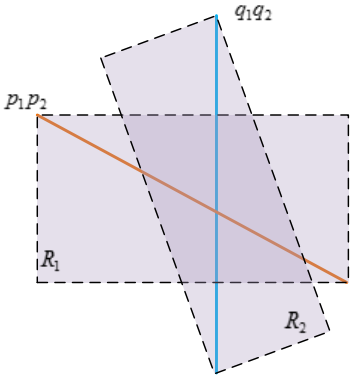

Figure 4. A case of rapid rejection test and crossover test

4 MAP BUILDING FOR PATH PLANNING

To build a map for path planning, the surrounding area of the robot must be divided into free movement space and restricted space. The popular map building methods include road marking and grid meshing (Zhu and Yan, 2010). To find a feasible path map for finding the shortest path in space, the working space of the robot manipulator should be converted to the feasible path in light of the manipulator model and obstacle model (Jia, 2010).

Monte-Carlo point cloud data generation

The random point cloud data generated by the Monte-Carlo method can directly map the points in the joint angular space to the Cartesian space. If the cloud is dense enough, the manipulator workspace can be well fitted. Instead of the nonlinear inverse kinematics equation, the forward kinematics equation was relied on to easily obtain the position data of each discrete point in the joint angular space and the Cartesian space (Tian et al., 2013).

Focusing on each joint angle feasible region $(\theta_{\min}, \theta_{\max})$, the point cloud data were generated based on the uniform distribution of random number function $unifrnd(\theta_{\min}, \theta_{\max}, n)$ of the Monte-Carlo algorithm. The joint point set $\{\theta_1, \theta_2, \theta_3, \theta_4\}$ was generated in the joint angular space and substituted into the D-H forward kinematics equation ${}_0^4T$ to get the Cartesian coordinates of the manipulator set $\{x, y, z\}$.

These random discrete values are the point of the Cartesian workspace of the robot manipulator combined with the angle of the random discrete point in the joint angular space.

As shown in Figure 5, the robot manipulator often does not collide into obstacles at the start of the operation. In this case, the detection plane perpendicular to the arm plane should be placed at the middle of the obstacle and the manipulator (Liu and Wang, 1996). Once the manipulator passes through the detection plane, it is necessary to determine if there is a collision by geometric method. The detection plane is parallel to the tangent plane of the obstacle, and can be expressed as $ax + by + c = 0$. According to

Figure 6, d is the distance limit of the detection plane from the obstacle, and the length of the 3 connecting links is $L_2 + L_3 + L_4$. Assuming that the connecting links bend as the manipulator moves to the tangent plane of the obstacle, the latter will move by d along the normal vector direction. Then, the detection plane equation is $plane_{test}$: $a\left(x + \frac{a}{\sqrt{a^2+b^2}}d\right) + b\left(y + \frac{b}{\sqrt{a^2+b^2}}d\right) + \mathrm{c} = 0$

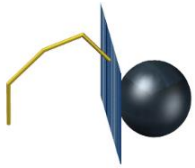

Figure 5. Three-dimensional schematic diagram of the detection plane

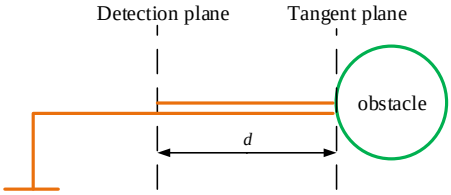

Figure 6. Three-dimensional Schematic Diagram of Detection Plane

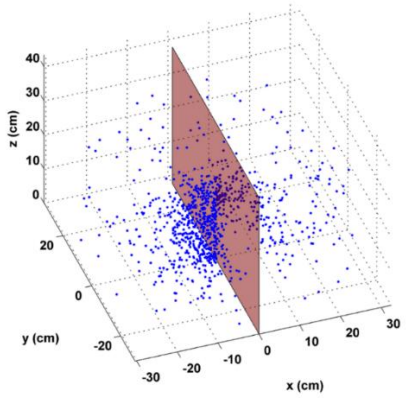

Figure 7. Monte-Carlo points cloud data

As shown in

Figure 7, the detection plane divides the point cloud into two parts. On the collision side to be detected by the detection plane, the obstacle collision detection function $collision(\theta_1, \theta_2, \theta_3, \theta_4)$ determines the feasibility of the discrete point of the point cloud data. If the function is valid at the discrete point, the robot manipulator will collide into the obstacle during the movement, and the discrete point should be removed. By substituting the filtered discrete joint points $\{\theta_1, \theta_2, \theta_3, \theta_4\}$ into equation ${}_0^4T$, the discrete point cloud can be obtained as:

$$\mathrm{Monte\_Carlo}\{(x, y, z)|p(\theta_1, \theta_2, \theta_3, \theta_4)\}.$$

