ROS-based robot offline planning simulation system

Liu Zhiheng*, Chen Jian, Mei Zhen and Li Chao
Wuhu HIT Robot Technology Research Institute Co., Ltd.
*Corresponding author’s e-mail: liuzhiheng@hitrobot.com.cn

Abstract. In order to realize the rapid and automatic generation of robot motion trajectory, a robot offline planning simulation system was developed based on ROS platform. This paper analyzes the model STL file, reconstructs the topology data structure, generates the model minimum bounding box, and obtains the model surface path through the slicing algorithm to generate the processing path. The system loads the Moveit motion planner, applies different motion planning methods, generates a collision-free, continuous motion trajectory, and visualizes the offline planning process. The simulation verification system can realize the functions of rapid path generation on the workpiece surface and obstacle avoidance motion planning of the robot arm. It can realize offline programming of robots and improve the efficiency of robot teaching.

1. Introduction
Robots are used more and more in modern industrial manufacturing. The traditional online teaching method is reliable, efficient, low-cost and easy to implement for low precision and simple operation objects. However, with the increase in product complexity and manufacturing accuracy requirements, there are problems such as cumbersome programming, low efficiency, and low precision. The method of teaching the robot to generate the robot control program can not meet the production needs, so it is urgent to develop the robot offline programming technology[1].

Since the application of off-line planning technology improves the efficiency of robot programming, the manufacturing flexibility, production efficiency and product quality of the system are improved[2-3]. Foreign offline programming technology has developed earlier and has mature commercial software, but most of them are expensive and have certain limitations. e.g. ABB Robot's RobotStudio can provide a complete offline planning system[4], but it can only be applied to its own brand of robots, and it is not universal; Robotmaster is mainly used for 3D milling, and it is not suitable for other processing fields. DELMIA is mainly used for processing in the automotive industry, and its expert software is difficult to operate. Robcad is mainly used for production line simulation, and its offline function is weak. Domestic offline programming technology started relatively late, mostly research in university laboratories and research institutes. e.g. Guo Jichang and others used the UG/Open function library to develop the offline programming system of KUKA KR30L16 robot, and realized key technologies such as motion simulation and detection[5]. Song Pengfei based on SolidworksAPI secondary development function, designed offline programming simulation system to achieve automatic generation of
FANUC robot operating program in Solidworks environment[6]. Li Xiang proposed an interactive 3D offline programming and simulation system based on OpenGL, which can detect collisions or singular points during robot operation[7]. Xu Xiang and other Qt for the software development framework combined with the functions of OpenGL and VRML, established a modular robot offline programming system[8]. With the Unity3D software as the system development platform, Tao Tao designed a cross-platform offline programming system to realize the motion simulation of the mobile client-to-system[9].

In summary, the offline programming system based on third-party software development has realized basic functions, but does not provide robot obstacle avoidance planning functions for complex working conditions. The ROS system is a robot software platform that provides complete obstacle avoidance planning. At present, the offline programming system based on ROS platform has no process of model processing, so it is not a complete offline programming system[10-11].

This paper builds a complete offline planning system based on the open source ROS platform combined with the processing technology to explore the offline planning method. Through the secondary development of the scalable view tool Rviz, the machining path planning of the workpiece surface is carried out. The integrated motion planning library Moveit plans the motion trajectory with obstacle avoidance function, and builds a visual simulation platform to realize the motion simulation function.

2. Model surface path planning
The STL file is imported from the outside through the Rviz visualization function. Since the shape of the geometric model is composed of a triangular mesh, the surface path of the model can be quickly planned by the slice processing method. The planning process is:

- Reading, topology reconstruction and storage of triangular mesh information of the STL model.
- Obtain the minimum bounding box OBB of the model component.
- Generating a set of plane clusters parallel to the calibration surface with any section of the minimum bounding box as the initial plane.
- Obtaining the intersection of each plane and the triangular grid of the spraying component in turn, and connecting the intersection points in sequence to form a strip path stroke.
- The path point at the end of the previous stroke is connected with the path point of the start point of the latter stroke, and finally the surface path of the model is formed.

