Triangle-mesh Based Cutter-Workpiece Engagement Extraction for General Milling Processes

by

Xun Gong

B.Eng., Zhejiang University, China, 2007

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2013

Abstract

This thesis presents a novel geometric modeling methodology of cutter-workpiece engagement extraction for general milling processes. Cutter-workpiece engagement (CWE) geometry is the instantaneous contact area between the cutter and the in-process workpiece. It defines how the cutting edge enters and exits the workpiece. It plays a crucial role for process simulation and directly effects the calculation of cutting force, torque and et cetera. Based on the result of physical simulation, the milling process can be optimized and the machining performance can be improved. Successful optimization depends on the accuracy of the extracted CWE.

The difficulty and challenge of CWE extraction comes from various types of cutters, changing geometry of in-process workpiece and multi-axis tool path of cutter movement. Existing methods confront difficulty to be available for general milling processes, which means for any type of cutter, any shape of in-process workpiece and any tool path, even with self-intersections. To fulfill the requirement of generality, this thesis proposes to model all geometries as triangle meshes throughout the simulation and certain strategy of CWE extraction is applied. Our methodology adopts ball pivoting algorithm for cutter swept volume generation. Octree space partition method is applied to speed up triangle-to-triangle intersection calculation which is used for Boolean operation between meshes. The reported method has been tested on several case studies of different complexity. The effectiveness of the proposed methodology shows its potential for further applications.

Preface

This dissertation is original, independent work by the author, Xun Gong.

A version of Chapter 2 & 3 has been published as Gong, X., and Feng, H. Y., 2013, “Triangle Mesh Based In-Process Workpiece Update for General Milling Processes,” Proceedings of the ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Paper #DETC2013-12408 (9 pages), Portland, Oregon, August 4-7, 2013. I am responsible for writing the complete draft of the ASME conference paper. The published version contains some minor revisions made by my supervisor, Professor H.-Y. Feng.

Table of Contents

List of Figures

List of Abbreviations

CWE	cutter-workpiece engagement
SV	swept volume
BPA	ball-pivoting algorithm
RV	removal volume

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Steve Feng, who has afforded me invaluable instruction and inspiration throughout these two years. He cares my life no less than my research. I feel so lucky to be a student of Professor Feng.

I would like to thank my examiners, Dr. Yusuf Altintas and Dr. Xiaodong Lu, who have spent time to come to my thesis defense. Their comments are much appreciated.

I would like to thank my parents and my girlfriend Rita. Without their constant support and understanding all my achievement would be impossible.

I would like to thank Vinson, who has been a wonderful roommate for years. We have shared much unforgettable time together. I would also thank Youtai and Yifei. They are like my family across the Pacific Ocean.

Last but not least, I would like to thank all my friends in UBC and Vancouver. I will never forget the happy time I have experienced as I lived and studied in such a wonderful place.

To my parents ~

Chapter 1: Introduction

Milling operation is a dynamic process in which the cutting condition and the in-process workpiece changes continuously. As today's manufacturing process is more complex and costly, to design and simulate the machining process in virtual environment becomes a necessary procedure in order to control the cutting condition effectively. Through the process simulation, specialists are able to visualize the product and predict cutting conditions before real machining. This will help them to optimize the milling operation and improve the overall performance. Basically it takes two steps to simulate the entire process: geometric modeling and physical modeling. The outputs of geometric modeling are available for physical simulation. Geometric modeling is the foundation of the system and it has to be as accurate as possible. The most important output is the cutter-workpiece engagement (CWE). It is the bridge that connects the two successive modeling procedures.

The following sections in this chapter further elaborate on the definition and importance of CWE in virtual machining and discuss the strategy for CWE extraction. After the selection of CWE extraction strategy and the confirmation of geometric modeling method with relative literature review, the detailed procedure of our methodology is explained at last.

1.1 Cutter-Workpiece Engagement Extraction

The main focus of geometric simulation is to correctly extract the CWE to prepare for physical simulation. The geometric meaning of CWE is the instantaneous contact area between the cutter and the in-process workpiece. As the cutter moves towards the workpiece, it removes the material of workpiece that is in contact with the cutter. The removed material is the reference for physical simulation.

1.1.1 Mechanics of Milling Process

In Altintas's model of cutting force in turning processes [1], cutting force is a function of machined chip thickness h. While in milling as shown in Fig. 1.1, the instantaneous chip thickness varies as a function of immersion angle ϕ of the cutting edge. It is approximated calculated as $h(\phi) = c \sin\phi$ where c is the feed rate. As a result the cutting force, the axial cutting force becomes $F_x(\phi)$, $F_y(\phi)$ and $F_z(\phi)$ only when $\phi_{st} \le \phi \le \phi_{ex}$. ϕ_{st} and ϕ_{ex} are the entry and exit angle of cutter respectively. The two angles define how the cutting edge enters and exits workpiece as it rotates during milling process. In this mechanics model, they are also the key parameter for the prediction of cutting force, torque and vibration.

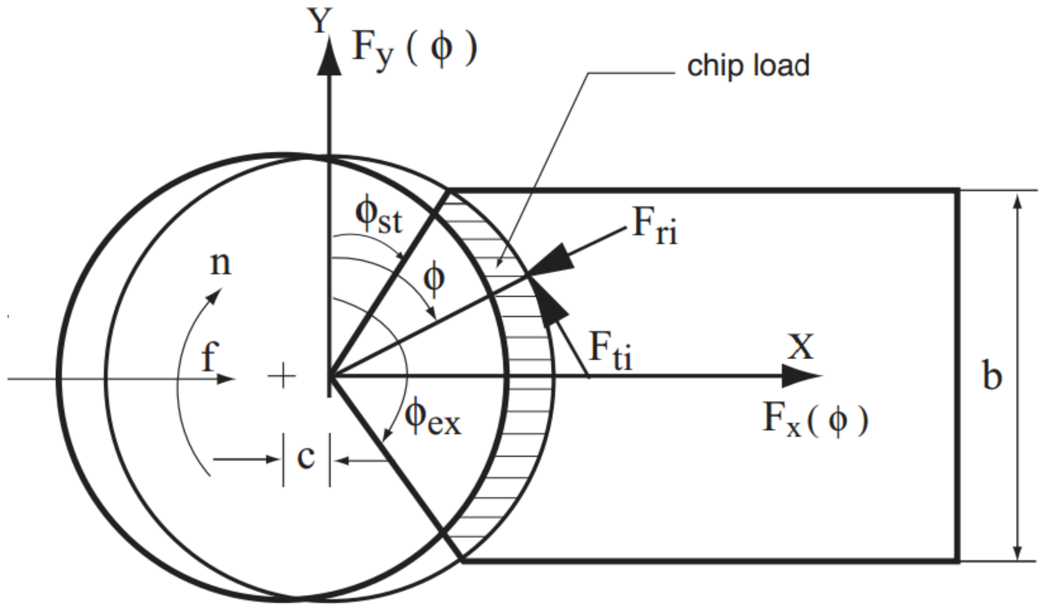


Figure 1.1: Geometry of milling process. [1]

1.1.2 Definition of Cutter Workpiece Engagement

Fig. 1.1 represents a 2D geometry of milling process. When doing physical simulation for the whole cutter, the cutter is divided into many slices. For each slice the mechanics model is applied and then the separated results compose the final simulation result. Thus, besides the geometry meaning, CWE is also a diagram that contains the entry/exit angles for each depth of cut as shown in Fig. 1.2.

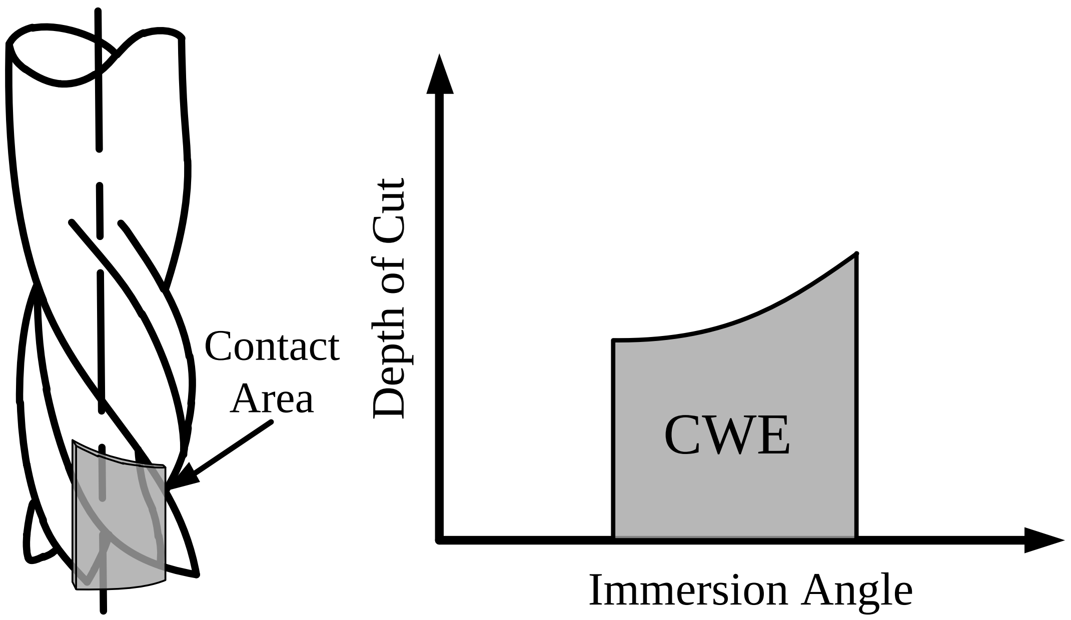


Figure 1.2: Cutter workpiece engagement diagram.

A successful physical simulation highly relies on the accuracy of CWE. People have been working on CWE under different modeling/simulation frames [2] but the applications of their work are always limited by the cutter geometry or tool path complexity. Before we proceed to simulate the geometric process, an appropriate strategy of CWE extraction must be selected based on literature review.

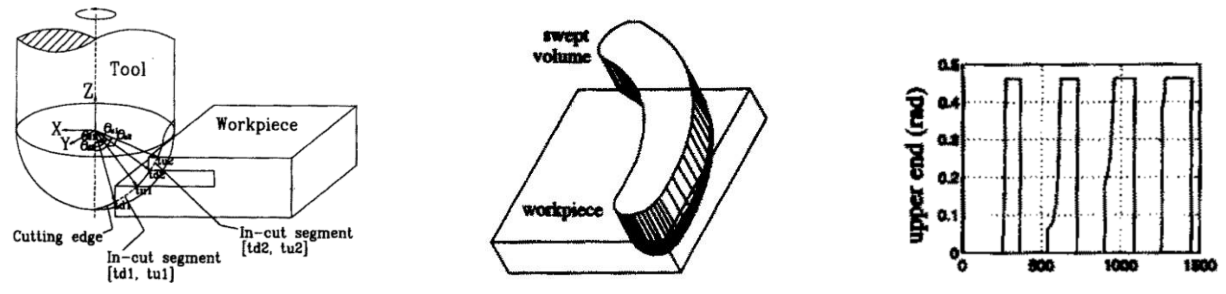
1.1.3 CWE Extraction Strategy

Conventionally, all CWE extraction methods are classified under two major categories: solid modeler based approach and discrete modeler based approach. The first approach is accurate but slow and the latter one is approximate but fast. In addition, each method inherits certain constraint from the geometric modeling. For example, some methods are designed only for ball-end mill and some methods are not available for 5-axis milling process. In this case, the detail algorithm of existing methods makes no reference value for further study, especially when the geometric modeling system is totally different. However, the general strategy for CWE extraction, which is isolated of how the cutter and workpiece are modeled, may provide useful information for our strategy selection. There are basically three kinds of strategies.

The first strategy offers that the CWE to be analytically calculated at each moment for each entire tool path. In 1997, EI-Mounayri et al. [3] models the boundary of workpiece as plane surfaces and the cutting edge as a Bezier curve in space. For each cutter's angular position, the Bezier curve is calculated and the intersection between cutting edge and workpiece is obtained as shown in Fig. 1.3.

.

In the same year, Imani et al. [4] use the similar method to extract in-cut segment of cutting edge for each angular position. Later Bailey et al.[5] model the cutting edge as NURBS curve. At each angular position, every point along the cutting edge is identified as in-cut or not according to previous tool path.



(a) (b) (c)

Figure 1.3: (a) definition of in-cut segment; (b) milling case; (c) result CWE diagram. [3]

In the second strategy, the cutter swept volume is first generated. This SV is used to update the in-process workpiece. Meanwhile the CWE is calculated for this short period. In the work by Sadeghi et al.[6], the cutting edge is modeled as the intersection of a plane and the ball part of the ball-end mill. After the workpiece is updated, the contact face is modeled and the immersion part of the cutter is calculated by intersecting the cutting edge and the contact surface boundary.

In 2003, Fussel et al. [7] use the swept volume to clip the vectors representing the surface of the workpiece. Fig. 1.4 explained their CWE extraction process. The workpiece is represented by a set of vectors called Z-buffer elements (ZDVs). While the workpiece is updated, the cutter SV clipped ZDV and the in-cut segment of ZDV is from P_{ent} to P_{ext} (Fig. 1.4(a)). Then segment (P_{ent}, P_{ext}) is projected to the surface of cutter. Finally all the projected in-cut segments by the SV compose the CWE on the cutter. In 2010, Yao et al. [8] use facets to model the tessellated free-form surface of workpiece. They generate cutter swept volume over fairly small step and then the cut-through facets are processed to calculate the CWE.

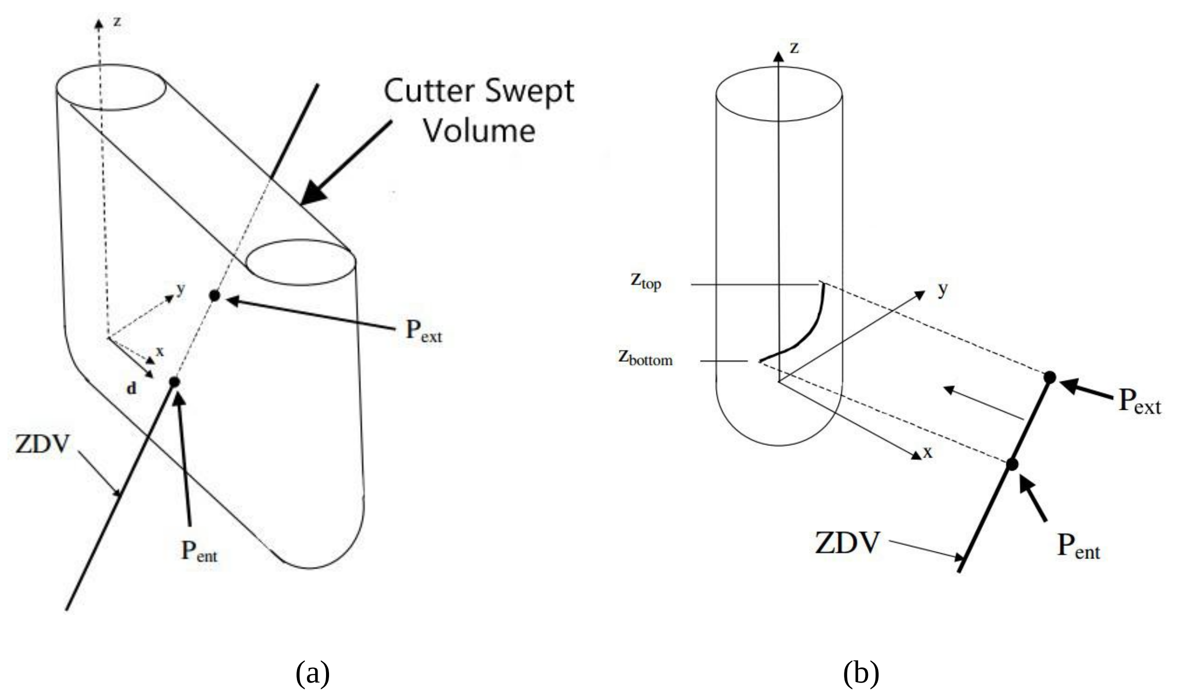
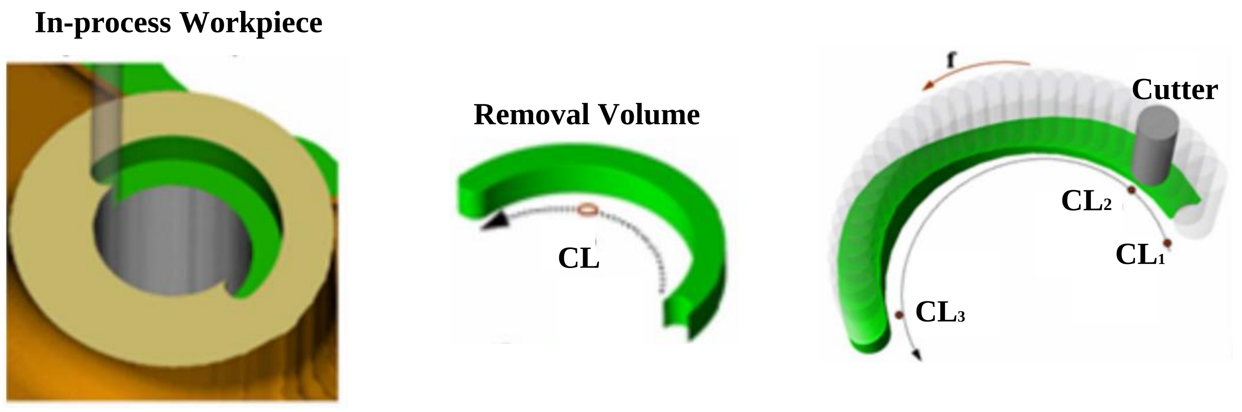


Figure 1.4: (a) definition of in-cut segment; (b) in-cut segment projection to form CWE. [7]

The last strategy is to calculate the CWE for intermediate cutter positions along a tool path. In 2005, Larue and Altintas [9] develop a method to calculate the intersection of taper ball end mill and ruled surface of workpiece. At each cutter position the CWE is generated by intersecting the cutter and workpiece. The result is a set of curves which are used for immersion angle calculation. Later in 2008 Aras et al. [10] use the removal volume (RV) to extract CWE. RV is the material of workpiece that is machined by cutter SV for certain tool path in Fig. 1.5(a)-(b). They place the cutter evenly along the tool path and CWE is extracted at each location as shown in Fig. 1.5(c). At the same time Ferry et al. [11] develop an analytic method using RV based on solid modeler. The significance of their work is that they introduced the concept of "RV update" to deal with self-intersecting tool path, which is quite normal in five-axis machining.