Construction of Delaunay tetrahedral grid map based on point cloud data

As its name suggests, the Delaunay tetrahedral map is created by Delaunay tetrahedrons. It is a

feasible path map of connecting lines between the path points. Based on point cloud data, the map can approximate the original workspace with accurate grids, fine structure, low redundancy, high storage efficiency and convenience in update. The rule of Delaunay tetrahedral partition is easy and effective. Each tetrahedron is set up by four closest points, and do not intersect with each other. The even meshing of the cloud data lays a solid basis for Floyd path planning algorithm.

For a 2D plane, the Delaunay triangular map is often generated by point-by-point insertion and incremental diffusion. The point-by-point insertion has been implemented extensively, thanks to its time and space efficiency. For instance, the method is incorporated into Matlab to compute Delaunay division for triangular network generation on the discrete cloud. The point-by-point insertion is implemented in the following steps. First, all the discrete points are traversed to find the set of convex hulls, and the outermost edge is computed by the Qhull algorithm (Xiao et al., 2010). Then, the triangular grids are generated by incremental insertion in convex hulls. After that, the discrete points are inserted into the new triangles in turn, according to the rule of Delaunay circumcircle/circumsphere. In this way, the Delaunay triangular map is gradually formed (Dong, 2005).

In this research, the 2D Delaunay triangulation was promoted to 3D Delaunay tetrahedralization. First, the internal structure was also subdivided, but into tetrahedrons. The tetrahedral subdivision algorithm is similar to the 2D Delaunay trigonometric algorithm. The new vertex was gradually added to the set of spatial points before applying the rule of Delaunay circumcircle / circumsphere. When a new point was added to the Delaunay grid, the new elements whose circumcircle passes through the point were removed to form a new section. In this way, a Delaunay tetrahedral grid map was formed eventually (Figure 8). Then, the vertices of the tetrahedrons in the Delaunay map were numbered, and the map of feasible paths for robot manipulator space $Delaunny(x, y, z)$ was formed (Figure 9).

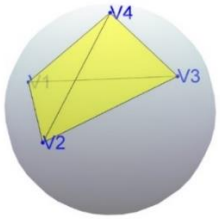

Figure 8. Delaunay Triangle cell grid

5 OBSTACLE AVOIDANCE PATH PLANNING ALGORITHM

The path planning algorithm is often implemented by path search in the Cartesian space. The inverse kinematics equation $inverse(x, y, z)$ should be called. However, the path search has to deal with a gigantic amount of data, and may fall into the dead loop. To solve the problem, the Floyd algorithm was introduced to optimize the individual path points, and realize the coarse and fine adjustments of the obstacle at different positions of the manipulator.

Floyd algorithm based on Delaunay grid map

Based on the Delaunay grid map, the Floyd method can compute the joint angle data of the path point directly from point cloud $\text{Monte\_Carlo}\{(x, y, z)|p(\theta_1, \theta_2, \theta_3, \theta_4)\}$, rather than call the inverse kinematics equation multiple times. In this way, the method greatly reduces the complexity and computing load of the path planning for the robot manipulator.

The Delaunay grid transforms the discrete cloud into a geometric body of tetrahedrons. Concerning the distance matrix $dist$ of tetrahedral edge length, the Floyd dynamic programming aims to find the starting point and the target point. Let $length_k[i](j)$ be the length of the shortest path from starting point i to target point j, with k being any tetrahedral vertex passing through the path. For any k> 0, there are two possible reasons that the shortest path between i and j does not exceed k: the path either contains the intermediate vertex k or does not contain the vertex. In the first case, the path length equals $length_{k-1}[i][k] + length_{k-1}[k](j)$. In the second case, the path length equals $length_{k-1}[i][k] + length_{k-1}[k](j)$. Following this train of thought, the Floyd dynamic programming algorithm was established with array $length_k[i](j)$ being the shortest path length in the iterative process (