2.1. Topological reconstruction of data structures
The topology between the meshes is reconstructed by reading the triangular mesh information[12]. The topology is built according to the following criteria: a triangular mesh has at most three co-edge triangles and at least one. All co-edge triangles are obtained by traversing the vertex information of the triangular mesh to establish a topological data structure of faces, edges, and vertices. It is possible to quickly retrieve the three sides and three vertex numbers of the face, the two vertices of the edge and the face number, the coordinates of the vertex and the face number. In order to save storage time and space, the coordinate information of the points only needs to be recorded once and indexed by number. The specific data structure is as follows:

```c
struct Facet  // Data structure
{
    int f_inx_; //Face number
    int vs_inx_[3]; // Number of three vertices
    int es_inx_[3]; // Number of three sides
    Ogre::Vector3 normal_; // Normal vector
    double area_; // Area
}
```
Struct Edge // Edge data structure
{
int e_inx_; // Side number
int vs_inx_[2]; // Number of two vertices
int fs_inx_[2]; // Number of two faces
double length; // Length
}

Struct Vertex // Vertex data structure
{
int v_inx_; // Vertex number
Ogre::Vector3 v_data_; // Coordinate information
std::vector<int> face_inx; // Face number
Ogre::Vector3 v_nor; // Normal vector
}

Figure 1. Adjacent faces of point

Since the STL model only provides the normal vector of the face, the normal vector of the intersection can be obtained indirectly from the normal vector of the face. Here we use the method of calculating the normal vector based on the area weight [13]. Analyze the geometric properties of the adjacent faces of the vertex p in figure 1, and retrieve its normal vector and area, respectively, and the normal vector of point p is:

\[ n = \sum_{j=1}^{m} s_j n_j \left( \sum_{j=1}^{m} s_j \right)^{-1} \]  \hspace{1cm} (1)

2.2. Generation of truncated plane clusters

In this paper, a plane of the model minimum bounding box (OBB) is used as the initial calibration plane of the truncated plane cluster, and a truncated plane cluster parallel to the calibration plane is generated according to the set stroke spacing parameter, as shown in figure 2. The OBB bounding box is obtained according to the reconstructed topological data structure, and principal component analysis (PCA) is performed on all the vertex information to obtain three feature vectors and serve as three main axes of the OBB.

The specific steps are: First, index the area of the adjacent triangle patch \( \Delta S_1, \Delta S_2, \cdots, \Delta S_k \) of each vertex p, calculate its weight \( \omega_p \), the mean of the weights \( \bar{p} \):

\[ \omega_p = \text{sum}(\Delta S_1, \Delta S_2, \cdots, \Delta S_k) / k \]  \hspace{1cm} (2)

\[ \bar{p} = \frac{1}{n} \sum_{j=1}^{n} \omega_j p_j \]  \hspace{1cm} (3)

where \( k \) is the number of patches, \( n \) is the number of vertices. Therefore, construct the covariance matrix as:

\[ C = \frac{1}{n} \sum_{j=1}^{n} (\omega_j p_j - \bar{p})(\omega_j p_j - \bar{p})^T \]  \hspace{1cm} (4)
Then calculate its eigenvector $v_1, v_2, v_3$, and set it to the direction of the major axis of the bounding box. Finally, all the vertices are projected on three main axes to obtain the bounding box center point and three axis lengths.

![Figure 2. Bounding box and Section plane cluster](image)

### 2.3. Generation of truncated plane clusters

Here, a slice algorithm is used to obtain a path point. Traverse the edges in the model to determine whether it intersects the section plane. If it intersects, it terminates the loop and calculates the intersection point, which is the path point of the section plane. Index the two adjacent faces of the intersecting edge, find the other edges that intersect the intersecting plane and calculate the intersection. By analogy, the intersections are recursively reciprocated to the two sides until there is no intersecting edge or the intersecting edges coincide (circular path). Finally, a path point intersecting the section can be obtained.

![Figure 3. Point normal vector and Point pose](image)

Analysis of the geometrical characteristics of the intersecting plane intersection point shows that the $p_0$ point pose can be obtained from its normal vector and machining direction, as shown in figure 3. The normal vector of point $p_0$ is determined by the characteristics of the surface. By indexing the data structures of the vertices $p_1$ and $p_2$ of the intersecting edges, the normal vectors $e_1$ and $e_2$ of the two points can be obtained respectively. Here, the position of $p_0$ point between point $p_1$ and point $p_2$ is used as the weight, and the normal vector $e_0$ of point $p_0$ is calculated:

\[
edder_0 = k_1e_1 + k_2e_2
g_k = l_1 / (l_1 + l_2)\]  
\[
k_2 = l_2 / (l_1 + l_2)\]  

Where $k_1$ is the weight of point $p_1$, $k_2$ is the weight of point $p_2$, $l_1$ is the distance between point $p_1$ and point $p_0$, and $l_2$ is the distance between point $p_2$ and point $p_0$.

