Path Planning for Industrial Robots in Free-Form Surface Polishing

Zhaosheng Li[1,2] and Wei Wang[1]
[1] Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
e-mail: lizhaosheng2017@ia.ac.cn, wei.wang@ia.ac.cn

Abstract—**The surface roughness of mold affects the product quality and the surface quality of the blade and aircraft has a direct impact on stability and reliability of related equipment, so the polishing plays an important role in manufacturing industry. Conventional industrial robots provide teaching pendant to plan the path, while the teaching of free-form surface is difficult and complicated. This paper utilizes the off-line programming to plan the path. Before the path planning, we read STL format files of the free-form surface objects and build topology relations. Comparing the direction-parallel and contour-parallel path patterns, we find that the direction-parallel pattern needs less degree of freedom and is much easier to avoid physical interventions and mechanic singularity, so we adopt the direction-parallel method to plan the path at last. We solve the intersection between the cutting plane and free-form surface based on the topology relations. We develop a simulation platform using the C++ and OpenGL and conduct the experiment. The experiment proves that the path generated by direction-parallel method based on topology relations is accurate.**

Keywords-path planning; free-form surface; robot; polishing

I. INTRODUCTION

During the process of mold manufacturing, the surface roughness of mold affects the product quality, and thus the polishing plays an important role for improving the quality of product. Molds are general polished manually as the finishing process after milling. This process takes the special technical skills and much time which leads to higher cost of the molds [1]. High-end manufacturing filed also requires the polishing such as automobile, turbine blades and aircraft manufacturing. The surface quality of these workpiece has a direct impact on the service life, stability and reliability of related equipment. The skilled workers cost much time to polish the workpiece [2]. The industrial robot is widely used in many enterprises such as machining, assembly and welding. Developing the industrial robot used to polish molds, workpiece and blade is absolutely essential. The first to do is planning the path of the polishing tool movement.

Some researches have been done before. F. Nagata applies the robot to the polishing process of PET(Poly Ethylene Terephthalate) bottle molds and get the paths from the cutter location(CL) data generated from 3D CAD/CAM systems [3]. K. Zhang utilizes the six-axis machining tool to polish the blade including the free-form surface and propose path planning method for machining on the blades [4]. X. Yang designs an automatic belt grinding system to grind the blade and plan the path of grinding [5].

Conventional industrial robots provide only a teaching pendant as user interface devices and the teaching of free-form surface is extremely difficult and complicated, thus the off-line programming using the CAD models is convenient and efficient. This paper utilizes the stereolithography(STL) files which are widely used in many industries such as computer-aided manufacturing and 3D printing. The STL files include plenty of unstructured triangular facets, and it is difficult to plan the path on the independent triangles. Therefore we use the half-edge method to build topology relations. The active edge method is used to extract to the polishing surface. Through comparing the direction-parallel and contour-parallel path patterns, we find that the direction-parallel path pattern needs less degree of freedom and is much easier to avoid physical interventions and mechanic singularity, and we adopt the direction-parallel method to plan the path at last. We present the specific algorithm of solving the intersection between the cutting plane and free-form surface based on the topology relations. We test our method by planning the path on two representative surfaces. The experiment proves that the path generated by direction-parallel method based on topology relations is accurate.

In the remainder of the paper, we firstly read the STL files and build topology relations. Then we utilize active edge method to extract the polishing surface. In addition, we adopt the direction-parallel method to plan the path after comparing the two path patterns and introduce the algorithm of cutting plane in detail. Afterwards, experimental results are demonstrated and analyzed. Finally, we draw the conclusions and describe the future work.

II. PRELIMINARY WORK

There are two kinds of CAD models typically used in the path planning: mesh and parametric. The mesh model has recently become the focus of considerable interest, because its geometric computation is simpler than the parametric model. There are several formats of mesh models, such as STL, OFF and NASTRAN. We focus our experiment study on the STL files. Then we will read STL files, build the topology relations and extract the polishing surface.