Figure 9). The iteration formula is as follows:

$$\begin{cases} length_0[i][j] = dist_{ij} \\ length_k[i][j] = \min\{length_{k-1}[i][j], length_{k-1}[i][j] + length_{k-1}[i][j]\} \ 0 \le k \le n-1 \end{cases}$$

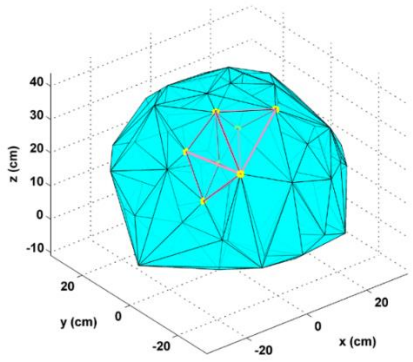

Figure 9. Delaunay tetrahedral mesh map

Then, the shortest path between the starting and target points was sought for on the Delaunay tetrahedral grid map, and the joint angle of all path points were obtained by the Monte-Carlo point cloud Monte_Carlo$\{(x,y,z)|p(\theta_1,\theta_2,\theta_3,\theta_4)\}$.

Path point optimization algorithm

The planned path points must differ greatly in joint angle, due to the random generation of the angle by Monte-Carlo method, and the non-unique mapping relationship between the Cartesian space and the joint angular space. Thus, the inverse kinematics equation was applied to optimize the joint angle with the minimum sum of adjacent joint angle difference e_1^i and the target point joint angle difference e_2^i. The algorithm is introduced in details below.