(a) (b) (c)

Figure 1.5: (a)-(b) removal volume generation; (b) CWE extraction along tool path. [10]

The strategies that we mentioned above have covered most of the existing methods. The first one requires numerous calculations as intersection is calculated at every angular position, however, this is not quite necessary. In milling operation, the spindle speed is much larger than the feed rate. At certain cutter location, the engagement area contains adequate information for further calculation. The second strategy introduces the CWE area which is directly used for physical simulation. Nevertheless, to generate sufficient CWE with small location interval along one tool path requires much more cutter swept volumes to be created priorly. The last strategy starts to use removal volume, which is much smaller than the whole workpiece, to generate CWE along the tool path. By adopting the “RV update” concept, the biggest concern about self-intersection tool path is solved. According to the comparison, the third strategy seems the most suitable one for CWE extraction for general milling processes. SV is generated and used to update the workpiece while RV is obtained. For certain cutter location along the tool path, the RV has to be updated first and CWE can be extracted correctly.

1.2 Cutter Swept Volume Generation

After the conceptual strategy is decided, specific geometric modeling method needs to be selected for our strategy. Simplicity, adaptability and accuracy are the main concerns for geometric modeling methods. The first key process of geometric modeling is the cutter swept volume generation and it has been a popular topic for years. Several different approaches have been developed.

Envelope theory is the first approach used for analytic modeling in which the cutter is modeled by explicit and implicit expressions. In envelope theory, the SV is represented by its boundary surface, which is denoted as envelope surface. According to the tangent condition, the envelope surface is always tangent to the cutter surface during cutter movement. In other words, the normal of cutter is parallel to the normal of envelope at the contact point. From another perspective, the vector of velocity is perpendicular to the normal of envelope.

Envelope theory and tangent condition property is the basis for SV generation and proposed by Wang et al. in 1986 [12]. Later Chiou et al. [13] [14] develop a closed-form solution of the swept profile of a generalized cutter for five-axis NC machining. To generate the analytic model of the SV, Du et al. [15] [16] adopt the method of moving frame and explicit representation introduced by Wang and Chiou. Envelope profiles are formulated and from these profiles the SV surface is constructed by non-uniform rational B-spline (NURBS) surfaces.

Envelope theory provides solid fundamental concept of SV generation but it is quite slow. To speed up the tool path simulation, the imprint method is developed. At each instance, there exists a curve on the cutting surface of the tool that describes the contribution of the cutter position to the final bottom swept surface. This curve is called the “generating curve” and it represents the imprint of the cutter on the machined. Sheltami et al. [17] first approximate the generating curves of toroidal cutter as circles for three-axis movements. Under Sheltami’s concept, Roth et al. [18] [19] use silhouette method to identify imprint curves as space curves for five-axis motions. Mann et al. [20] further generalize the imprint method to general surface of revolution instead of merely toroidal surface.

To cover a larger set of cutters, new parametric approaches are developed focusing on cutters with canal surfaces. As described in the work by Pottmann and Peternel [21], general canal surfaces are defined as the envelope of a family of spheres which are located along a spline curve and tangent to both surfaces. The envelope of SV can be then efficiently generated by considering two families of spheres centered along the tool trajectory with varying radius. Aras [22] uses the same methodology to analytically generate graze points for swept surface of a milling tool undergoing five-axis movement. At the same time Gong et al. [23] propose two closed-form solutions for calculating the envelope surface of a generic tool directly from CL-data. A special characteristic of surface of revolution during the enveloping process is developed, and it is used to simplify the calculation of the envelope surface for surfaces of revolution greatly.

The latest and novel approach is the Gauss map method proposed by Seok et al. [24] [25]. The tool shape is transformed from Euclidean space into tool map (T-map) on the unit sphere and the velocity vector of a cutter is transformed into contact map (C-map) using the Gauss map. By finding the intersecting regions between the two maps, closed intersection curves are found and used to establish the swept surface. They made their efforts to solve the tool path self-intersection problem, but the application is restricted to fillet end, ball end and flat end mills.

For all the above significant research work, they are based on analytic methods and usually use exact expression of cutting profile to compose the whole SV. This type of methods are limited by certain constraints. When modeling the cutter using mathematical expression, it is hard to model sharp edge/point on the surface. For some methods they require different algorithms to adjust for each type of cutter with unique analytic model. For some methods it is impossible to obtain the SV as a whole closed form model. In addition, most methods cannot deal with tool path self-intersection situation. This conclusion somehow indicates that analytic modeling method has potential limitations for general applications. These obstacles motivate us to consider approximate modeling method for this problem.

Approximated approach has appeared several times in previous research. Blackmore et al. [26] start their work with complex math, but they triangulate the cutter first to estimate the graze points. For Seok's work, they generated the SV by triangulation to simplify the following procedures after obtained the egress/ingress surfaces and interpolated graze points.

Triangle mesh model is believed to be the most promising modeling method for this topic and has the following advantages: (1) Triangle mesh models the solid with a set of connected triangles, which transforms the 3D model into 2D representations and simplifies calculation for proceeding operations. (2) Triangle mesh is able to model the cutter in a closed form and more importantly, it has no limitation on the surface features and can be used to model any type of cutter. (3) Triangle mesh is compatible with most CAD software as they are all able to generate STL files which represent models in triangles. Triangle model is also popular in computer graphics area and ready for rendering and other operations. Based on these considerations, we believe that triangle mesh is the modeling approach that should be used for cutter modeling and SV generation.

1.3 Workpiece Modeling

The modeling of workpiece is the other part of this methodology which is as important as cutter SV generation. All cutting conditions in milling process are calculated based on the interaction between cutter and in-process workpiece during simulation. In addition, this virtual workpiece can be used to verify the final result of milling process. Performance of simulation includes execution time, accuracy and memory needs and these factors vary with different modeling approach. Basically the workpiece can be modeled using solid modeling or approximate sampling representation.

The direct solid modeling approach is typically implemented by Constructive Solid Geometry and NURBS based Boundary representation. This approach is capable for accurate modeling and its history-based feature requires a lot of memory. For a complex machining process, the number of tool paths can be as large as thousands and the computational complexity makes the physical simulation impractical. This major disadvantage of solid modeling approach motives the use of approximate sampling representation. The model of workpiece is decomposed into a collection of basic geometric primitives, and the intermediate workpiece during simulation is extracted directly after each tool path update. There have been several proposed approximate representation schemes for the workpiece model, including Z-map, vector, voxel, dexel and polyhedral representation.

Z map representation [27] [28] is the current prevailing approach for workpiece modeling because of its simplicity and robustness. Z map represents the workpiece as a set of vectors along z-axis, thus impossible to model overhang shape and impractical to accurately represent vertical wall or sharp edge features. To overcome this deficiency, Jerard et al. [29] develop discrete vector model which model the surface of workpiece as vectors that are normal to the surface. Later Park et al. [30] adopt a hybrid modeling method which combines discrete vertical vectors (DVV) and discrete normal vectors (DNV) together. This hybrid approach is adequate to model in-process workpiece for roughing operation and look into sharp features for finishing operation. However, if users want to simulate cutting condition for intermediate workpiece, this model is not sufficient since it lacks feature information.

The voxel-based representation which brought up by Jang et al. [31] is developed for both the cutter and the workpiece. They claim that this approach has advantage on Boolean operations for workpiece update, yet they did not talk about accuracy issues and both feature representation and preservation are not clear.

The latest novel modeling approach is the triple dexel model (TDM) which is extended from the single dexel model (SDM). It is proposed by Seok et al. [32] with good demonstration. In their method, the workpiece is modeled using orthogonal triple dexel model in three directions. These dexels are trimmed after each tool path update and then the surface is constructed by marching cubes approach. Marching cubes approach faces several disadvantages like poor triangle quality and cannot represent features such as sharp edges or corners. Although they attempt to fix these problems by inserting new points, the point representing sharp feature is merely an estimate instead of accurately calculated. Moreover, the surface normal is hard to obtain after the workpiece is updated by the cutter swept volume for multi-axis movement. Thus the accuracy of the model is not studied in depth.

From the previous research work we notice that virtual workpiece modeling is not an easy task and the challenges are very obvious. Basically the desired modeling should be able to fulfill three requirements: (1) the update process should be fast, accurate and robust; (2) the in-process workpiece should be accessible directly and able for physical simulation; (3) many surface features, including overhang shape, vertical walls and sharp edges should be preserved not only for finishing operation but also for roughing operation. These features have great impact on the result on prediction of cutting force, torques and etc. One promising modeling approach which can fulfill all the above requirements is again the triangle mesh modeling. Unlike other approximate modeling, triangle mesh is the only method to model the workpiece geometry as a water tight integrity. It is geometrically isotropic thus able to represent surface details in any direction. It has no limitation on feature preservation no matter it is over hang shape, vertical wall or sharp edges. As well, when the cutter swept volume and workpiece are both modeled as triangle mesh, the Boolean operation is only a triangle-to-triangle 2-dimension calculation. The intersection between two meshes can be generated accurately. The accuracy of these intersections, most of which contribute to sharp features of updated workpiece, will guarantee an accurate simulation result.

After we reviewed the existing methods for modeling cutter swept volume and workpiece and discussed their limitations, triangle mesh modeling is proposed and interpreted. The detailed procedure of our methodology is demonstrated in Fig. 1.6. Initially, the model of cutter and model of workpiece are triangulated. First, cutter SV #1 is generated along certain tool path segment. Then Boolean operation between SV #1 and the workpiece is taken to get the updated workpiece and the RV of SV #1. Second, if we want to know the CWE at any location, SV #2 needs to be generated from the beginning of this segment to this location. This SV is used to update RV to avoid overestimating the CWE area when tool path has self-intersection. The final step is to calculate the overlap between the cutter at this location and the up-to-date RV for CWE. In the following sections of this paper, cutter swept volume generation, workpiece update and CWE extraction will be elaborated in more details.

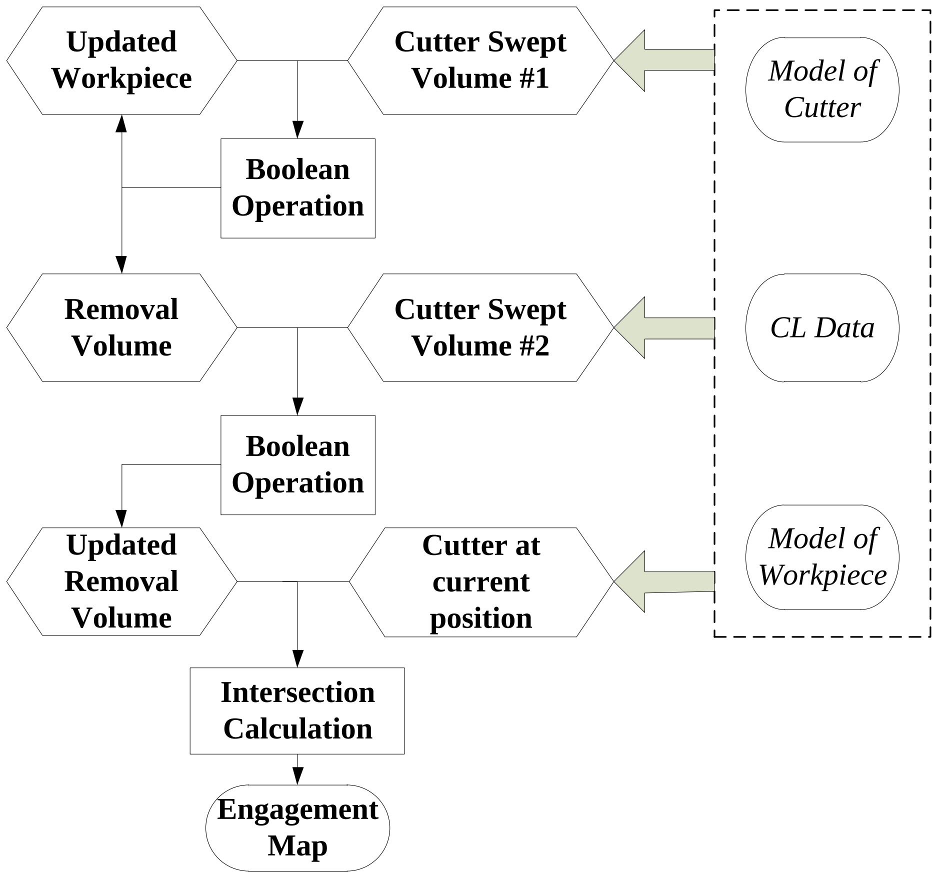


Figure 1.6: Flow chart for cutter-workpiece engagement extraction.

Chapter 2: Cutter Swept Volume Generation

During milling operation, the cutting condition and in-process workpiece changes continuously. However, in the simulation environment, digital computer can only work in discrete time domain thus it simulates the milling operation as interaction between model of workpiece and model of cutter swept volume during a time period. Cutter SV is the space that the cutter occupies as it moves between two CL point and SV generation is the first step of geometric modeling. To simulate the material removal process, SV is subtracted from the workpiece and part of the surface of SV updates the local surface of workpiece. SV can be very intricate and hard to generate because of either the complex cutter geometry or multi-axis tool path. For general cases, existing methods would confront many difficulties and become invalid. The following modeling and generating approach for cutter SV adopt approximated triangle-mesh representation and solve the previous issue.

First the cutter is modeled as triangle mesh. The triangles and their vertices are used to represent the surface of the cutter. Then triangles are abandoned and only vertices are repeatedly sampled along the trajectory. Since many vertices are actually lying inside of the cutter SV, most of these vertices are removed by a filtering operation according to their normal and a point cloud which approximately outlines the surface of cutter SV is obtained. Then the ball-pivoting algorithm which is modified for our case is adopted to construct the surface of the point cloud. At last, a close manifold triangle mesh of cutter SV is generated and ready for the next step.

2.1 Triangulation of Cutter Model

Usually cutters are defined in analytical form. There are several typical milling cutters like flat-end mill, ball-end mill, fillet-end mill and taper-ball-end mill. Each of them has its unique geometry features. Their non-uniform representations compel researchers to deal with different features individually and restrict their methods to limited types of cutters. To be applicable to general cutters or even user customized cutter, the intuitive way is to model the cutter surface without considering geometry features.

We choose triangle mesh to model the surface of cutter as a set of connected triangles. Each triangle has three vertices and a normal while each vertex also has a normal itself. Fig. 2.1 displays four triangle mesh cutters including flat-end mill, tapered-ball-end mill, fillet-end mill and generalized cutter. They all share the uniform representation, and are treated equally in the following procedures.

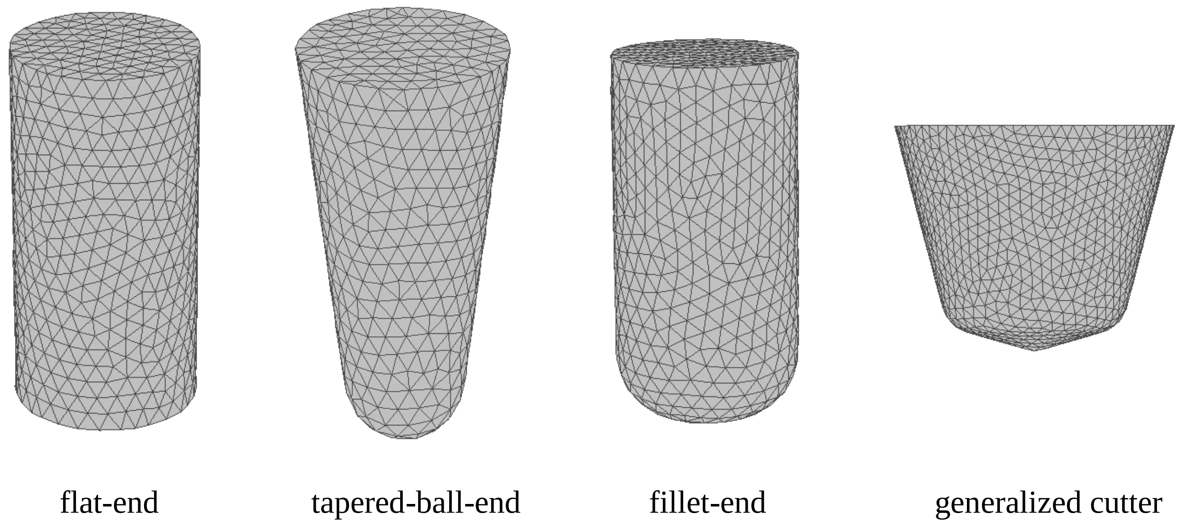


Figure 2.1: Triangulated model of various cutter.

Triangle mesh is a discrete model that differs from the exact model. It uses numerous flat triangles to approximate the smooth surface of the cutter so deviation is unavoidable. When the cutter model is triangulated, the triangle size, which is represented by the average edge length, is user defined. Ideally the triangle size should be selected as small as possible to minimize the error from exact model. However, smaller triangle size leads to more triangles. Reducing edge length by half results in quadruple triangle number, which will slow down the all the rest operations. So in practice the triangle size should be selected appropriately to balance the accuracy and execution time.

2.2 Generation of Point Cloud for Swept Volume

For a designed milling process, CAM software will create a CL data file which contains all the information a milling machine needs to manufacture the desired product including spindle speed, feed rate, cutter location and et cetera. Then this CL file will be post-processed to G-code for specific milling machine to guide the cutter movement in real machining. We choose CL data file for the simulation rather than the G-code file. The advantage of CL data file is that it is universal and independent from machine configuration while G-code file may vary for different machines. By using CL file, our system could simulate the milling process without considering the milling machine at all.