The machining direction on the same section plane is in the section plane, so it is perpendicular to the normal vector $es$ of the section plane. And the machining direction is along the tangential direction of the curved surface, so it is perpendicular to the surface normal vector and is approximated by the normal vector $e0$ of the point $p_0$. Therefore, the machining direction vector $e_k$ is:

\[
edder_k = e_0 \times e_i\]  

According to the general situation established by the robot tool coordinate system, the normal vector $e_0$ is taken as the $z$-vector $RZ$ of the pose, the direction vector $e_k$ is taken as the $x$-vector $RX$ of the pose, and the $y$-vector $RY$ of the pose is obtained by $RZ \times RX$. Then get the spatial pose of $p_0$.

At this point, a path point position and posture of the section plane can be obtained. Generate path points for all truncated planes, and connect the end points of each leg to form an offline path for the
entire model. The offline path is a series of discrete pose points, although it reflects the actual feature information of the workpiece surface, but cannot be directly sent to the arm movement. Since the offline path requires smooth processing, robot arm collision detection, trajectory planning, etc. Moveit is required to optimize the path.

3. Motion planning of the manipulator

The Moveit motion planning library includes forward and inverse kinematics calculations, obstacle introduction, collision detection, and motion planning. Based on the Rviz plugin mechanism, a robotic arm scene can be generated for visual simulation. Load the motion planner specifically through the following modules:

- Robot Model, the configuration assistant loads the arm URDF model and configures the joint parameters, the robot arm configuration, the end operator, etc.
- Planning Scene, by adding a restraining object in the movement space of the robot arm, the movement of the mechanical arm avoids the obstacle and ensures the normal operation of the mechanical arm.
- Motion Plan, instantiate the object Robot Model and Planning Scene, load the motion planner implementation to call the motion planning function.

Complete the loading of the motion planner to plan the motion of the path point. Specific exercise planning process:

1. Set the initial state of the manipulator motion planning according to the robot processing technology requirements. Here, the setStartState function is used to reset the initial state of the arm.
2. Since the planned offline path is the position that the mechanical arm needs to reach in Cartesian space, the motion path can be obtained by planning the mechanical arm motion using the Cartesian coordinate path (computeCartesianPath function) provided in Moveit. However, this function only has collision detection and cannot perform obstacle avoidance planning. Output failure results for a path point with collisions.
3. The success rate fraction of the plan is returned by the computeCartesianPath function to determine whether the current plan is successful. Here, the fraction threshold is set to 0.1 (10%). When the planned success rate is less than 10%, it is regarded as a planning failure and the planning process is interrupted. Save the planned path points and generated motion points, and set the current path point to the initial state of the next stage plan.
4. The current initial state point does not satisfy the condition of the Cartesian coordinate path planning, and the function with obstacle avoidance planning is used. Set the position of the planning point by the setPoseTarget function, which can adjust the attitude of the arm in a wide range and be used for the collision point in the transition path. This function is based on sampling-based obstacle avoidance motion planning, which has a large amount of calculation and a long planning time, and is not suitable for planning of large-scale points. Therefore, after the transition point adjusts the posture, continue to use the computeCartesianPath function for motion planning.
5. Repeat steps 2-4 until the robot motion planning for all path points is completed.

After Moveit's robotic arm motion planning, a series of motion points that the robot can perform are obtained. The post-processing module of the robot controller generates a command recognized by the robot to control the movement of the robot arm along the planned trajectory.

4. Simulation test and result analysis

The offline path planning method proposed in this paper is mainly for the path planning of the workpiece surface, and is suitable for processing fields such as spraying and cleaning. The effectiveness, completeness and accuracy of offline path planning are verified by planning different workpiece surfaces. Here, the surface path planning is performed on the plane, the curved surface and the annular surface respectively, and the line spacing is set to 50 mm, and the Ubuntu 16.04 system and the ROS Kinetic version are adopted. Obtain the path point pose, as shown in figure 4, where
yellow represents the machining path, red represents the z-direction, blue represents the x-direction, and green represents the y-direction.

Analyze the simulation results:

- The path point position is obtained by the intersection of the section plane and the patch, and it can be visually seen that the plane path points are regularly distributed according to the shape of the triangle mesh.
- The z-direction of the path point is always perpendicular to the patch, and the x-direction is along the direction of motion. It can be seen that the attitude of the curved path point can change as the shape of the patch changes.
- The path of the toroidal plane is planned along the surface of the patch, and there is no overlap or missing.