A. Read STL Files

STL is a file format created by 3D systems. The file format is supported by many other software packages and used for interchanging data between CAD/CAM systems. An STL file describes an unstructured triangulated surface by the unit normal and vertices(ordered by the right-hand rule) of the triangles using a three-dimensional cartesian coordinate system as shown in Fig. 1. The STL files include ASCII

978-1-7281-3326-3/19/$31.00 ©2019 IEEE

and binary formats. According to the characteristics of the ASCII and binary STL files, we read STL files by using C++ program.

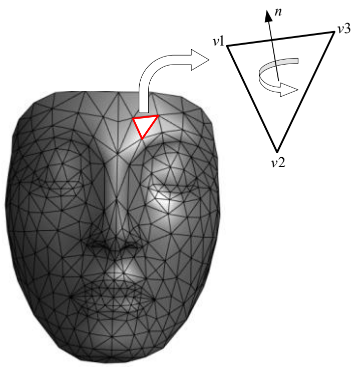


Figure 1. The mesh surface of the face acquired by reading the STL file.

The data redundancy of the STL files is very large as almost every vertex is recorded repeatedly 6 times. At the same time, it is difficult to analyze the free-form surface feature and plan the path on hundreds of unstructured triangular facets. It is essential to build topology relations of STL format files which can remove the redundancy and improve the efficiency of path planning.

B. Build Topology Relations

Half-edge structure was taken to build topology relations of STL format files in this paper. Half-Edge structure is presented in Fig. 2.

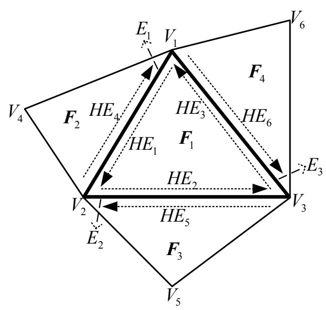


Figure 2. Half-Edge structure.

An edge in the mesh surface is divided into two directed edges. Each triangle consists of three half edges and three vertices, and all of them meet the right-hand rule. For example, the edge E_2 is divided into two directed half edges HE_2 and HE_5. The triangle face F_1 has three half edges HE_1, HE_2 and HE_3. Each edge has two adjacent triangles such as E_2 is shared by F_1 and F_3. Each Half-Edge has an adjacent half edge such as the HE_5 is adjacent half edge of HE_2. Building topology relations is to create the adjacent information among triangles, half edges and vertices. The topology relations are as follows:

- When a triangle is known, the three vertices, the normal vector, the three half edges and the three adjacent triangles of the triangle are also known.
- When a half edge is known, the two vertices, the triangle, the adjacent half edge, the adjacent triangle of the half edge are also known.
- When a vertex is known, the adjacent vertices, the half edges and the triangles of the vertex are also known.

1) The data structure of topology relations

During the process of building the topology relations, the data structures of the vertices, half edges and triangles need to be built. The data structure is divided into four classes: Vertex, HalfEdge, Facet, STLSolid. All the implementation of the algorithm is based on the four classes.

2) The algorithm of topological reconstruction

The main work of the topological reconstruction process is to merge duplicate vertices which asks for lots of search and insertion, so the speed of searching vertices affects the speed of the topological reconstruction. In the process of building topology, the set container of C++ Standard Template Library is used. Sets are containers that store unique elements following a specific order. It is very fast to access and search the data for the set container which is implemented as binary search trees. The set container is very suitable for topology construction because of its high efficiency. The value of an element is also the key used to identify it. It is essential to overload the comparison operator. The comparisons of the vertices and half edges themselves are as follows:

$$\begin{array}{l} \left[p_1.x < p_2.x \right] \quad or \\ \left[\left(p_1.x = p_2.x \right) and \left(p_1.y < p_2.y \right) \right] or \\ \left[\left(p_1.x = p_2.x \right) and \left(p_1.y = p_2.y \right) and \left(p_1.z < p_2.z \right) \right] \end{array} \tag{1}$$

and

$$\begin{array}{l} \left[hf_1.starVex < hf_2.startVex \right] \quad or \\ \left[\begin{array}{l} \left(hf_1.starVex = hf_2.startVex \right) \; and \\ \left(hf_1.endVex < hf_2.endVex \right) \end{array} \right] \end{array} \tag{2}$$

The algorithm of topological reconstruction is as follows.