2.2.1 CL Data Interpretation and Application

In the CL file, each CL data contains 6 parameters (X,Y,Z,I,J,K), the first three (X,Y,Z) represent the CL point coordinate and the remaining three (I,J,K) compose a unit vector representing the cutter orientation. If the remaining three are omitted, it means that at this cutter location the cutter orientation is unchanged. A sample CL data file generated by NX looks like the follows,

```
……
SPINDL/6366.000000, RPM, CLW
RAPID
GOTO/-12.9241,48.0671,-5.7167,-0.4147500,-0.0360700,0.9092200
FEDRAT/MMPM,1273.2000
GOTO/1.9264,48.0602,-3.6041
GOTO/4.4361,48.0590,-3.2471,-0.3825657,-0.0134490,0.9238304
GOTO/6.9655,48.0590,-3.0521,-0.3276059,0.0100221,0.9447613
GOTO/9.5072,48.0602,-3.0734,-0.2501283,0.0359958,0.9675433
GOTO/10.7768,48.0615,-3.1730,-0.2058923,0.0502911,0.9772815
GOTO/12.0419,48.0633,-3.3330,-0.1598842,0.0654786,0.9849617
GOTO/14.5553,48.0686,-3.7555,-0.0923112,0.0968677,0.9910072
……
```

To apply CL data on the cutter model, the translational movement vector T and rotational matrix R are derived from $CL(X,Y,Z,I,J,K)$,

$$T = (X, Y, Z) \tag{2.1}$$

$$R = R_z(\varphi)R_y(\theta) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.2}$$

where φ, θ are the axis rotation angle around *Y-axis* and *Z-axis* and are calculated based on the geometric relation between them and the unit orientation vector (I,J,K).

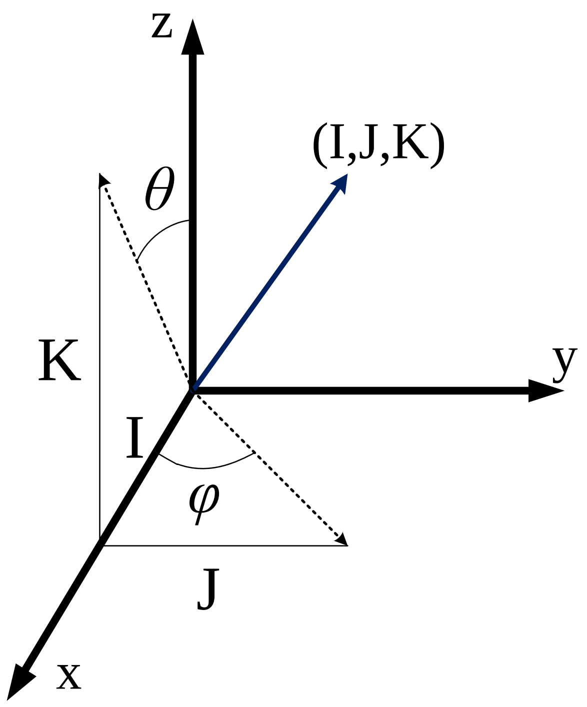


Figure 2.2: Geometric relation between orientation vector and rotational angles.

As shown in Fig. 2.2, the unit vector (I,J,K) represents the orientation of the cutter axis. The two dash arrow lines are the projected vectors on *XZ-plane* and *XY-plane* respectively. From the geometric relation, rotation angle φ and θ can be calculated as

$$\theta = \cos^{-1}(\mathrm{K}) \tag{2.3}$$

$$\varphi = \tan^{-1}(\mathrm{J}/\mathrm{I}) \tag{2.4}$$

To move the cutter to the desired cutter location $CL(X,Y,Z,I,J,K)$, the translational movement and rotational movement are applied separately. First the pivot point of the cutter, which is usually the cutter tip as $Pivot(x, y, z)$, is moved to new position by adding the translational movement vector,

$$new\_Pivot = Pivot(x, y, z) + T \tag{2.5}$$

The pivot point determines the cutter location and is the reference for cutter orientation. Then the orientation of the cutter axis will undergo rotational movement. To simplify the calculation, the rotational matrix R is applied to the relative position vector from the pivot point to every vertex of the cutter. For vertex V_j, its original position is $P_j(x, y, z)$, j is the index of the vertex. Its relative position vector is

$$L_j = P_j(x, y, z) - Pivot(x, y, z) \tag{2.6}$$

By multiplying the rotational matrix R_i to L_j, we get the new relative position from the new pivot point to the vertex. Combine translational and rotational movement together, the new position of vertex V_j is calculated as follows,

$$P_j = Pivot(x,y,z) + T + R \cdot L_j \tag{2.7}$$

Repeat this operation to all vertices and the cutter is replaced at new position as indicated in this CL data. After the procedure to apply one CL data on cutter model is explained, it is ready to generate cutter SV from CL data file.

2.2.2 Cutter Vertices Sampling and Filtering Operation

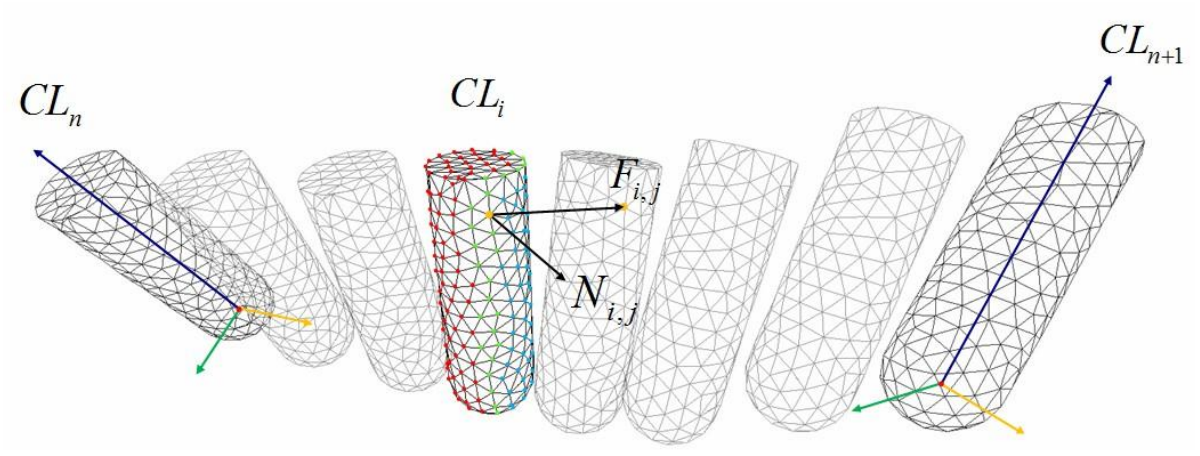
As defined, cutter SV is the space possesses by the cutter as it moves according to the CL file. It is the union of all cutter models at every moment between successive cutter locations. Apparently it is impractical to handle this union operation in simulation environment. In our methodology, we propose to sample the vertices of the cutter model evenly between two CL points CL_n and CL_{n+k} with small interval. The result of the sampling is a point cloud which represents the geometry shape of the SV. It can be used to construct the surface of SV.

First the cutter is copied and placed at each CL position between CL_n and CL_{n+k}. Then cutter sampling by linear interpolation is adopted to fill in the space between neighboring CL points. To be specific, if we want to evenly sample cutters between CL_n and CL_{n+1} as shown in Fig. 2.3(a), the exact position of i th duplicated cutter should be

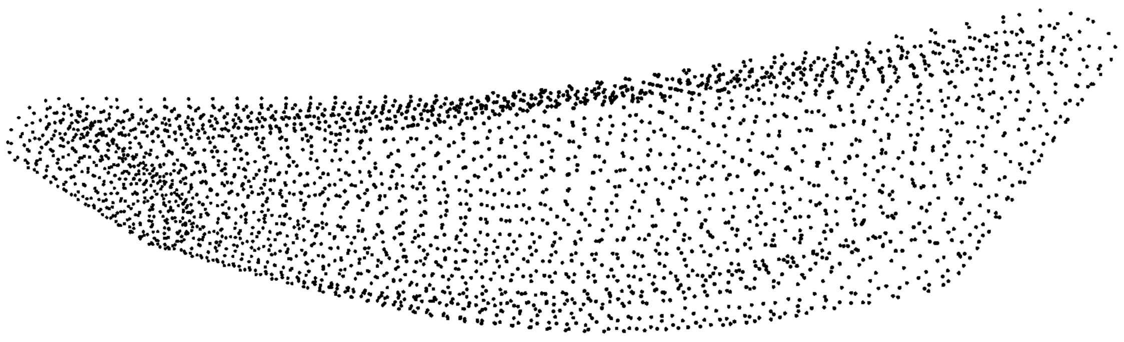
$$CL_i = i\frac{CL_{n+1} - CL_n}{m+1} + CL_n \qquad (2.8)$$

m is the total number of duplicated cutters. It is selected to make the distance of adjacent cutters be average triangle edge length. This will not create too many unnecessary cutters but enough to represent the SV.

Only the vertices and their normal of the cutter are used for SV generation. We remove all the triangles of the cutters, and this would result in a very large point cloud. Most of the points actually lie inside the SV and obviously do not contribute to the surface of SV. This point cloud is available to construct the surface of the SV but these inside redundant points will slow down the operation tremendously. To speed up the surface construction process, a filtering operation is required to remove those redundant points.



(a)



(b)

Figure 2.3: (a) Sampled cutter models along tool path; (b) Point cloud for SV generation.

Feed direction F is a key variable to identify which part of the cutter at each location may contribute to the SV. For multi-axis machining, as cutter orientation changes with time, feed direction may vary along the cutter axis. In this situation, the feed direction of every vertex needs to be calculated individually. Mathematically the feed direction of vertex V_j at time t is the first order derivative of the position function $P_j(t)$

$$F_j(t) = \frac{d}{dt} P_j(t) \tag{2.9}$$

In practice, the position of a vertex is not a continuous function $P_j(t)$. It is a discrete function $P_j[i]$ of interpolated CL data. So the calculation of feed direction $F_j[i]$ at CL_i becomes a unit vector point to the next position of the vertex

$$F_j[i] = \frac{P_j[i+1] - P_j[i]}{\left| P_j[i+1] - P_j[i] \right|} \tag{2.10}$$

Besides feed direction, the vertex normal N_j is the other parameter that is essential for filtering operation. It needs to be updated for every new position since the orientation of cutter is changing. If this normal updating step is omitted, the cutter will still look the same but the normal of all vertices is incorrect. The original normal of V_j is generated during the cutter model triangulation. When cutter moves to CL_i, the rotational matrix R_i is applied directly on N_j, so

$$N_j[i] = R_i \cdot N_j \tag{2.11}$$

The implementation of filtering follows the rules based on the relation between feed direction $F_j[i]$ of each vertex and the normal $N_j[i]$. It is obvious that the surface of the cutter SV are composed of three parts: part of the beginning position which forms the start surface, part of the end position which forms the end surface, and grazing points of intermediate positions which form the side surface of the SV.

For the beginning position of SV, only vertices with normal opposite to its feed direction will be remained. The other vertices are deleted because they do not belong to the surface of SV.

$$V_{i,j} \quad is \quad \begin{cases} deleted, & if \quad \left\langle F_{i,j}, N_{i,j} \right\rangle < 90^\circ \\ remained, & if \quad \left\langle F_{i,j}, N_{i,j} \right\rangle \geq 90^\circ \end{cases} \tag{2.12}$$

For the ending position of SV, the reasoning is similar and the condition is reversed.

$$V_{i,j} \quad is \quad \begin{cases} deleted, & if \quad \left\langle F_{i,j}, N_{i,j} \right\rangle > 90^\circ \\ remained, & if \quad \left\langle F_{i,j}, N_{i,j} \right\rangle \leq 90^\circ \end{cases} \tag{2.13}$$

For each interpolated position, the vertices with normal either opposite to or same as its feed direction will be inside of the SV and useless. Only the vertices with normal perpendicular to its feed direction would remain,

$$V_{i,j} \quad is \quad \begin{cases} deleted, & if \quad \langle F_{i,j}, N_{i,j} \rangle \neq 90^\circ \\ remained, & if \quad \langle F_{i,j}, N_{i,j} \rangle = 90^\circ \end{cases} \tag{2.14}$$

As triangle mesh is an approximate modeling method, the vertices will not cover every point on the surface of the cutter and few points satisfy the condition (2.12) – (2.14). For this consideration, all the filtering conditions become less demanding for implementation. For example, at interpolated positions, the filtering rule becomes

$$V_{i,j} \quad is \quad \begin{cases} deleted, & if \quad \left| \langle F_{i,j}, N_{i,j} \rangle - 90^\circ \right| > \varepsilon \\ remained, & if \quad \left| \langle F_{i,j}, N_{i,j} \rangle - 90^\circ \right| \leq \varepsilon \end{cases} \tag{2.15}$$

ε is a small user defined positive number. By introducing this, the "perpendicular" condition is no longer restricted to 90 degree but $90 \pm \varepsilon$ degree. The purpose of filtering is to decrease the size of the point cloud. A redundant set of points will not cause problem in the following surface reconstruction step but an inadequate one will. The number ε should be selected conservatively according to the vertex density of original cutter model.

After the sampling and filtering operation, we get the desired point cloud as shown in Fig. 2.3(b). This point cloud reveals the shape of SV and will be used to construct the surface of SV.

2.3 Surface Construction of Cutter Swept Volume

Surface construction is a very popular in reverse engineering and industry design. Usually the surface is reconstructed from 3D scanned data. Unlike the scanned points which are more or less on the desired surface of the model, in our application, an arbitrary trajectory may have self-intersection and some points may occur inside the SV even after the filtering operation. For this reason not all points will contribute to the surface and most of the surface reconstruction methods like 3D Delaunay Triangulation are not valid, except the ball-pivoting algorithm (BPA) proposed by F. Bernardini in 1999 [33].

The ball-pivoting algorithm belongs to the region-growing category. The concept behind BPA is quite simple. Given a point cloud, there exists a ball with certain radius β and it cannot pass through the point cloud without touching any point of it. In other words, keep the β-ball in contact with the points and it can "roll" on the surface of the point cloud. So BPA is able to ignore all inside points automatically when constructing the surface. This property makes BPA different from other surface construction methods and suitable for our special application. We can place the β-ball on the point cloud touching three points, then pivot it around. The new touched points are added to form triangles until the whole surface is constructed.

The pivoting operation is shown in Fig. 2.4. The β-ball is touching triangle (V_0, V_1, V_2). This triangle can be either the initial triangle or the previous created triangle by BPA. The pivoting edge (V_0, V_1) is called the "active edge". The pivoting direction is decided by the normal n_f of the triangle so that the ball will not pivot inside of the surface. While pivoting, the ball stays in touch with point V_0 and V_1 until it touches the candidate point V_k.

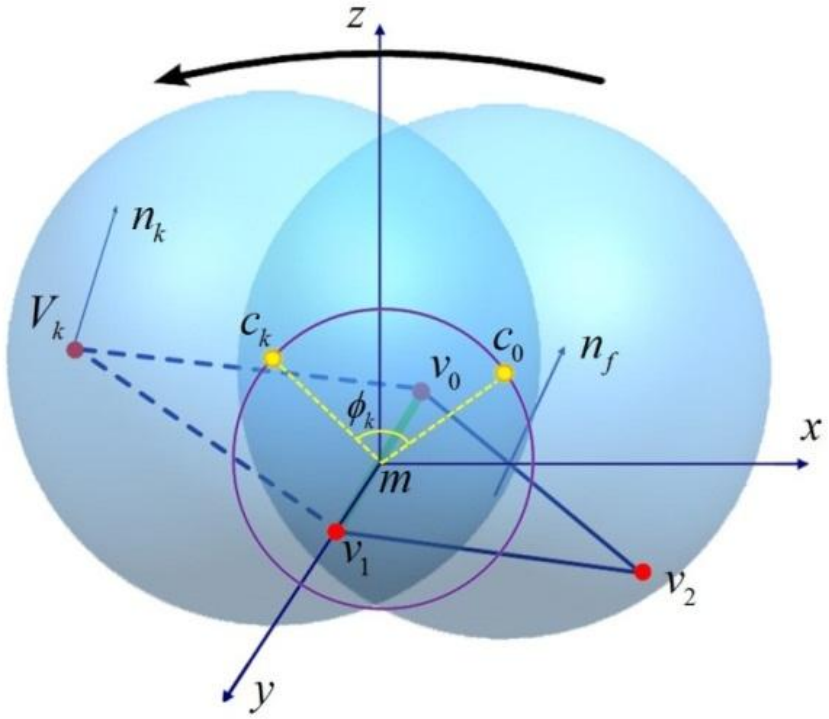


Figure 2.4: Operation of ball pivoting

In implementation, first a plane is created centered at point m, the middle of edge (V_0, V_1), and perpendicular to this edge. This plane restricts the movement of the ball. The trajectory of the ball center is a circle centered at m with radius α on the plane. The radius α is smaller than β. It is calculated as

$$\alpha = \sqrt{\beta^2 - \frac{|V_0V_1|^2}{4}} \tag{2.16}$$

Suppose when the ball touches point V_k, the ball center is at c_k. Obviously the distance from V_k to c_k is the ball radius β. To find point c_k, a virtual sphere is created centered at V_k with radius of β. The intersection point between the circle (m,α) and the virtual sphere (V_k,β) is exactly point c_k. The angle between $\overrightarrow{mc_0}$ and $\overrightarrow{mc_k}$ is the pivoting angle ϕ_k. If triangle (V_0,V_1,V_k) is valid, it will be added to the mesh. In addition, edge (V_0,V_1) is deactivated; edge (V_0,V_k) and edge (V_1,V_k) become active and ready for next pivoting operation.

BPA also has a mechanism to avoid creating too small triangles in order to make the constructed mesh more consistent. User can define a distance threshold. When a new point is added, its surrounding points within the threshold distance are labeled *unusable*. They will not be considered for the following pivoting operations.

The general flow of the surface construction is presented in Fig. 2.5. First the ball size is selected according to the density of the point cloud. The only criteria is to make sure the ball is able to "roll" without falling inside. Then the initial triangle is assigned and its three edges are activated. For each active edge, the ball pivots around it. The ball is assume to touch all nearby points, and for each point the ball center position c_k is calculated. Only the point with smallest pivoting angle $\angle(\overrightarrow{mc_0},\overrightarrow{mc_k})$ is selected to form the new triangle. This selection guarantees that the ball does not contain any other point in it. After new triangle is added to the mesh, new pivoting operations are proceeded until there is no more active edge. At this point a closed manifold mesh is constructed.