Path refining algorithm

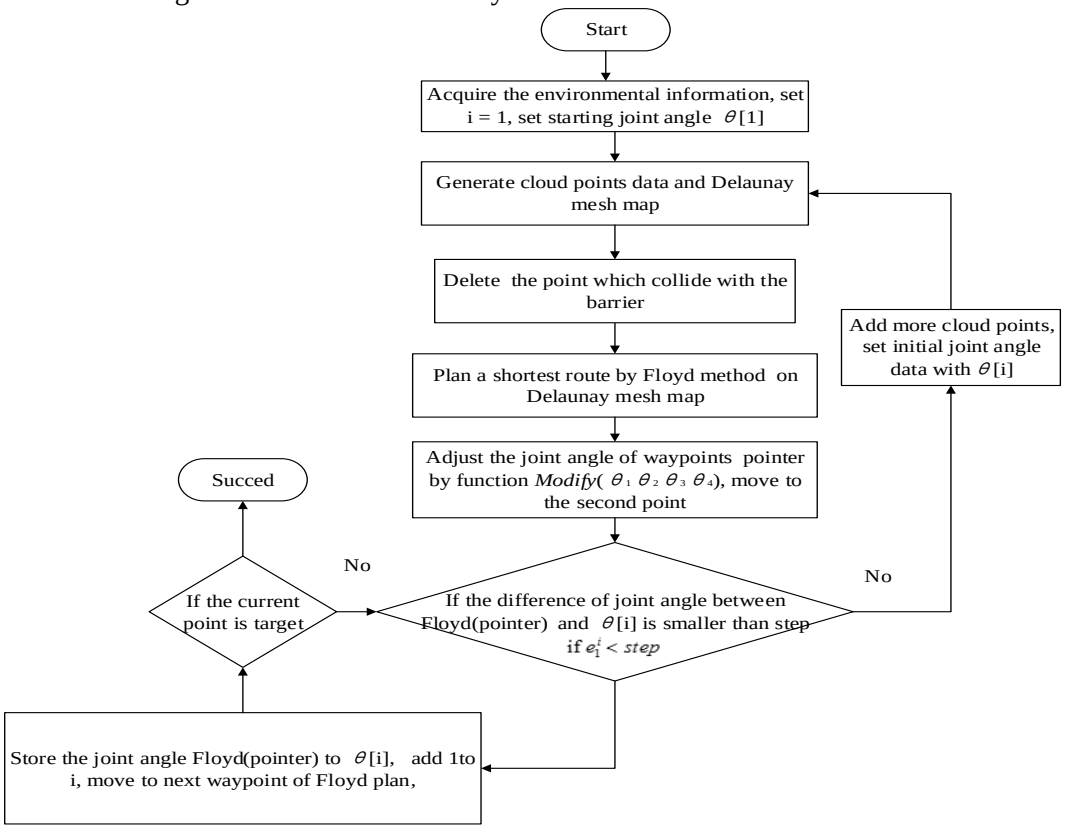

Figure 10. Flow chart of path refining algorithm

For the same reasons in 5.2, the joint angle trajectory is not as smooth and continuous as expected, concerning the optimal path obtained by Floyd algorithm on the Delaunay map. To overcome the defect, the local path points were optimized and the joint angle was adjusted by $modify(\theta_1,\theta_2,\theta_3,\theta_4)$ function. Meanwhile, the path points were traced back to re-plan the path lines if the joint angle of these points fluctuated significantly.

Based on the Monte-Carlo point cloud data, the Floyd algorithm was employed to plan the path sequence of the current point to the target point in the Delaunay grid map until the accuracy reached $e_1^i < step$ and the target point was arrived at (Figure 10).

The path refining algorithm enhances the convergence quality and speed by changing the intermediate path point and correcting the path. In view of the random generation of Monte-Carlo

point cloud data, the global optimal path can be produced by making full use of the known environmental information and random obstacle information.

6 SIMULATION AND EXPERIMENT

Overview

To verify the robot manipulator path planning algorithm based on Delaunay map, a 5-DOF space manipulator was taken as the target of simulation. The D-H coordinates and parameters of the manipulator are shown in Figure 11 and Table 2, respectively. The obstacle was simulated by a cube model (length: 0.1m; width: 0.13m; height 0.1m) (Figure 12).

Table 2. Defined Denavit-Hartenberg parameters

Link	L_1	L_2	L_3	L_4
Length(cm)	10.3	10.3	12.7	11.5

The kinematics analysis was performed in a simulation system programmed in Matlab. The system ran on a computer (2.5 GHz CPU; 4GB RAM). The Cartesian coordinates of the initial path point were (5, 10, 20) with a joint angle of (1.11, 1.23, -0.13, -4.07), and the target path point coordinates were (0, 20, 0) with a joint angle of (1.57, 1.02, -1.82, -1.82). Hence, the joint angle difference$\Delta\theta$ between the initial point and the target joint was (0.46, -0.21, -1.68, 2.64). The differences $\Delta\theta_4$and $\Delta\theta_3$ were the larger ones between the two points. After obtaining the adjacent joint angle difference e_1^i, the author set$\Delta\theta_3$ with $abs(\theta_3^i - \theta_3^{i-1})/5$, set $\Delta\theta_4$ with $abs(\theta_4^i - \theta_4^{i-1})/10$ and $step = 0.8$. Table 3 shows the joint angle of the path point generated by the first iteration of the path optimization.

Table 3. The path point of the joint angle after first path planning

Joint angle of waypoint	θ_1	θ_2	θ_3	θ_4
1	1.11	1.23	-0.13	-4.07
2	1.17	0.96	-0.03	-4.47
3	1.67	0.87	-0.98	-4.23
4	1.69	1.21	-1.52	-3.55
5	1.58	1.56	-1.87	-2.84
6	1.73	1.54	-1.89	-2.38
7	1.57	1.02	-1.82	-1.82

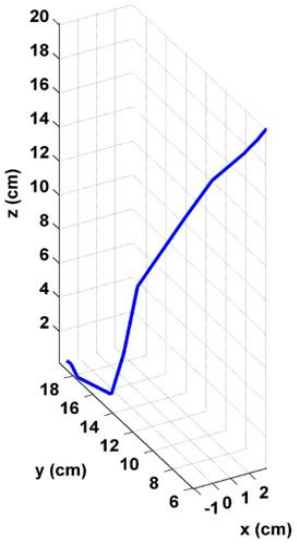

Figure 11. End of the actuator track diagram

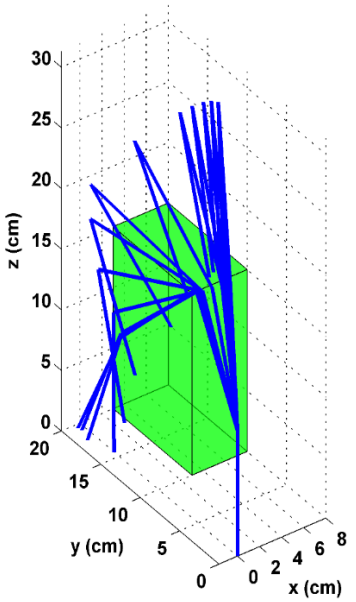

Figure 12. Optimized Robot manipulator pose series during a path to avoid obstacle