Step1: Declare vertex set container *vertexSet* and half edge set container *halfEdgeSet*. Read *num* triangles.

Step2: For the *ith* triangle, judge if $i \le num$. If not, end. If so, continue the next step.

Step3: Build three vertices. For *ith* triangle, insert three vertices in the *vertexSet*. If success, add the vertex to the vertices array. If not, the vertex exists. At last, complete other relations, such as the index of vertex and the relation of vertex and facet.

Step4: Build three half edges. Insert three half edges into the *halfEdgeSet* and vertices array. Search adjacent half edges of the three half edges. If search success, add the adjacent relations. At last, complete other relations, such as the index of the facet, the relation of the half edge and vertices, the relation of the half edge and the facet and the adjacent facets.

Step5: Insert the facet into the facets array. Let $i = i + 1$. Return the step2.

C. *Extract Polishing Surface*

The process of extracting polishing surface mainly include two steps, determine if the triangle is needed and extract the triangles and boundary half-edges. The two steps are as follows.

1) Determine if the triangle is needed

To determine if the triangle is needed, we improve edge-based method for defining feature boundaries which rely on the dihedral angle between two triangles. The normal vector of a new triangle is n_1, the given normal vector is n_2 and the threshold is $threshold$. If

$$\frac{n_1 \bullet n_2}{|n_1||n_2|} \geq threshold \tag{3}$$

the new triangle is needed.

2) Algorithm of extraction polishing surface

We use the active edge method to extract the polishing surface. The brief principle of the method is as follows:

Let triangle 1 be the first triangle and add the three half edges to the active edge list in Fig. 3. The list includes *AB*, *BG* and *GA*. Then traverse the active edge list and search the adjacent triangle of the half edges. The list deletes the *AB*, and includes the half edges *AD*, *DB*, *BG* and *GA*. Repeat above the process.

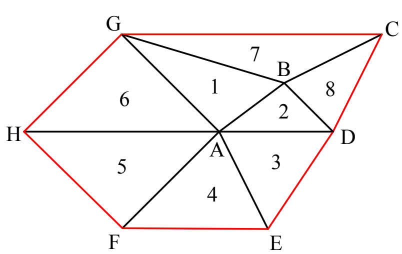


Figure 3. A brief instance of extract the polishing surface.

When the adjacent triangle dissatisfies the threshold condition, delete the half edge in the active edge list and add the half edge to the boundary half edge list. At last, the active edge list is empty and the boundary half edge list includes *ED*, *DC*, *CG*, *GH*, *HF* and *FE*. The polishing surface is the triangle 1, 2, 3, 4, 5, 6, 7 and 8.

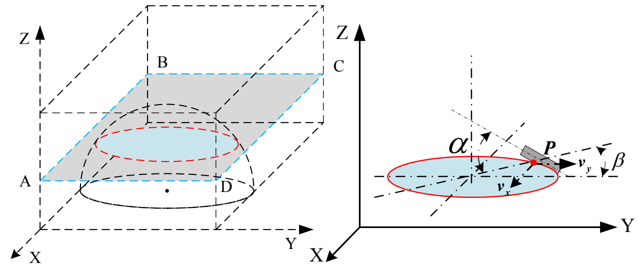
III. PATH PLANNING

A. *Path Pattern*

There are two main path patterns, contour-parallel path pattern and direction-parallel path pattern [6].

1) Contour-parallel pattern

A series of cutting planes which are vertical to Z-axis cut the surface of the work-piece. The cutting plane intersects with the work-piece generating the color one and its boundary red line is the path of polishing tool in Fig. 4(a).