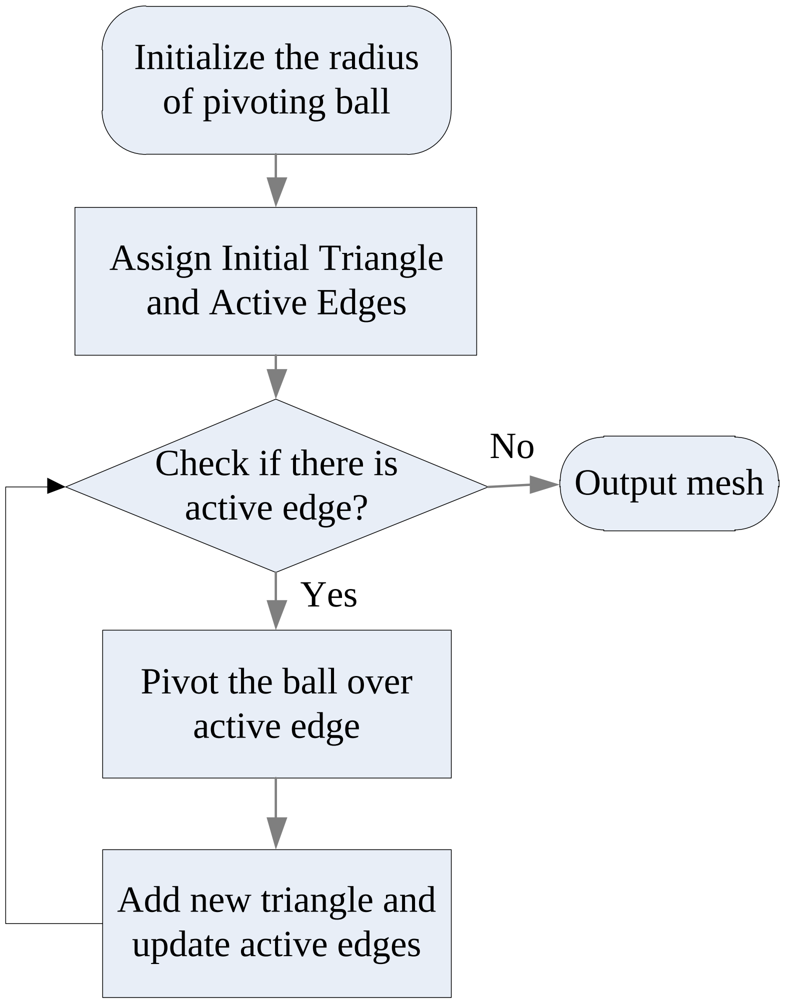

Initialize the radius of pivoting ball
Assign Initial Triangle and Active Edges
Check if there is active edge?
No
Output mesh
Yes
Pivot the ball over active edge
Add new triangle and update active edges

Figure 2.5: Flow chart of surface construction

For further details about the BPA, please refer to [33]. However, two parts need to be highlighted in this flow chart because they are modified from the original BPA. First, there is only one initial triangle in our method. While in the original BPA, the initial triangle is assigned more than once. The original BPA tries to reconstruct mesh for the whole point cloud and searches for new initial triangle when there is no active edge. In our application, some points are inside the surface, and they do not contribute to the mesh. The surface of the cutter SV is one integer and only one initial triangle is enough. Second is that our initial triangle is assigned to points which are definitely on the surface, while the initial BPA assigned it randomly. The reasoning is similar to the first one. If the initial triangle is assigned unluckily to the points inside the cutter SV, the output of BPA is absolutely wrong.

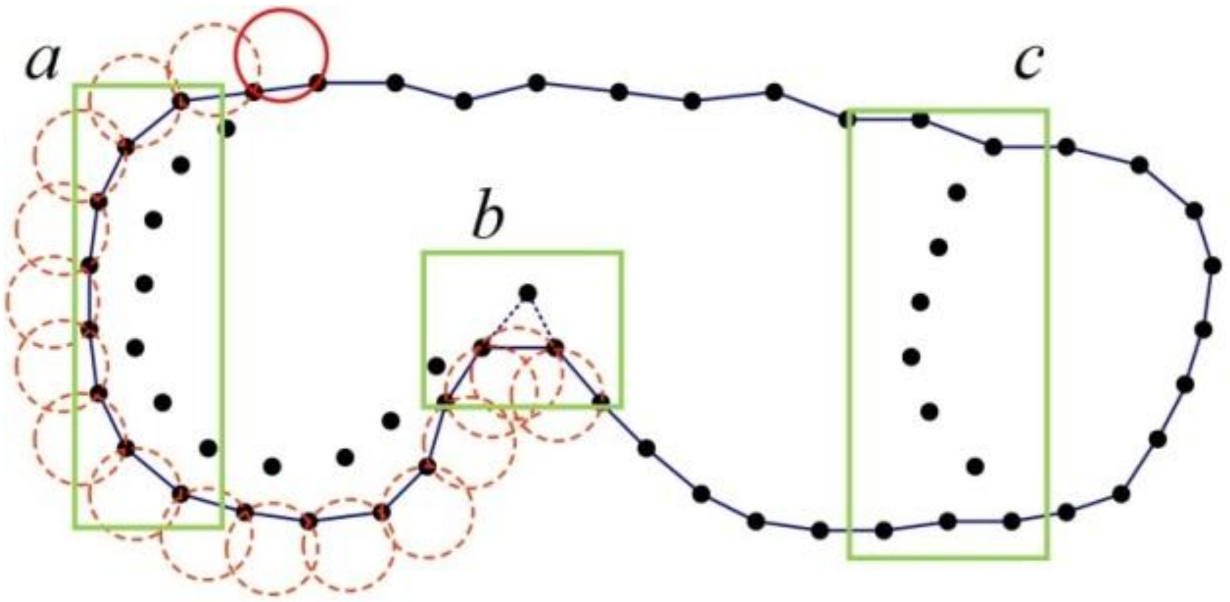


Figure 2.6: Frame of surface construction

Figure 2.6 visualized the framework and benefit of our modified BPA by a 2D example. Given the point cloud of black points, a red "ball" is pivoting/rolling around it to form up the "mesh". From part a we can see that even there are redundant points beneath the surface, BPA will skip them. This explains that when filtering out most of the redundant points, we should be conservative to loosen the restriction, but will not make a big change to the final mesh. Part b shows the limit of BPA when constructing concave edge. A pivoting ball is unable to touch the edge point since the ball will touch the point next to it first. By selecting smaller size of the ball, the surface here can be more accurate but there will never be a sharp edge. Part c highlights the advantage of our modified BPA. Compared to the original BPA, the modified one will not create mesh for inside points. Thus it has the ability to generate cutter swept volume from self-intersecting tool path.

After the surface of the cutter SV is constructed, isolated points which don’t belong to any triangle will be deleted. This mesh is ready to be subtracted from the workpiece for updating.

Chapter 3: Workpiece Update

Although the cutter is machining the raw workpiece in the beginning, throughout the rest of the milling process the cutter is interacting with the in-process workpiece. Physical simulation requires the model of in-process workpiece to identify information of cutting conditions. As well, another basic function of geometric simulation is to visualize the milling process and the final products before real machining. For these considerations, updating workpiece is a very important portion of geometric modeling. In this chapter the details from modeling the workpiece to generating the updated workpiece are explained step by step.

3.1 Model of Workpiece

The first step of workpiece update is to model the workpiece in a suitable way. The desired modeling method should be accurate, easy to update and allow direct access to in-process workpiece. Although analytic expression models the workpiece in the most accurate way, it cannot be update directly. Its historical-based property makes the accessing time become longer after the workpiece is updated many times. Meanwhile some discrete surface modeling methods like vector-based or sliced layer-based representation have difficulties preserving local features for the workpiece. As mentioned in the introduction chapter, our methodology utilizes triangle mesh for the whole geometric modeling process. The workpiece is also modeled as closed manifold triangle mesh like the cutter. When representing the workpiece, triangle mesh is isotropic thus able to preserve details in different directions. After each update the in-process workpiece is an independent mesh that cost the similar memory as the initial workpiece and is directly accessible. Also this makes the modeling process more consistent.

Figure 3.1 is a triangle mesh model of the initial workpiece. When it is a square block, there is no error between mesh model and exact model since the surface of workpiece is always flat. The triangle size of the workpiece model is close to that of the cutter model to ensure the update process to be successful.

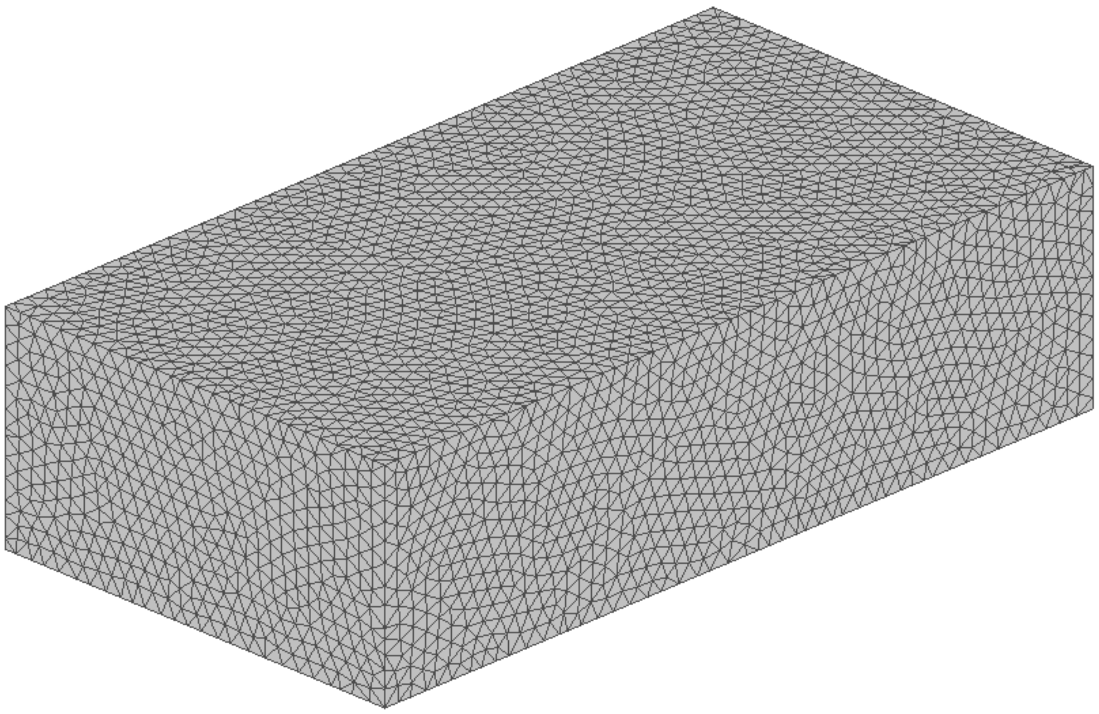


Figure 3.1: Triangle mesh model of initial workpiece

3.2 Boolean Operation between SV and Workpiece

The workpiece update process is essentially a Boolean operation between meshes of the cutter SV and the workpiece. When the cutter moves following the tool path, it removes the material of the workpiece that lies in the cutter SV. From geometric point of view, the updated in-process workpiece is the result of a subtraction Boolean operation.

Figure 3.2 demonstrates several common Boolean operations in geometry. The darken area is the result geometry of $A \cap B$, $A \cup B$ and A-B respectively. Notice the result shape is a combination of inside/outside parts of each geometry according to the rule of each Boolean operation.

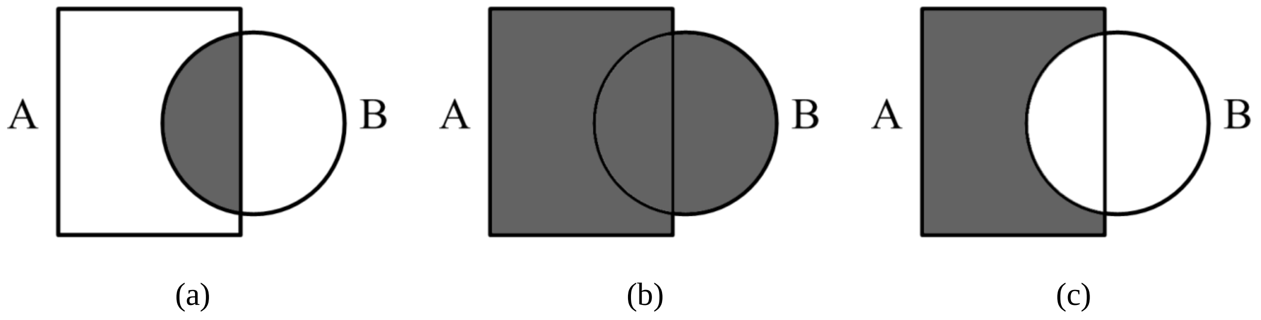


Figure 3.2: Results of Boolean operation (a) $A \cap B$; (b) $A \cup B$; (c) A-B

If we take geometry A as the workpiece and B as the cutter SV, the result of A-B is exactly the updated workpiece. As plotted in Fig. 3.2(c), the result of A-B is part of A that is outside of B. To be specific, the boundary of A-B has two types. The first one is the boundary of A that is outside of B, and the other one is the boundary of B that is inside of A. Therefore, inside/outside partition is the most important step for Boolean operation in geometry and it is performed based on intersection between the two elements. To identify the inside/outside part of each mesh model, intersection between the two meshes needs to be calculated first.

3.2.1 Triangle-to-Triangle Intersection Calculation

The intersection between two meshes is used to divide each mesh into inside and outside parts. In addition, it represents sharp edges of the updated workpiece. These sharp edges are the surface feature that may contribute to physical simulation. Hence, the exact intersection should be calculated to retain the sharp features of updated workpiece.

As triangle mesh, both the models of workpiece and cutter SV are composed of triangles. To calculate the intersection between these two meshes is indeed calculating all the triangle-to-triangle intersections between the two sets of triangles. The algorithm for triangle-to-triangle intersection calculation is adopted from Tomas Möller's work [34]. The output of the intersection between two triangles would be a pair of points.

To test whether the two triangles T_1 and T_2 intersect with each other, we need to check if both triangles intersect with the plane that contains the other triangle. First the plane equation of triangle T_2 is computed,

$$\begin{aligned} &N_2 \cdot X + d_2 = 0, \\ &d_2 = -N_2 \cdot V_0^{\,2} \end{aligned} \tag{3.1}$$

where X is any point on the plane, N_2 is the normal of T_2 and V_0 is the first vertex of T_2.

Then the signed distances from the vertices of T_1 to the plane are computed by inserting the vertices to the plane equation,

$$d_{V_i^1} = N_2 \cdot V_i^1 + d_2, \; i = 0,1,2. \tag{3.2}$$

If the signs of $d_{V_i^1}$, $i = 0,1,2$ are not the same, T_1 is intersecting the plane of T_2. Then the same test is done for T_2. These two tests avoid a lot of calculation for many triangle pairs.

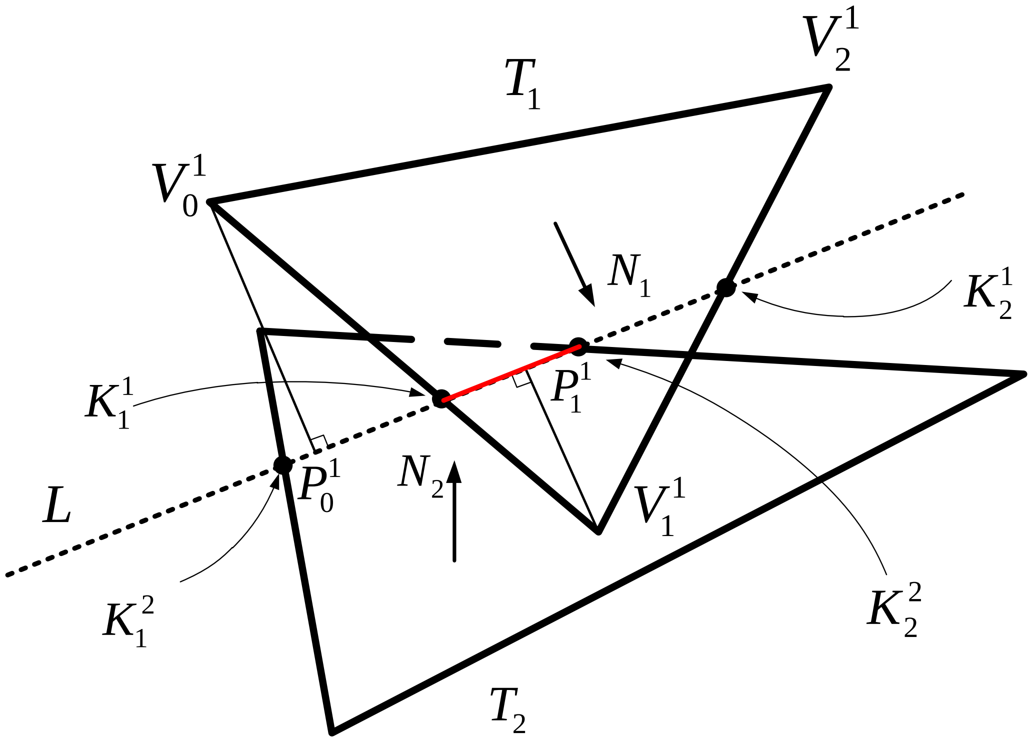


Figure 3.3: A sample of triangle-to-triangle intersection calculation.