The above path planning for different surfaces verifies that it has offline planning performance. However, for a workpiece, it is sometimes necessary to process a plurality of faces, so that a combined path planning of a plurality of patches is required. According to the model features, the 3D modeling software divides into different planes or surfaces for surface path planning. Here, the cuboid is taken as an example for path planning, and the path point pose is obtained as shown in figure 5.

The simulation platform of the offline planning system was built, and the six-degree-of-freedom industrial robot was used to simulate the surface contour of the die casting. The offline planning simulation system interface is written based on QT, as shown in figure 6. It can realize the modification of the imported model pose, select different patches for path planning and motion planning, import and delete obstacles, save path points and motion points. Figure 7 shows the motion simulation of the robot along the contour of the die casting, which is convenient for the user to observe the process of the robot. The robot runs from the initial state to the infeed point, moves along the blue contour to the retraction point, and returns to the initial state. After confirming that there is no problem during the exercise, save the robot movement point. The robot is processed and sent to the robot controller to complete the offline planning and simulation process of the robot.
5. Conclusion
Based on the open source ROS platform, this paper develops an off-line simulation system for surface path planning and robot arm motion planning. The model layered slicing algorithm is used to solve the problem that the model can't handle in the ROS system. Reconstruct the mesh information topology data structure, generate the model minimum bounding box, quickly slice the path point position and posture, and complete the path planning of the workpiece surface. Based on the Moveit motion planning open source library, the motion planning process is executed to simulate robot motion and generate continuous, collision-free motion trajectories. Through simulation experiments on different workpieces, it is verified that the simulation system has the offline planning function of the robot, which can improve the efficiency and accuracy of robot programming. The system has great application potential for offline planning in the fields of industrial robot cleaning and spraying.

Acknowledgement
This work is supported by National Key Research and Development Program of China under Grant 2017YFB1303900.
References

[1] Lin, Y.Z., Liu, Q.Q., Xu J., Liao, J.F. (2015) Research status and development trend of industrial robot off-line programming system. Mechatronics, 21(7): 8-10.

[2] Polden, J., Pan, Z., Larkin, N., Duin, S.V., Norrish, J. Offline programming for a complex welding system using DELMIA automation. (2011) Lecture Notes in Electrical Engineering, 88: 341-349.

[3] Wylie, N., Wylie, S.R., Cullen, J.D., Al-Jader, M. (2010) NDE system for the quality control of spot welding in the automotive industry. Sensors Applications Symposium. IEEE.

[4] Holubek, R., Delgado Sobrino, D.R., Košťál Peter, Ružarovský, R. Offline programming of an ABB robot using imported CAD models in the RobotStudio software environment. (2014) Applied Mechanics and Materials, 693: 62-67.

[5] Guo, J.C., Zhu, Z.M., Yan, G.R., Ma G.R. Development of off-line programming system for arc welding robot based on UG. (2016) Electric Welding Machine, 46(10).

[6] Song, P.F., He, R.L., Miao, J.Z., Shan, D.R. Offline programming system based on Solidworks for industrial robot. (2013) Manufacturing automation, 35(9): 1-4.

[7] Li, X., Qian, H., Fu, Z., Zhao, Y.Z. The off-line programming and simulation system design of robot based on QT and OpenGL. (2013) Mechatronics, 19(4): 56-59.

[8] Xu, X., Gan, Y.H., Dai, X.Z., Duan, J.J. Extensible robot off-line programming system based on QT. (2015) J. Huazhong Univ. of Sci.&Tech. (Nature Science Edition). (s1): 513-516.

[9] Yan, T., Zhang, P., He, C.J., Zhang, Z.P. Research and development of the cross-platform industrial robot off-line programming system based on Unity3D. (2016) Modular Machine Tool & Automatic Manufacturing Technique, (10): 99-104.

[10] Lu, Z., Han, Y.Y., Ji, L., Guo, W. Development of an off-line programming system for UR robots based on ROS. (2017) Computer Simulation, (09): 322-326+331.

[11] Wen, K.C. ROS-based motion simulation and trajectory planning of palletizing robot. 2016.

[12] Yang, G., Liu, W.J., Wang, W., Tian, F.J. Research on the rapid slicing algorithm based on STL topology construction. (2009) Modern Manufacturing Engineering. (10): 32-35.

[13] Sun, Y.W., Liu, W.J., Wang, Y.C. Research on the algorithm of NC tool path calculation for triangular surface machining. (2002) Chinese Journal Of Mechanical Engineering, 38(10): 50-53.