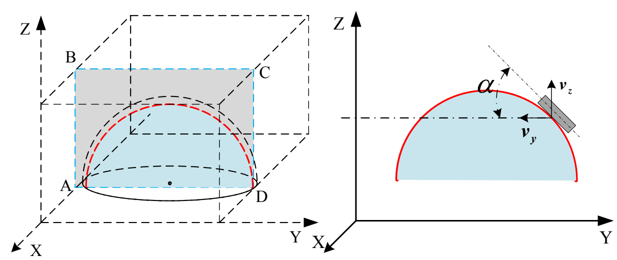
(a) Contour-parallel method (b)The movement of polishing tool

Figure 4. Contour-parallel pattern.

The path planning method not only needs the accurate trajectory but also needs to guarantee the correct pose of the tool. The movement of the polishing tool is shown in Fig. 4(b). In the process of machining, the feed motion of the tool includes the movement on the cutting plane and the rotation around the point *P*. The rotation motion of the tool main includes the rotation of the cycloid angle and the elevation angle. The motion on the plane is that the tool moves along the X-axis and Y-axis, namely the v_x and v_y.

2) Direction-parallel pattern

Likewise, a cluster of cutting planes which are vertical to X-axis cut the surface of the work-piece. The cutting plane intersects with the work-piece generating the color one and its boundary red line is the path of polishing tool in Fig.5(a).



(a) Direction-parallel method (b)The movement of polishing tool

Figure 5. Direction-parallel pattern.

The movement of the polishing tool is shown in Fig. 5(b). The feed motion of the tool includes the movement on the cutting plane and the rotation. The rotation of the tool is the change of the α .The movement on the plane includes the motion along Y-axis and Z-axis, namely v_x and v_y .

By analyzing the above two patterns, we find that the contour-parallel pattern makes tool have two linear motions and two rotation motions, but the direction-parallel pattern makes the tool have two linear motions and one rotation. Compared with the contour-parallel pattern, the direction-parallel path pattern needs less degree of freedom and is much easier to avoid physical interventions and mechanic singularity, so we adopt the direction-parallel method to plan the path.

B. Planning Algorithm

This paper uses the direction-parallel method to plan the path. Two different directions paths sliced by a series of parallel plane are shown in Fig. 6.

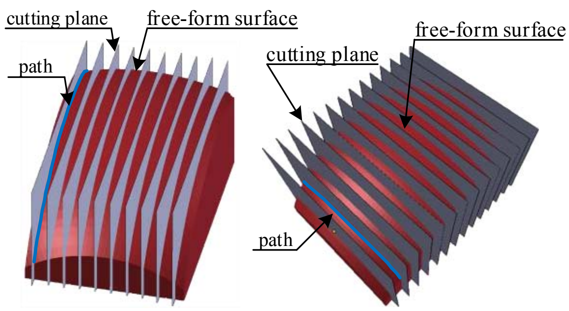


Figure 6. Different directions of path on the free-form surface.

1) Calculate the coordinate of intersection.

The cutting plane, which is parallel to the plane YOZ, and the triangle *ABC* intersect at the point *P* as shown in Fig. 7.

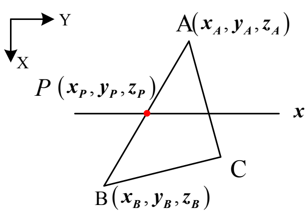


Figure 7. The cutting plane and the triangle intersect.

The x of the cutting plane is known, namely the x of the point P is known. According to the character of the straight line and $x_p = x$, we can get

$$\begin{cases} y_p = \dfrac{x - x_A}{x_B - x_A}\left(y_B - y_A\right) + y_A \\ z_p = \dfrac{x - x_A}{x_B - x_A}\left(z_B - z_A\right) + z_A \end{cases} \tag{4}$$

Then the coordinate of the intersection has been calculated.