Fig. 3.3 is an example that T_1 and T_2 have passed the test and are potentially intersecting. The intersection line of T_1 and T_2 is $L = O + tD$, where $D = N_1 \times N_2$ and O is some point on it. In this example, O is computed as

$$O = \frac{V_0^1 \cdot d_{V_1^1} - V_1^1 \cdot d_{V_0^1}}{d_{V_1^1} - d_{V_0^1}} \tag{3.3}$$

The next step is to calculate the intersection point K_1^1 between the triangle edge $\overline{V_0^1V_1^1}$ and L. $P_{V_i^1}$ are the projection of V_i^1 on L. The scalar value of $P_{V_i^1}$, which is the t in $L = O + tD$ is calculated,

$$p_{V_i^1} = \frac{D \cdot (V_i^1 - O)}{\|D\|^2} \tag{3.4}$$

As noticed, $\Delta V_0^1 P_0^1 K_1^1$ and $\Delta V_1^1 P_1^1 K_1^1$ are similar. According to the similar triangle formula, we are able to compute the coordinate of K_1^1. Similarly, the second intersection point K_2^1 between $\overline{V_0^1V_1^1}$ and L is obtained in the same way. $\overline{K_1^1K_2^1}$ is the intersection segment of T_1. Repeat this procedure on T_2 then we get intersection segment $\overline{K_1^2K_2^2}$. If there is any overlap between these two segments, that is the intersection between T_1 and T_2. In this sample in Fig. 3.3, it is $\overline{K_1^1K_2^2}$.

One tricky situation is that the vertex of one triangle happens to be lying on the other triangle and the intersection cannot be computed successfully. To increase the robustness of this triangle-to-triangle intersection algorithm, when $d_{V_i^j} = 0,\ i \in \{0,1,2\},\ j \in \{1,2\}$, minor disturbance is applied to the position of V_i^j. This will ensure they are either completely intersecting or not intersecting at all, and avoid the ambiguity.

3.2.2 Octree Space Partition

Triangle mesh model is a set of triangles. To calculate the intersection of two triangle meshes, we need to compute the intersection of each pair of triangles that come from each mesh respectively. If the workpiece has N triangles and the SV has M triangles, there will be $N*M$ pairs of triangles. Usually the size of each mesh is too big compared to the actual number of triangles that are intersecting with the other mesh. If we operate $N*M$ triangle-to-triangle intersections, the computation time will be unacceptably long. Therefore, before we compute the intersection, an octree space partitioning algorithm is used to narrow down the size of candidate intersecting triangles.

The concept of space partition is to distribute the triangles of each mesh into many small cells. It has two functions. First, only a portion of cells contain triangles that are potentially involed in the intersection. This will reduce the required calculations of triangle-to-triangle intersection, and speed up the process. In addition, the relative position information of each cell helps us to identify the inside/outside part of the mesh. This is the key information to obtain the correct result of Boolean operation.

The result of the octree space partition is two sets of cells which are labeled *inside*, *ouside* or *boundary* for each mesh. As implied in the name, octree space partition would subdivide a cell by bisecting it in three dimensions and generate eight smaller cells of the same size. The procedure of octree space partition and the following steps are illustrated in a 2D version in Fig. 3.4. Label information is color-coded so please refer to the electronic version.

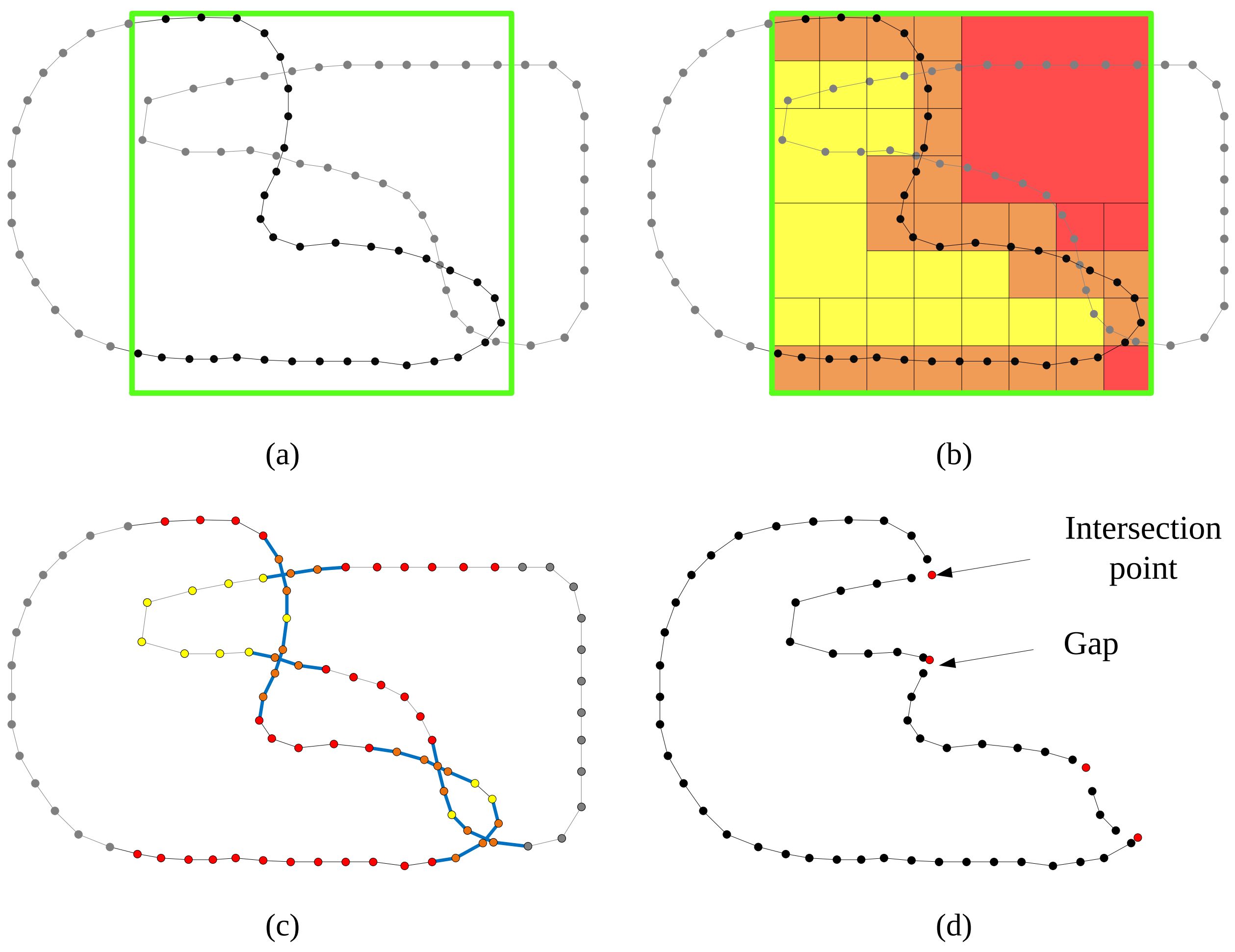


Figure 3.4: Octree space partition and intersection calculation.

Initially the bounding boxes for the two meshes are calculated. There is no need to do space partition for the entire mesh since obviously the remote points are not intersecting with the other mesh. So the smallest cubic box that covers the intersection part of the two bounding boxes, as shown the green square in Fig. 3.4 (a), is assigned as the root of the octree. We use a cubic box so that after subdivision each cell maintains proper shape.

The first step is to partition the space according to vertices of the left mesh. Any cell that contains vertex is subdivided into eight smaller cells. This partition operation repeats until there is no vertex in the cell. The other criteria to terminate the partition is the size of cell. When the length of the cell reaches certain limit, it will not be subdivided anymore. Then the cells that have vertices inside are labeled as *boundary* cells. From these *boundary* cells and vertices inside, other empty cells are identified as *inside* cells or *outside* cells.

The *inside/outside* identification process is illustrated in Fig. 3.5. The center located cell is labeled as *boundary* with vertices inside. Among these vertices, the one closest to the left cell, vertex P_1, is selected and its normal is $\overrightarrow{N_1}$. $\overrightarrow{P_1C_1}$ is the direction vector from P_1 to the center of left cell. The sign of $\overrightarrow{P_1C_1} \cdot \overrightarrow{N_1}$ determines whether the left cell is *inside* or *outside*. For the sample in Fig. 3.5, $\overrightarrow{P_1C_1} \cdot \overrightarrow{N_1} < 0$ so the left cell is *inside*. Similarly, the right cell is outside since $\overrightarrow{P_1C_1} \cdot \overrightarrow{N_1} < 0$. Identify all cells around *boundary* cells in the same way and the result of space partition for the left cell is shown in Fig. 3.4(b).

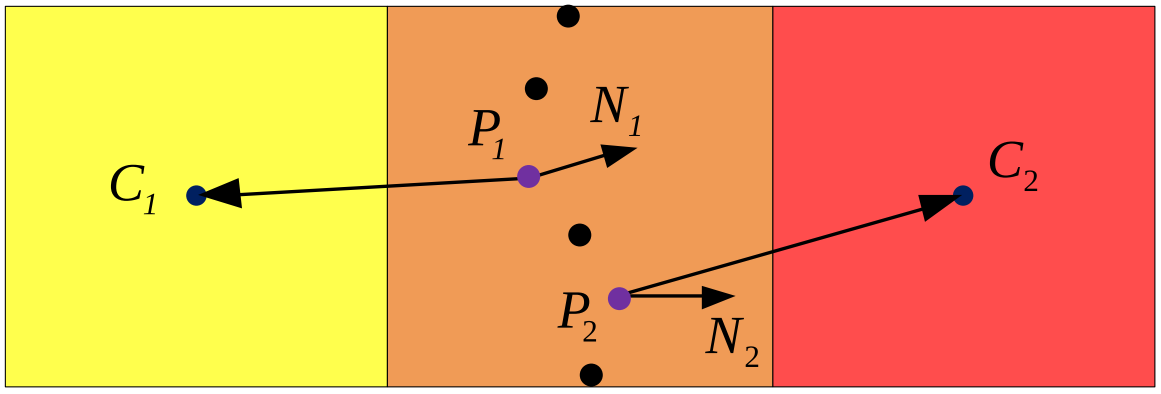


Figure 3.5: Inside/outside cells identification from boundary cell.

With the space partitioned according to the left mesh, the vertices of the right mesh are label *inside*, *outside* or *boundary* based on which cell they belong to in the same way. Now the relative position information of the right mesh is completed and the whole process is repeated on the other mesh. Fig. 3.4(c) is the result that all vertices within the root space from both meshes are labeled.

At this point, triangles that have at least one *boundary* vertex are placed in the candidate pool for triangle-to-triangle calculation. Although the octree space partition algorithm is quite popular, its application of *inside/outside* identification is not 100% robust. In the special case shown in Fig. 3.6, a cell is labeled as *inside* according to mesh A since it contain no any circular point. Though most of the cell is actually inside of mesh A, there is a tiny portion of it that is outside

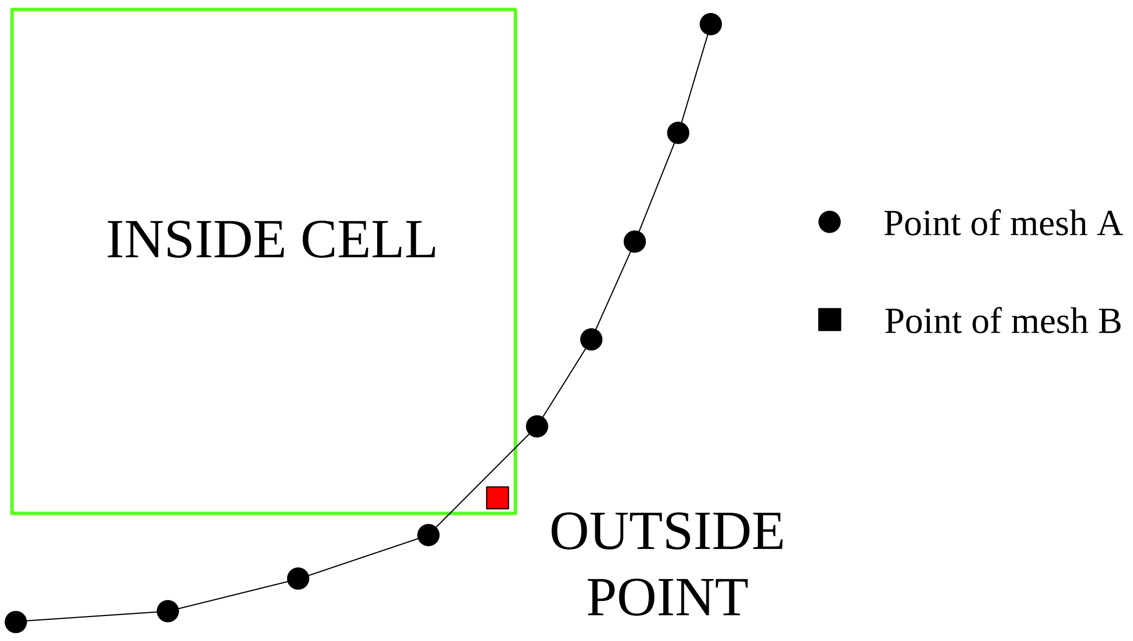


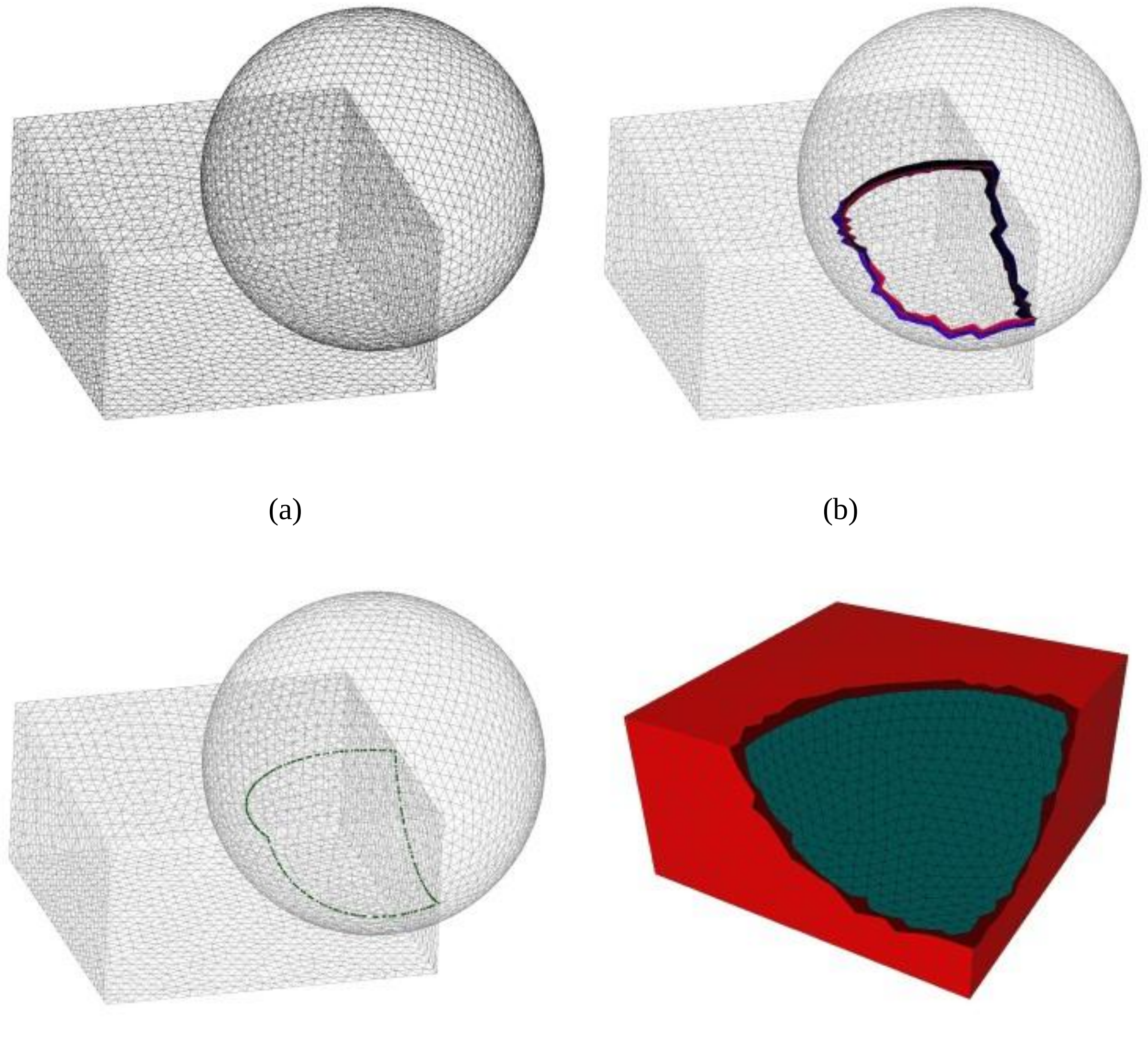
Figure 3.6: Special case when *inside/outside* identification fails.

and a square point from mesh B happens to be there. For this point, if it inherits the label from the cell it belongs to, it would be labeled *inside* while it is outside. This will cause various kinds of problem for some the following operations. To avoid the omissions of some intersecting triangles, neighbors of the candidate triangles are also placed in the pool as a safety factor.

There are two candidate pools for both meshes and these triangles would be used for intersection calculation. Even after this expanding operation, the size of this pool is still way much smaller than the total size of the mesh so the intersection calculation time is greatly reduced. These triangles are shown in Fig. 3.5(c) as the blue segments. Every triangle of one pool may intersect with one or more triangles of the other pool. Now for each pair of triangles that come from the two candidate pools, triangle-to-triangle intersection test and calculation is operated as explained in section 3.2.1. The output would be a set of intersection points and these points will represent the sharp edges of the updated workpiece.

After all the intersection points are generated, candidate triangles which are actually involoved in the intersection will be removed and each mesh is automatically separated into several disconnected pieces. According to the property of the subtraction Boolean operation, part of the workpiece and part of the cutter swept volume will remain for the updated workpiece. The rest which will not contribute to the updated workpiece are removed completely in Fig. 3.5(d).

Fig. 3.7 is a 3D example of the intersection calculation. The blue and red triangles in Fig. 3.7(b) are the potentially intersecting triangles in the candidate pools from each mesh. The green dots in Fig. 3.7(c) are the generated intersection points. Fig. 3.7(d) is the result parts which will be used to construct the updated workpiece. There is a gap between the separated triangles and in the next section, they will be stitched together with the intersection points.