For batter accuracy, at the completion of the first path planning, the step size was reduced to *step*/2 from the point int((*i*+1)/2) of the first path, and the first path was divided into two segments: (1, int((*i*+1)/2) and (int((*i*+1)/2, *i*). The initially planned path and the optimized path are given in Figures 11 and 12, respectively.

6.1 Experimental validation

The target manipulator is an industrial serial manipulator, whose features have been extensively explored in the reference (Gómez-Bravo et al., 2012). Thus, the experimental validation mainly focuses on the application of our algorithm on the

arm end movement along the planned path. The obstacle was a 0.1m×0.1m×0.13m cuboid. The end points of the manipulator were set as point (5, 10, 20) and point (0, 20, 0). Under the given conditions, our algorithm was implemented on the serial manipulator. The angle point sequence of the path points is recorded in Figure 13.

During the test, the manipulator moved swiftly and smoothly from the starting point to the target point without hitting the obstacle. The results prove that speed and effect of our obstacle avoidance path planning algorithm. Suffice it to say that the algorithm has great application potential in the robot manipulator control.

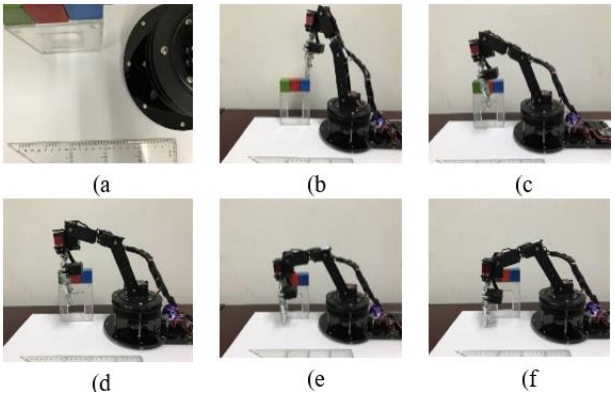

Figure 13. Snapshots of manipulator operation during a path to avoid a cylindrical obstacle

7 CONCLUSION

In this paper, the global planning and the local planning were combined into an organic whole. First, the Monte-Carlo method was adopted to construct the Delaunay tetrahedral grid map. Then, the Floyd algorithm was employed to find the path of the collision-free global shortest path of the Cartesian space in the Delaunay map. In other words, an obstacle avoidance path planning algorithm was developed for robot manipulator.

The innovation points of this research are as follows. First, the joint angle of the Cartesian space point was directly acquired and the optimization range was narrowed down by using the point cloud data obtained by Monte-Carlo method as the path points in path planning. Second, the Delaunay tetrahedral mesh map quickly generated the feasible paths of the manipulator in the Cartesian space. Third, the map generated by the Monte-Carlo method made full use of the known environmental information, produced the global optimal path, and tackled the obstacle information in a timely manner. Fourth, the repeated use of our algorithm optimized the path points with severely fluctuating joint angles, and led to the satisfactory trajectory of the robot manipulator.

However, this research has not predicted the next state of the manipulator in an environment with moving obstacles. In light of our algorithm, the long-term moving-obstacle avoidance path planning will be discussed in further research.

8 ACKNOWLEDGEMENTS

This work is supported primarily by the National Key R&D Program of China with no.2017YFC0806608, Funding of Innovation Program for Graduate Education of Army Logistics University of PLA.

9 REFERENCES

► Chen, W. D., Zhu, Q. G. (2011). Mobile Robot Path Planning Based on Fuzzy Algorithms, Acta Electronica Sinica, 39(4), 971-974.

► Xu, X. Q., Zhu, Q. B. (2012). Multi Artificial Fish-Swarm Algorithm and a Rule Library Based Dynamic Collision Avoidance Algorithm for Robot Path Planning in a Dynamic Environment, Acta Electronica Sinica, 40(8), 1694-1700.