2) Path planning based on topology relations

We get the boundary of the free-form surface extracted previous section which are used to calculate the first and the last intersection of one path. We calculate the range of the free-form surface along cutting direction, so we can get the coordinate of the cutting plane according to the cutting distance. The algorithm of path planning is as follows.

Step1: Calculate the x coordinate of the cutting plane $X[i]$. Declare the vector container *nodeVec* which stores the intersection.

Step2: Traverse the boundary half edges array and get two half edges $bHF_i[2]$ which intersect with the cutting plane. Declare the current triangle which has the boundary half edge $bHF_i[0]$. Add the intersection between the plane and $bHF_i[0]$ into the *nodeVec*.

Step3: Traverse the three half edges of the current triangle and calculate the intersection between the plane and the half edge. If the intersection is new, add the intersection into the *nodeVec*. Else, continue.

Step4: Update the current triangle. If the intersection last searched lies in the half edge, we update the current triangle to the adjacent triangle of the half edge. If the intersection lies in the vertex, we traverse the adjacent triangles of the vertex and update the current triangle to the adjacent triangle which intersects to the plane.

Step5: Judge if the half edge last searched is the boundary half edge. If so, end. If not, return step3.

The example is shown in Fig. 8. The point P_1 is the first intersection between the cutting plane and boundary. Search the other two half edges of the triangle which has the half edge above and find the second point P_2. Update the current triangle to F_2 which is adjacent to the half edge having the intersection P_2. Search the other two half edges of F_2 and find the section P_3. The point P_3 lies in the vertex, so traverse the adjacent triangles of the vertex and update the triangle to F_5. Then we can get the intersection P_4. Then it is end because the P_4 is on the boundary.

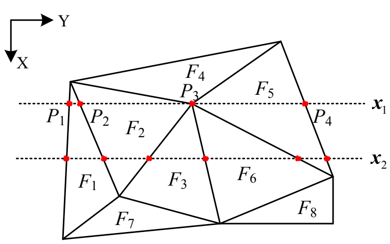


Figure 8. The cutting plane and many triangles intersect.

The steps above help us get one path, then update coordinate of the cutting plane and we can get a series of paths.

IV. EXPERIMENT RESULTS

A. Experiment Environment

The experiment environment is in the Qt creator 4.03 based on Qt 5.6.1. In order to display the STL model and the paths planned by the above algorithm, we use the OpenGL which is a cross-platform application programming interface. The development of user interface software and the calculation of the path planning is in Qt creator 4.03.

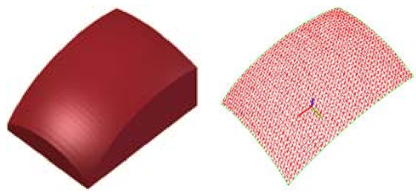
B. Path Planning Results

We conduct the experiment on two representative surfaces. The results are as follows.

1) An object with a free-form surface

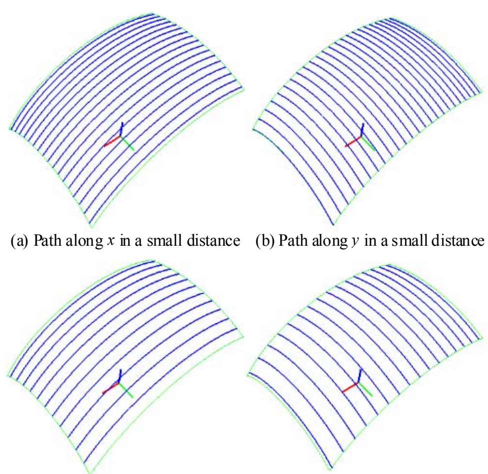
The object is shown in Fig. 9(a) and the free-form surface extracted to be polished is shown in Fig. 9(b).

The results of path planning are shown in Fig. 10. For the different directions and different distances of the path planning, we get the desired results. The cutting planes vertical to Y-axis cut the free-form surface in a small distance shown in Fig. 10(a). The cutting planes vertical to X-axis cut the free-form surface in a small distance shown in Fig. 10(b). The cutting planes vertical to Y-axis cut the free-form surface in a big distance shown in Fig. 10(c). The cutting planes vertical to X-axis cut the free-form surface in a big distance shown in Fig. 10(d).