(a) (b)

(c) (d)

Figure 3.7: Intersection calculation on triangle meshes.

3.2.3 Stitching Operation

Our approach for Boolean operation keeps most of the existing triangles instead of recreating them. The majority of the updated workpiece are inherited from the models of the workpiece and cutter SV. It ensures that most of the original information and details not to be lost. On the other hand, this means we are already very close to finishing the updated mesh except the gaps between the remained meshes and the intersecting points.

The resources for the updated workpiece are all the intersecting points and disconnected triangle meshes. In general cases, they will form up several contours, meanwhile around each contour of intersecting points there's a gap between two meshes. In the following content we will demonstrate the stitching operation for single-contour case. And this operation can be easily extended for multiple-contour cases. In this section, the stitching operation is explained step by step.

First step is to sequence the intersection points derived from the triangle-to-triangle intersection calculation. As elaborated in section 3.2.1, the result of intersection between two triangles is a pair of points since the intersection segment has two ends. On the other hand, as both of the models of workpiece and SV are closed meshes, each triangle edge is shared by two triangles. This means that for every intersection point, there exists another point at exactly the same location and these two points belong to different intersection segments. As shown in Fig. 3.8, triangle T_1, T_2 belong to one mesh and T_3 belongs to the other mesh. T_3 intersects with both T_1 and T_2. The intersection points between T_1 and T_3 are the left two points meanwhile the right two points are the intersection points between T_2 and T_3. As noticed, there are overlapped intersection points at the center position since the edge is shared by T_1 and T_2. Due to this property, we are able to remove one of the overlapped intersection points and sequence all the rest points following certain direction around the contour. This sequencing information helps us to fill the gap later.

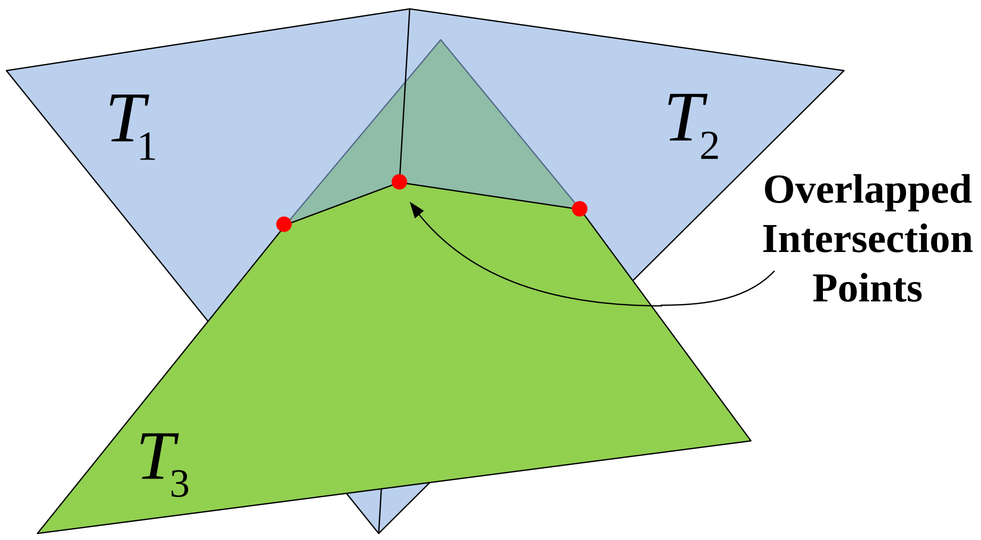


Figure 3.8: Explanation of overlapped intersection points.

Fig. 3.9(a) is a sample of single-contour gap with separated meshes and sequenced intersection points. The two meshes on each side of the gap belong to original workpiece and cutter swept volume accordingly. For every triangle edge on the boundary of the meshes, we search for the closest intersection point and connect them to form up a triangle. Meanwhile this intersection point is labeled as “connected” using different colors in Fig. 3.9(b). Blue points are connected to the blue mesh while green points are connected to the green mesh. If a point is connected to both meshes, it is red.

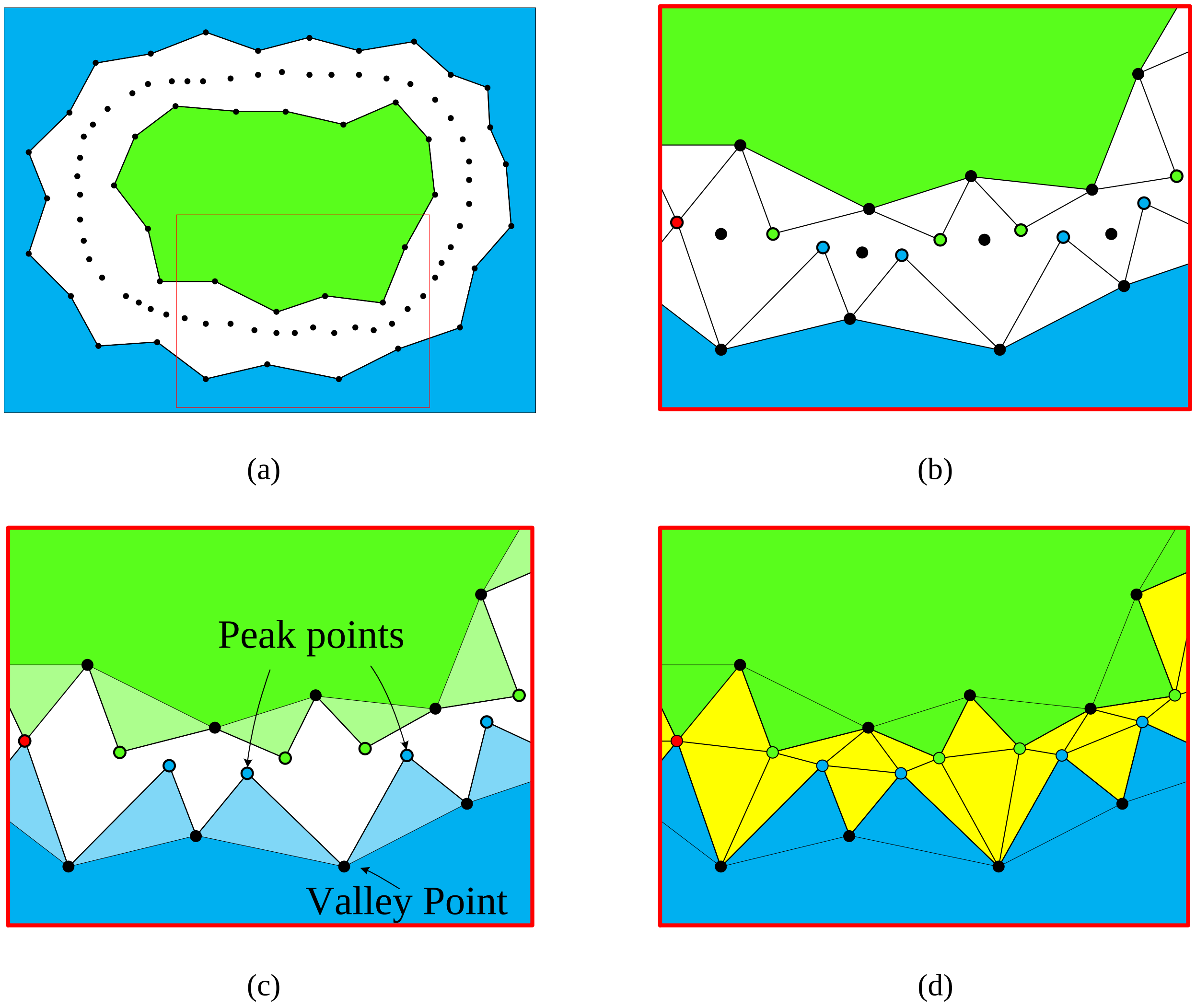


Figure 3.9: Stitching single-contour gap.

There are too many intersection points compared to the number of edges around the contour. We only keep the labeled (colorful) intersection points and the others are removed. Now we have indented meshes that are almost closed in Fig. 3.9(c). With the help of sequencing information, the intersection points are ordered and we are able to get the intersection points between two "peak" points which are connected to the same mesh. For example, between the neighboring blue points, there are green points and these points together with the blue points are connected one by one and all of them are connected to the “valley” point that is connected to both two “peak” points. After repeating this process for both sides of the contour, the contour is stitched with close manifold triangle mesh as in Fig. 3.9(d) and this process can be applied on multi-contour cases.

As mentioned above, these stitching steps are valid only for general cases that around any contour there is only one boundary of workpiece and one boundary of cutter SV on each side of the gap. However, it has to be noticed that in practice the real cases are not always so perfect. It is possible that the removed intersection triangles separate the remained part of SV into two separate pieces. Fig. 3.10(a) shows the special case that around one contour the number of boundaries of each mesh does not match. Some additional work is needed.

The concept of our solution is intuitive. Since our algorithm is already adequate for general cases, what we should do is to transform special case to general case by some fixing. Before the stitching operation, the numbers of boundaries of both meshes are examined to avoid applying general stitching operation on special case to make mistakes. If they do not match, the following fixing operation needs to be done:

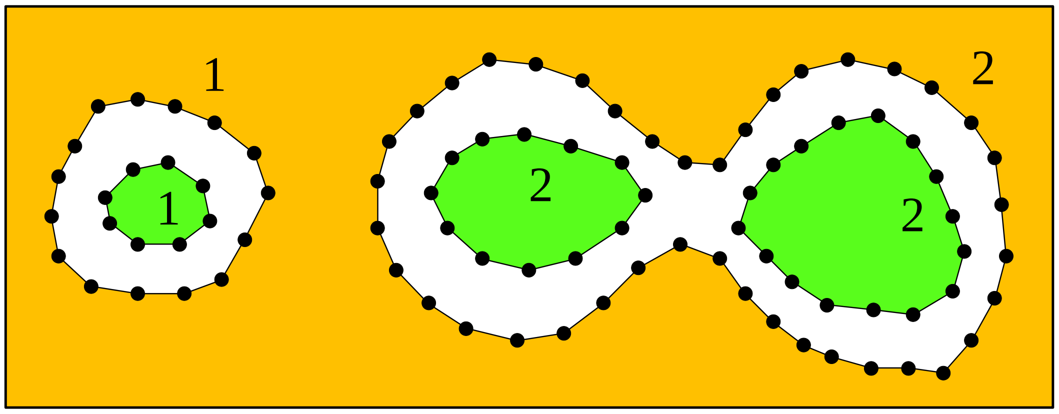
1. Assign each boundary of workpiece a unique code. In Fig. 3.10(a), the left boundary is #1 and the left one is #2.

2. For each boundary of cutter SV, find its nearest boundary of workpiece and copy the assigned code from it. So the left SV boundary is #1 and the right two boundaries are #2.

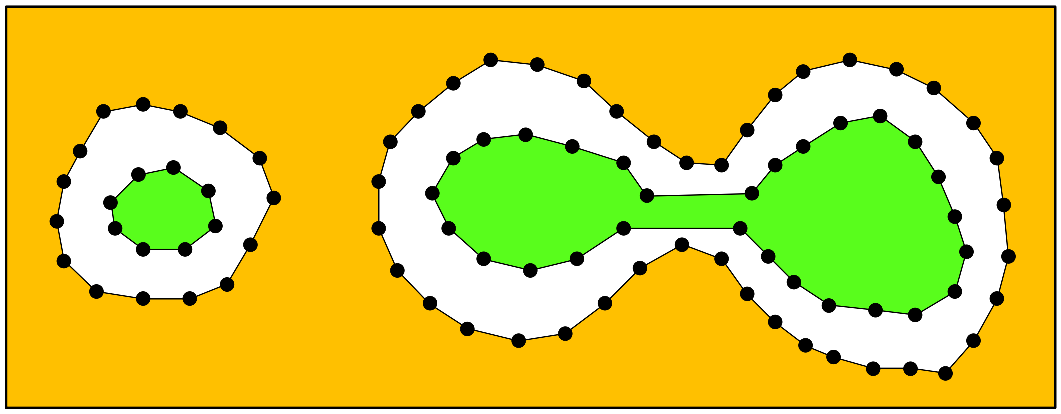
3. Find two boundaries of cutter swept volume that have the same code. Since the numbers of boundaries do not match, there must be at least two boundaries of cutter SV sharing the same code. In the example in Fig. 3.10, the right two boundaries of cutter SV are both #2.

4. Add triangles between the two boundaries to connect them. Thus these two boundaries become one boundary.

5. Re-examine the number of boundaries of both meshes. If they do not match, repeat step 1 – 4 until they are the same.



(a)



(b)

Figure 3.10: Special case for stitching operation: (a) before fixing; (b) after fixing.

This fixing operation can successfully transform the special case into general case and our stitching operation can be applied then. The special case does not appear frequently. It occurs only when the cutter almost touches the surface of the workpiece in the middle of its path. By fixing the special cases, the robustness of our methodology is increased.

Up to now, the Boolean operation between the model of workpiece and model of cutter SV and the workpiece update process is well elaborated. The size of the updated workpiece will not increase and can be directly used for further operations. At the same time the sharp features of in-process workpiece are retained by accurately calculating the intersection points and stitching them with remained parts of each mesh.

Chapter 4: Cutter-Workpiece Engagement Extraction

The main focus and contribution of this thesis is the CWE extraction. We proposed a novel CWE extraction methodology which is different from the existing ones. In Chapter 2 and Chapter 3, the geometry modeling methods and procedures are introduced and they provide enough preparation for us to extract the CWE for general milling process, regardless of the variation of cutter shape and tool path. In this chapter the procedure of the extraction process and its features will be discussed in full details.

In Chapter 3 we talked about the procedure of Boolean operation between the cutter SV and the workpiece model. According to the strategy of CWE extraction in Fig. 1.8, at the same time of updating the workpiece by *subtraction*, another Boolean operation *conjunction* is taken to generate the removal volume (RV). The only difference between these two operations is that the remaining part is inside or outside based on the property of Boolean operation in geometry. The physical meaning of RV is the material that has been removed by the cutter for a time period and will be used for CWE extraction. Compared to the whole workpiece, the size of RV is relatively small and will simplify the following operation.

For simple milling process like 3-axis machining, the RV and cutter can be directly used for CWE extraction. When dealing with complex 5-axis machining, the RV needs to be updated before we calculate the CWE. Because if tool path has self-intersection, the cutter may move through the same point in RV more than once. In other words, part of the intersection area, which is the engagement, is possibly already machined by previous movement and does not contribute to current CWE. To avoid this kind of error, it requires RV to be updated before we extract the intersection between cutter and RV.

In the following sections of this chapter, the RV update process is introduced. Then the intersection between cutter and updated RV is calculated to extract the engagement area represented by intersection points. At last, the intersection points are mapped to the engagement diagram which has more physical meaning and can be used for physical simulation.

4.1 Removal Volume Update

Removal volume update is to transfer the RV to its status when the cutter has moved to certain position. The update process is shown in Fig. 4.1. Suppose the RV is generated by cutter movement from CL_A to CL_B in Fig. 4.1(b). To extract the CWE at CL_C between CL_A and CL_B, another SV is generated from CL_A to CL_C in Fig. 4.1(c), which is denoted as cutter SV #2 in Fig. 1.8. By doing subtraction Boolean operation between this SV #2 and the RV, the RV is updated to the moment when the cutter is at CL_C. This update process eliminates the possible interference from previous movement and makes sure that the intersection or common area between cutter and updated RV is exactly the part of cutter that is cutting material, as known as the CWE area.

The dash boundary in Fig. 4.1(c) is the boundary of SV #2. Since RV is derived from SV #2 and SV #1 and SV #2 share the same tool path, part of their surfaces would overlap which makes the Boolean operation between them impossible. Both the *inside/outside* test and intersection calculation are too tricky to handle. To overcome this problem, the SV #2 needs to be expanded a little bit in advance.

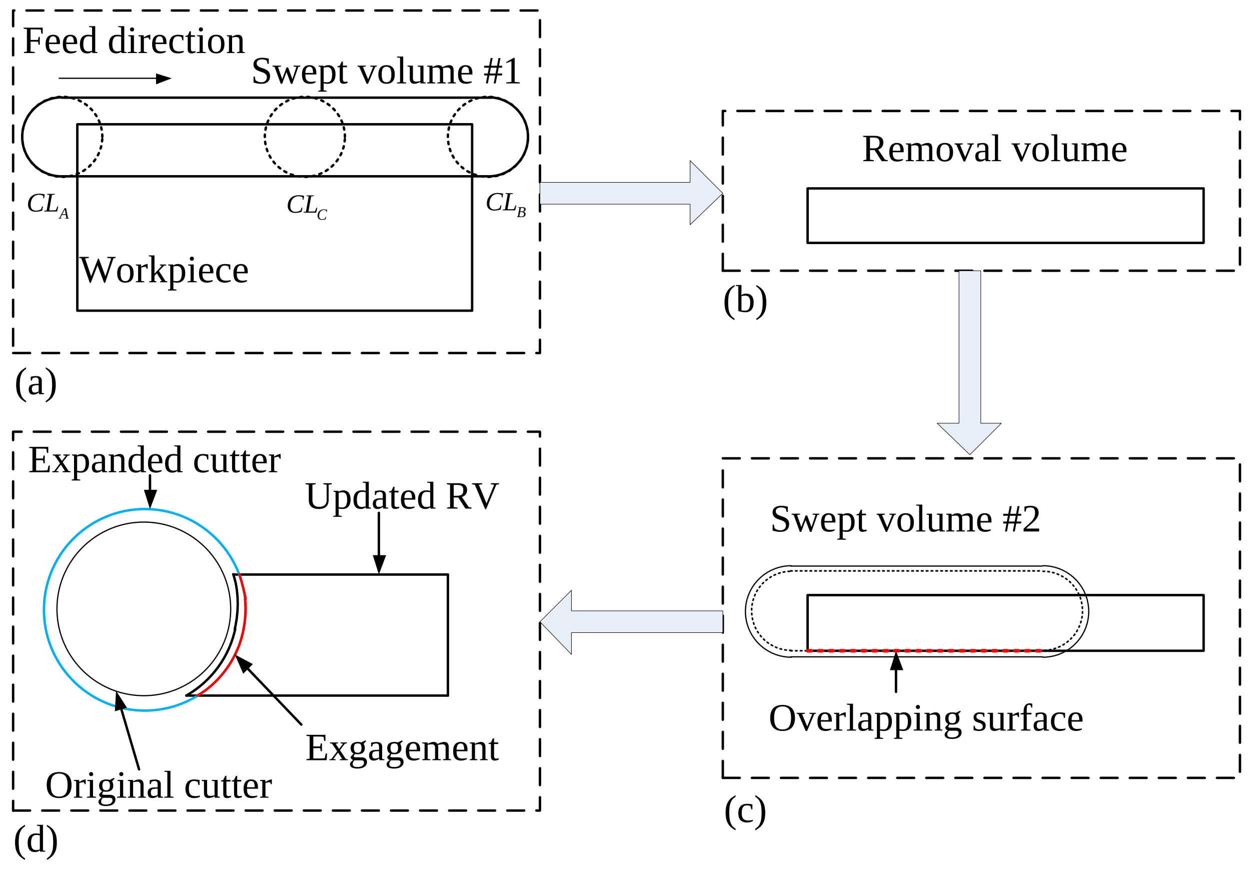


Figure 4.1: Flow chart of CWE calculation in 2D.