►Dong, H., Nie, H., Chen, J., Chen, M. (2017). Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection, Robotics and Computer-Integrated Manufacturing, 49, 98-104.

►Dong, H.,W. (2005). A Triangular Mesh Reconstruction Algorithm for Points Cloud, Computer Engineering, 31(15), 30-32.

►Fang, C., Zhao, J. (2010). New Dynamic Obstacle Avoidance Algorithm with Hybrid Index Based on Gradient Projection Method, Journal of Mechanical Engineering, 46(19), 30-37.

►Gómez-Bravo, F., Carbone, G., Fortes, J. C. (2012). Collision free trajectory planning for hybrid manipulators, Mechatronics, 22(6), 836-851.

►Jia, Q. X. (2010). Path Planning for Space Manipulator to Avoid Obstacle Based on A~* Algorithm, Journal of Mechanical Engineering, 46(13), 109-115.

►Jiang, L., Li, G., Zhou, Y., Sun, K., Liu, H. (2013). Obstacle avoidance control for 7-DOF redundant manipulators, Optics and Precision Engineering, 21(7), 1795-1802.

►Keshtkar, M. M. (2017). Energy, exergy analysis and optimization by a genetic algorithm of a system based on a solar absorption chiller with a cylindrical PCM and nano-fluid, International Journal of Heat and Technology, 35(2), 416-420.

►Lee, S. D., Song, J. B. (2016). Sensorless collision detection based on friction model for a robot manipulator, International Journal of Precision Engineering and Manufacturing, 17(1), 11-17.
►Liang, C. H., Zeng, S., Li, Z. X., Yang, D. G., Sherif, S. A. (2016). Optimal design of plate-fin heat sink under natural convection using a particle swarm optimization algorithm, International Journal of Heat and Technology, 34(2), 275-280.
►Liu, L. F., Wang, Y. J. (1996). A Three-Dimensional Algorithm on Detecting Collision Between Robot and Its Environment, ROBOT, 18(1), 50-54.
►Qi, R. L., Zhang, W. J., Wang, T .J. (2014). An Obstacle Avoidance Trajectory Planning Scheme for Space Manipulators Based on Genetic Algorithm, ROBOT, 36(3), 263-270.
►Qian, K., Jia, K., Song, X. (2015). Robot manipulator avoidance planning based on low-dimensional mapping and Q-learning, Journal of Huazhong University of Science & Technology, 43, 468-472.
►Rubio, F., Llopis-Albert, C., Valero, F., Suñer, J. L. (2016). Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory, Robotics and Autonomous Systems, 86, 106-112.
►Shi, E., Chen, M., Li, J., Huang, Y. (2014). Research on Method of Global Path-planning for Mobile Robot Based on Ant-colony Algorithm, Transactions of The Chinese Society of Agricultural Machinery, 45(6), 53-57.
►Tian, H. B., Ma, H. W., Juan, W. (2013). Workspace and Structural Parameters Analysis for Manipulator of Serial Robot, Transactions of The Chinese Society of Agricultural Machinery, 44(4), 196-201.
►Wang, S. K., Zhu, L., Wang, J. Z. (2015). Path paln of 6-DOF robot manipulators in obstacle environment based on navigation potential function, Transactions of Beijing Institute of Technology, 35(2), 186-191.
►Wang, T. C., Xie, Y. Z. (2016). BP-GA data fusion algorithm studies oriented to smart home, Mathematical Modelling of Engineering Problems, 3(3), 135-140.
►Xiao, G. R., Gan, W. J., Chen, W. T. (2010). Exchange of Delaunay Tin Betweenmatlab and Gmt in Crustal Strain Calculation, Journal of Geodesy and Geodynam Ics, 30(3), 122-126.
►Zhang, J. H., Hu, P., Zhang, X., Liu, J., Liu, X. (2017). Closed Loop Control Algorithm for Obstacle Avoidance Based on the Transformation of Master and Slave Tasks, Journal of Mechanical Engineering, 53(1), 21-27.
►Zhu, D. Q,, Yan, M. Z. (2010). Survey on technology of mobile robot path planning, Control and Decision, 25(7), 961-967.