(a) The object (b) Free-form surface

Figure 9. The object and the free-form surface to be polished.



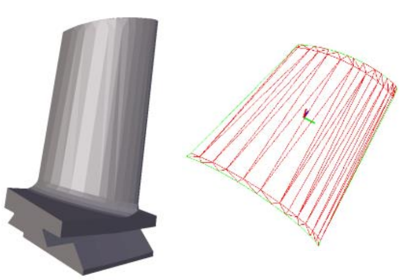
(a) Path along x in a small distance (b) Path along y in a small distance

(c) Path along x in a big distance (d) Path along y in a big distance

Figure 10. The results of path planning.

2) *The stator blade*

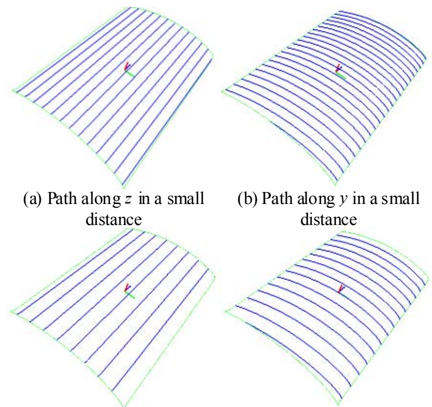
We also conduct the experiment on the blade surface. The CAD model of the blade is shown in Fig. 11(a) and the free-form surface is shown in Fig. 11(b).



(a) The blade (b) Free-form surface

Figure 11. The CAD model of the blade and the free-form surface to be polished.

The results of the path planning for the blade are shown in Fig. 12. We conduct the experiment for different path directions and different cutting distance. We also get the desired paths of the blade.



(a) Path along z in a small distance (b) Path along y in a small distance

(c) Path along z in a big distance (d) Path along y in a big distance

Figure 12. The results of path planning on the blade surface.

V. Conclusion and Future Work

In this paper, we utilize the direction-parallel method to plan the path of the free-form surface. Before the path planning, we read STL files, build topology relations and extract the polishing surface that lay the foundation of the latter work. Then comparing the two path patterns, we adopt the direction-parallel method to plan the path at last. We test our method by planning the path on two representative surfaces. The experiment proves that the path generated by direction-parallel method based on topology relations is accurate.

Future work will concentrate on polishing techniques, polishing system and improvement of path planning.

References

[1] Daqi Li, Lei Zhang, Ji Zhao, Xu Yang and Shijun Ji, "Research on polishing path planning and simulation of small mobile robot," 2009 International Conference on Mechatronics and Automation, Changchun, 2009, pp. 4941-4945.

[2] B. Tsong-Jye Ng, Wen-Jong Lin, Xiaoqi Chen, Zhiming Gong and JingBing Zhang, "Intelligent system for turbine blade overhaul using robust profile re-construction algorithm," ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004., Kunming, China, 2004, pp. 178-183 Vol. 1.

[3] F. Nagata, T. Hase, Z. Haga, K. Watanabe. "CAD/CAM-based position/force controller for a mold polishing robot," Mechatronics, vol.17, pp. 207-216, 2007.

[4] K. Zhang, G. Zhu, S. Liu, B. Qian, X. Zhang and C. Zhang, "Path planning for machining on surface of a blade," 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, 2017, pp. 2093-2098.

[5] Xu Yang, Ji Zhao, Lei Zhang and Daqi Li, "Research on the automatic belt grinding system for machining blade with complex surface," 2010 2nd International Conference on Advanced Computer Control, Shenyang, 2010, pp. 530-534.

[6] J. M. Zhan, X.Q. Zhou, L.Y. Hu, "Study on Path Planning for Industrial Robots in Free-Form Surfaces Polishing", Key Engineering Materials, Vols. 392-394, pp. 771-776, 2009.