The solid boundary in Fig. 4.1(c) is the expanded SV #2. After this operation, the surfaces of each mesh are clearly isolated and Boolean operation becomes available. Then subtraction operation is taken between the two meshes to generate the updated RV in Fig. 4.1(d).

4.2 Intersection Area Calculation.

As shown in Fig. 4.1(d), the cutter and the updated RV simulate the exact moment when the cutter moves to CL_C. The contact area between these two mesh is what we define as the CWE area. As in implementation, contact area is hard to get and intersection area is calculated instead.

The expansion on SV #2 makes the Boolean operation possible, however it subtracts more than desired. One consequence of the expansion is that there is a gap between the cutter at CL_C and the updated RV. To calculate their intersection successfully, another expansion on the cutter is required. This expansion should be larger than the previous one so the gap will be. After the expansion, intersection calculation is the same as we did for workpiece update using triangle-to-triangle calculation and octree space partition in section 3.2.1 & 3.2.2. The output is a set of points representing the boundary of the intersection area.

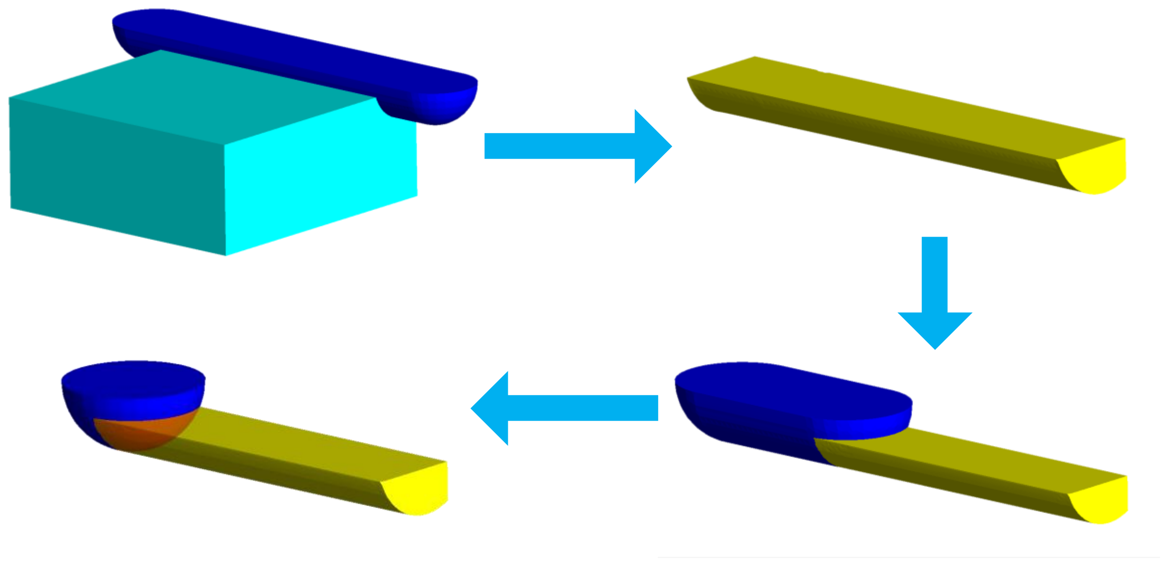


Figure 4.2: Flow chart of CWE calculation in 3D

A 3D example is shown in Fig. 4.2. It simulates the same process in Fig. 4.1. The red area in the last figure is the output engagement area and its boundary is a set of points which will be mapped to the engagement diagram.

4.3 Engagement Diagram Generation

From the intersection calculation between cutter and updated RV, the output is merely some points in the space with 3D coordinates that are not usable for physical. They have to be transformed to certain data type to be directly used in the simulation. The typical data type of CWE is a pair of angle and height. Every intersection point from section 4.2 represents the entry and exit angle of cutting at that height. This angle will be used for physical simulation for that particular slice of cutter as explained in the introduction. All the points together will form up an engagement diagram.

4.3.1 Coordinate Transformation

The mapping process which converts the point coordinate (x, y, z) into data type (ϕ, h) is operated in the local coordinate system of the cutter. As the cutter moves, this coordinate system keeps changing and needs to be defined for each cutter position. This requires much work and is not very convenient. In our methodology, we propose the idea that by using the CL data, the intersection points are reversely transferred to the surface of the original cutter. Meanwhile the feed direction, which is critical in the mapping process, is re-calculated. Then the engagement data (ϕ, h) could be computed just in the simple original cutter coordinate system with origin at (0,0,0). The depth of cut h is just the z value of the point and the angle value ϕ is calculated in the XY-plane.

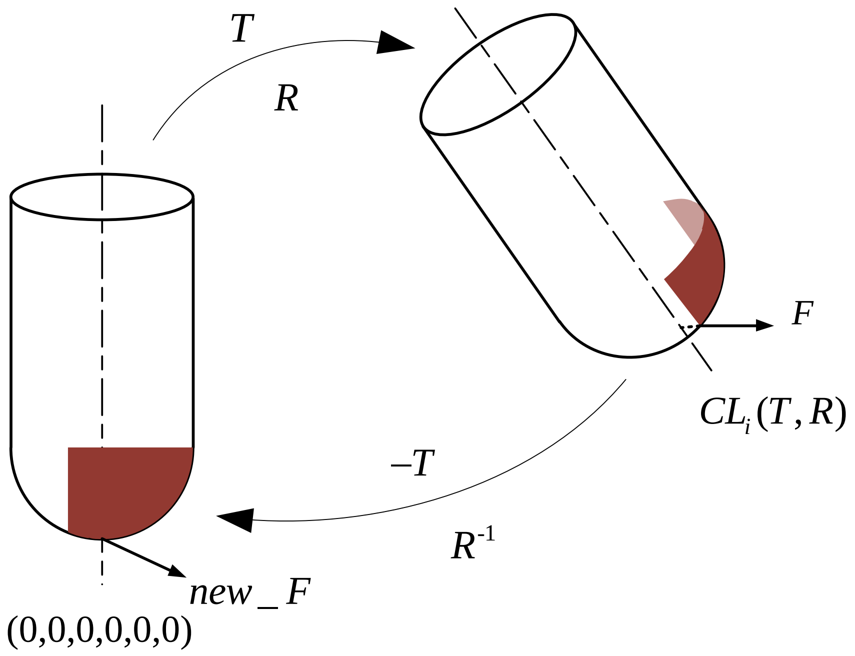


Figure 4.3: Coordinate transformation.

Fig. 4.3 shows the concept of reverse coordinate transfer. The upper arrow indicates that the model of cutter is moved to certain cutter location by applying the translational vector T and rotational matrix R, which are derived from the CL data. Since the intersection points are on the surface of the cutter, they can be moved back to the surface of the original cutter by reversely applying the translational vector $-T$ and rotational matrix R^{-1}. Given the $new\_Pivot(x, y, z)$, for every intersection point $p(x, y, z)$, its corresponding position on the surface of the original cutter is calculated

$$
\begin{aligned}
&new\_p(x,y,z) = new\_Pivot(x,y,z) - T + R^{-1} \cdot L \\
&L = new\_p(x,\mathrm{y},\mathrm{z}) - new\_Pivot(x,y,z).
\end{aligned}
\tag{4.1}
$$

As shown in Fig. 4.3, the engagement area is transferred from the actual cutter to the original cutter model. Also the new feed direction is obtained as

$$new\_F = R^{-1} \cdot F \tag{4.2}$$

This direction will be used in the mapping process and is the reference to define the entry and exit angle.

4.3.2 Mapping to Engagement Diagram

After we transferred all the intersection points to the original cutter coordinate system, the mapping process becomes much easier. The mapping process is in Fig. 4.4. For each point, its new z value will be its axial distance to the cutter tip, as called *Depth of Cut* in the Engagement Diagram in the right of Fig. 4.4. Next we connect the point and its projector on Z-axis. As in the physical model, the feed direction is defined as 90 degree. So a new direction which is 90 degree counterclockwise from $\vec{F}$ is introduced. The angle from this new direction to this line connecting the intersection point and its projection will be the value *Angle* in the Engagement Diagram. Repeat these procedures for all points and the whole diagram for the engagement map is obtained for physical simulation.

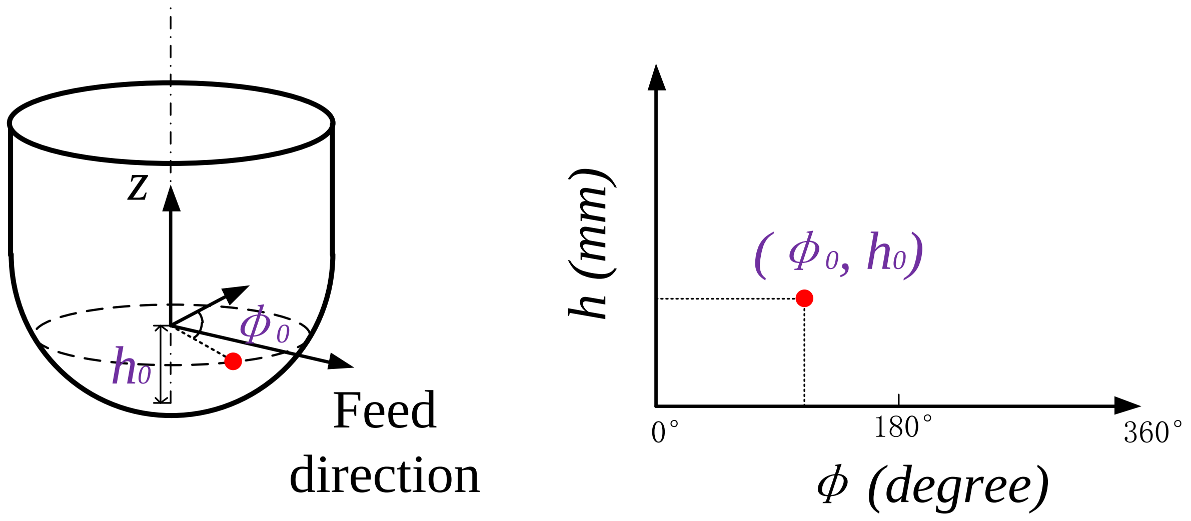


Figure 4.4: Mapping from coordinate to engagement diagram.

For simplicity consideration, for all intersection points the angle is calculated using the feed direction which is derived from translational movement only. In five-axis machining, rotational movement is also applied and thus making the feed direction vary along the cutter. In other words, the actual angle should be calculated individually for each intersection point. In our study the Engagement Diagram may require some additional work for physical simulation.

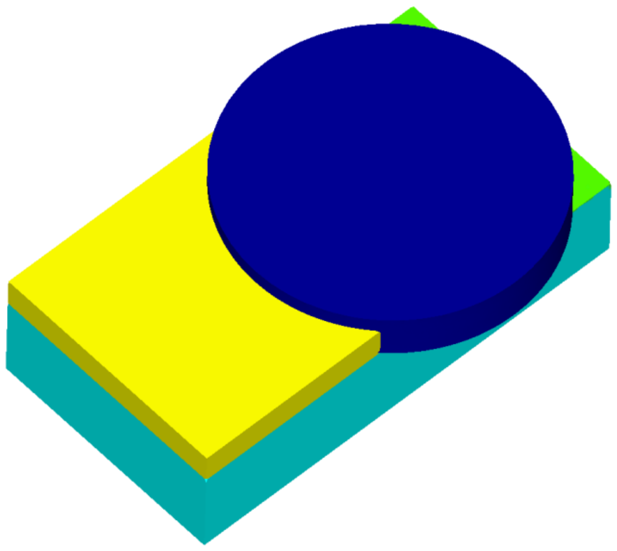
Chapter 5: Case Studies and Discussion

To demonstrate the utility and performance of our CWE extraction methodology for general milling processes, six case are presented in this chapter. They are classified into three categories according to their complexity. In each case study, two or three types of CWE diagram are plotted, including high performance result, normal performance result and the analytic result. The high performance result is derived from models with smaller triangle size while the regular performance one utilizes larger triangle size. The analytic result is calculated analytically for simple cases only and plays the role as reference. All the configurations for each case study, including cutter diameter, triangle number, triangle edge length and execution time can be found in the Appendix. This chapter also visualized the importance and necessity of RV update and analyzed the error of engagement diagram.

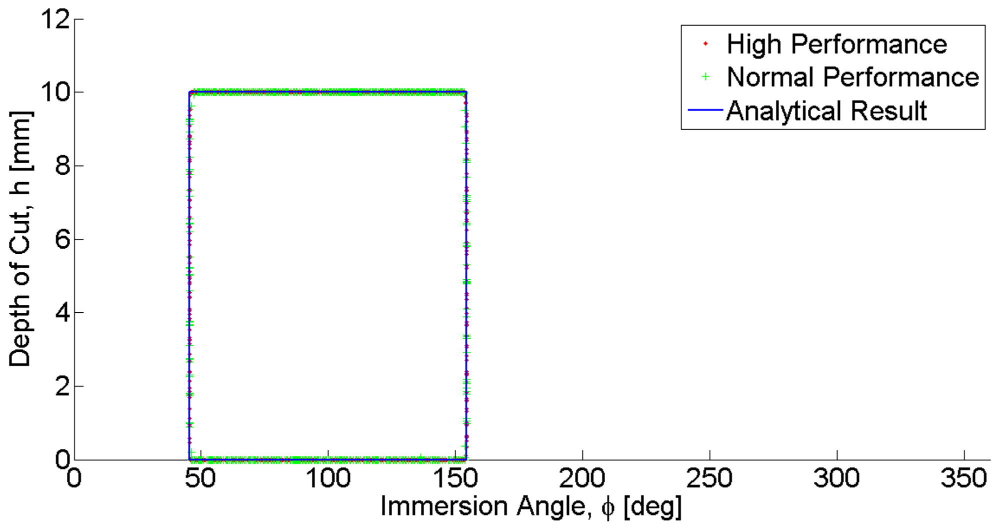
5.1.1 Case Studies with Results

The first category contains three simple case studies in which there are only horizontal movements. Fig. 5.1-5.3 showed the face milling, the profile milling and the plane surface milling. Fig. 5.1(a), Fig. 5.2(a) and Fig. 5.3(a) simulated the milling process and explained where the CWE is extracted. The light blue part is the updated workpiece with green surface on it, which represents newly machined surface. The yellow part is the updated removal volume and the dark blue model is the cutter.

As shown in Fig. 5.1(b), Fig. 5.2(b) and Fig. 5.3(b), our results are quite close to the exact analytic results. The error comes from the expansion of cutter when cutter and updated RV are intersected to generated intersection points as mentions in section 4.2. The expansion is the major reason for errors to occur and is inevitable due to model's nature instincts. The expansion makes the intersection points deviated from the cutter surface, thus the mapping result is inaccurate. The influence of the expansion will be analyzed at the end of this chapter.

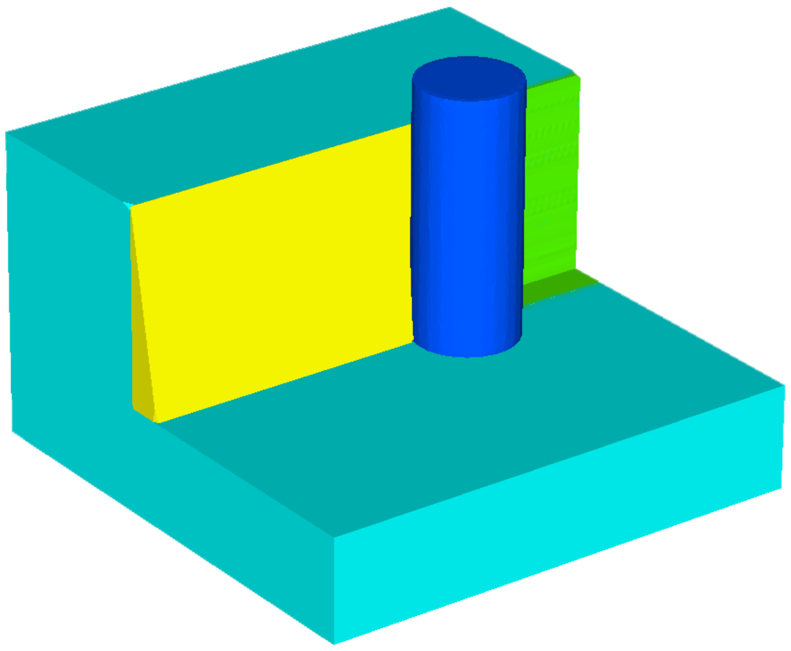


(a)

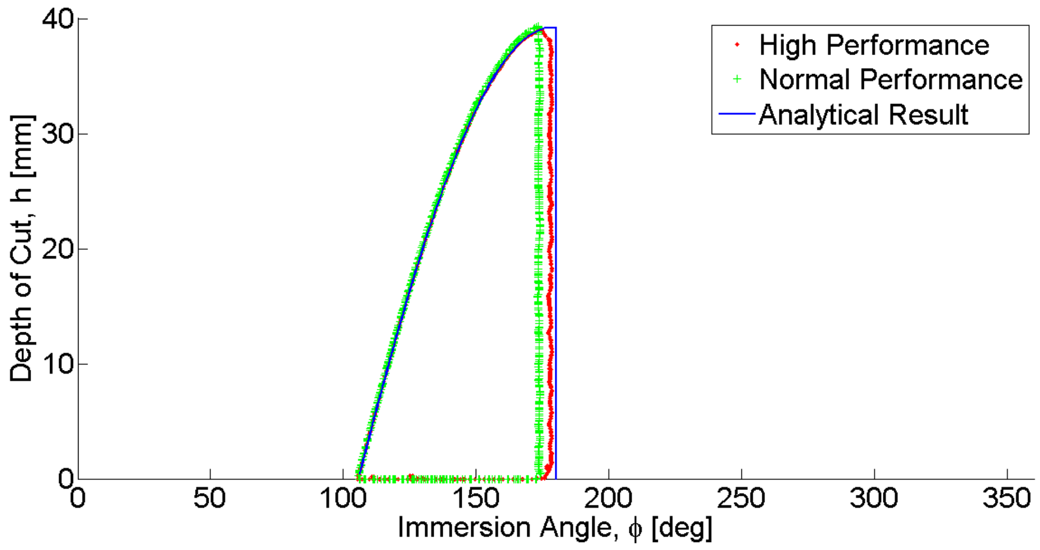


(b)

Figure 5.1: Case #1: face milling.

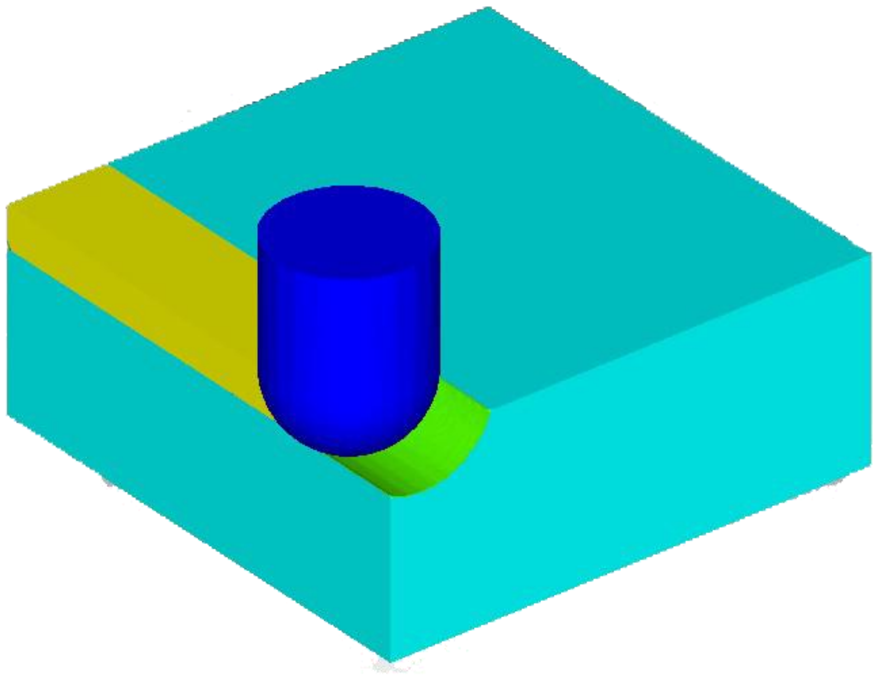


(a)

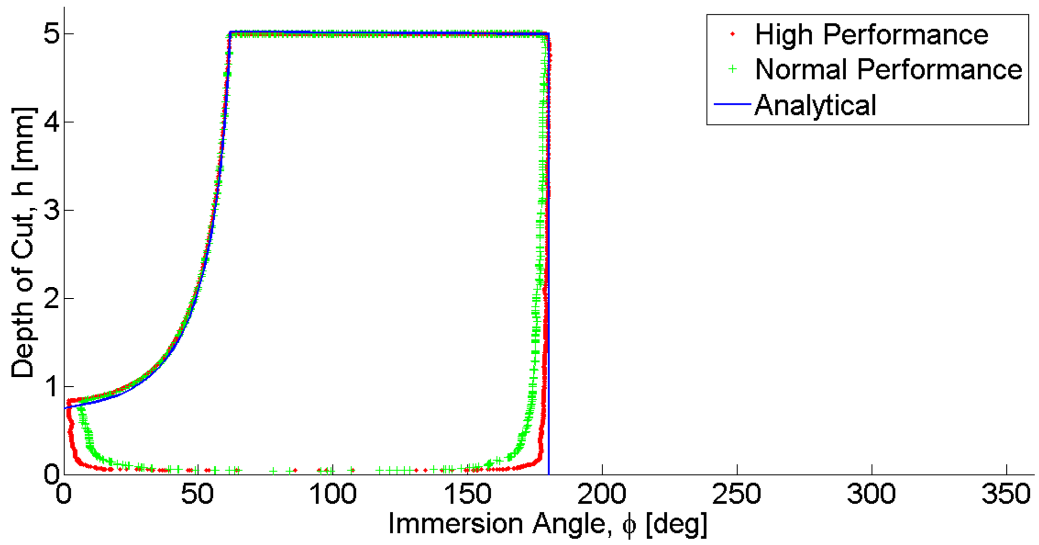


(b)

Figure 5.2: Case #2: profile milling.



(a)



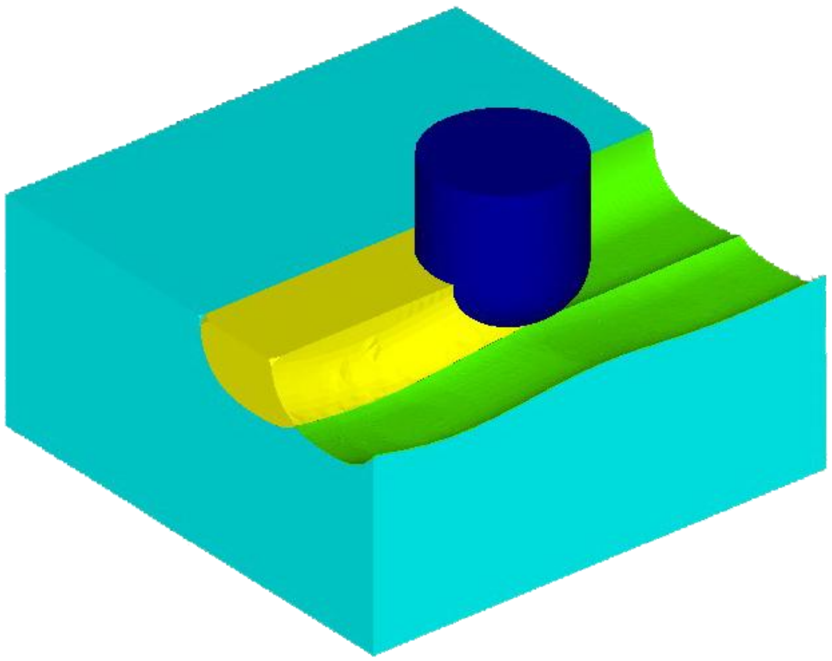
(b)

Figure 5.3: Case #3: Horizontal ball-end milling.

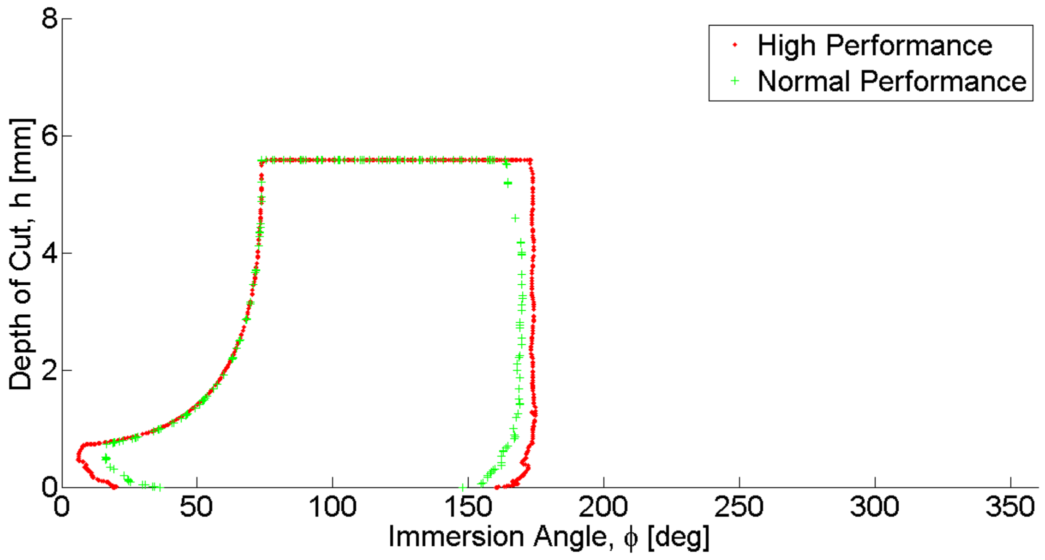
There are two expansions in our methodology and theoretically they can be infinite small. However, as the surface of SV is generated from point cloud, it is only the approximate model of real SV and its roughness prevents the RV update to be successful when the expansion is too small. Since the triangle size of cutter affects the roughness of SV and RV, it also determines the level of expansion. The expansion amount is defined as a portion of average edge length and the minimum ranges from 1% to 10%. The results of first category cases demonstrated that our methodology is able to generate accurate CWE area, especially when the angle is far from 0 ° and 180 °. The performance can be improved by utilizing models with smaller triangle size, meanwhile the execution time will increase greatly.

The second category contains two case studies of free form sculpture surface milling. They are more complex than the first category. Case #4 uses fillet-end mill to machine a sculptural surface and case #5 contains 5-axis tool path. They are simulated in Fig. 5.4 and Fig. 5.5 with engagement diagrams. At this stage the analytic result is almost impossible to generate so only high performance and normal performance results are plotted. It is convictive to believe that these results are also very accurate.

The last category has the most complicated and challenging case, in which the cutter is a taper-ball-end mill and the multi-axis tool path has self-intersection. In our last case study, the cutter is machining the side surface within a cavity. Fig. 5.6(a) the left shows the workpiece and cutter swept volume. Fig. 6(b) shows the updated removal volume and the cutter and Fig. 6(c) shows its results.

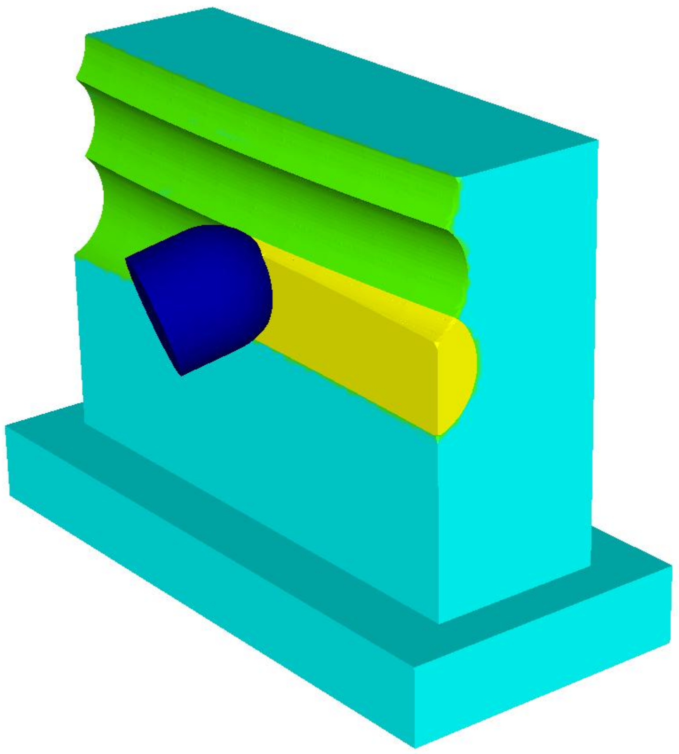


(a)

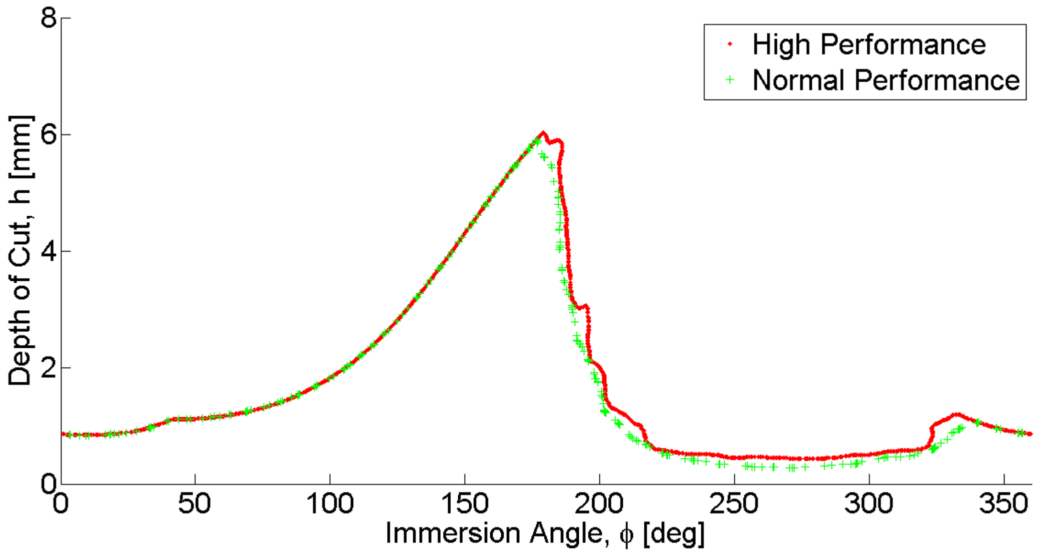


(b)

Figure 5.4: Case #4: fillet-end 3-axis sculptural surface milling.

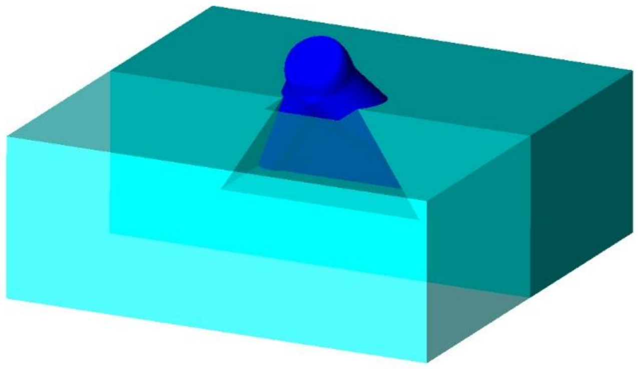


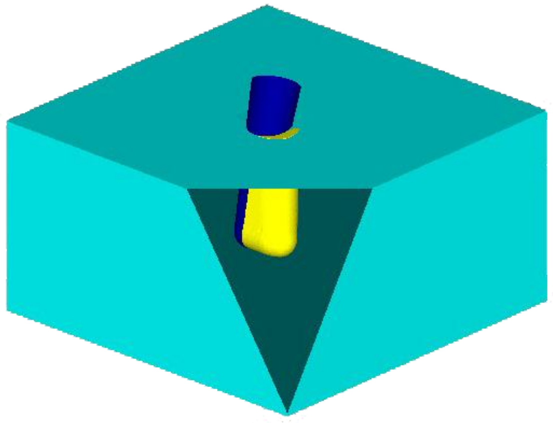
(a)

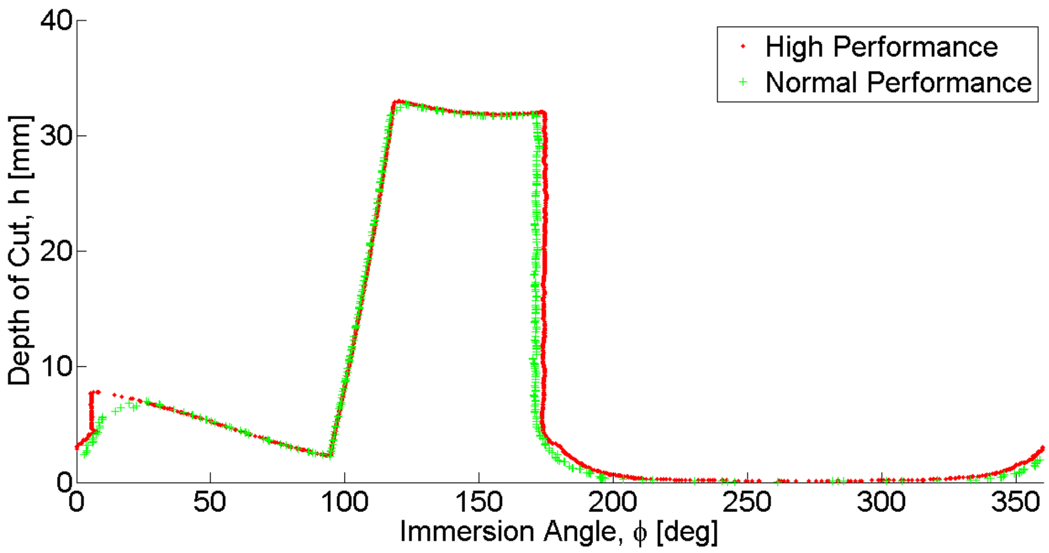


(b)

Figure 5.5: Case #5: ball-end 5-axis blade surface milling.


(a)


(b)



(c)

Figure 5.6: Case #6: 5-axis Taper-ball-end milling with self-intersection

Fig. 5.7 illustrated the necessity of RV update to successfully extract correct CWE when tool path has self-intersection. In this example where the RV is not updated, when we calculate the intersection between the dark blue cutter and the RV, the cutter would be considered as cutting materials that was already removed by the light blue cutter. In addition to the real CWE area, at least part of the fake CWE area will be added to the result CWE area. The only solution is to update the RV first, then the fake CWE area will be eliminated before CWE extraction and only real CWE area would remain. This guarantees that our CWE result is correct at any circumstance.

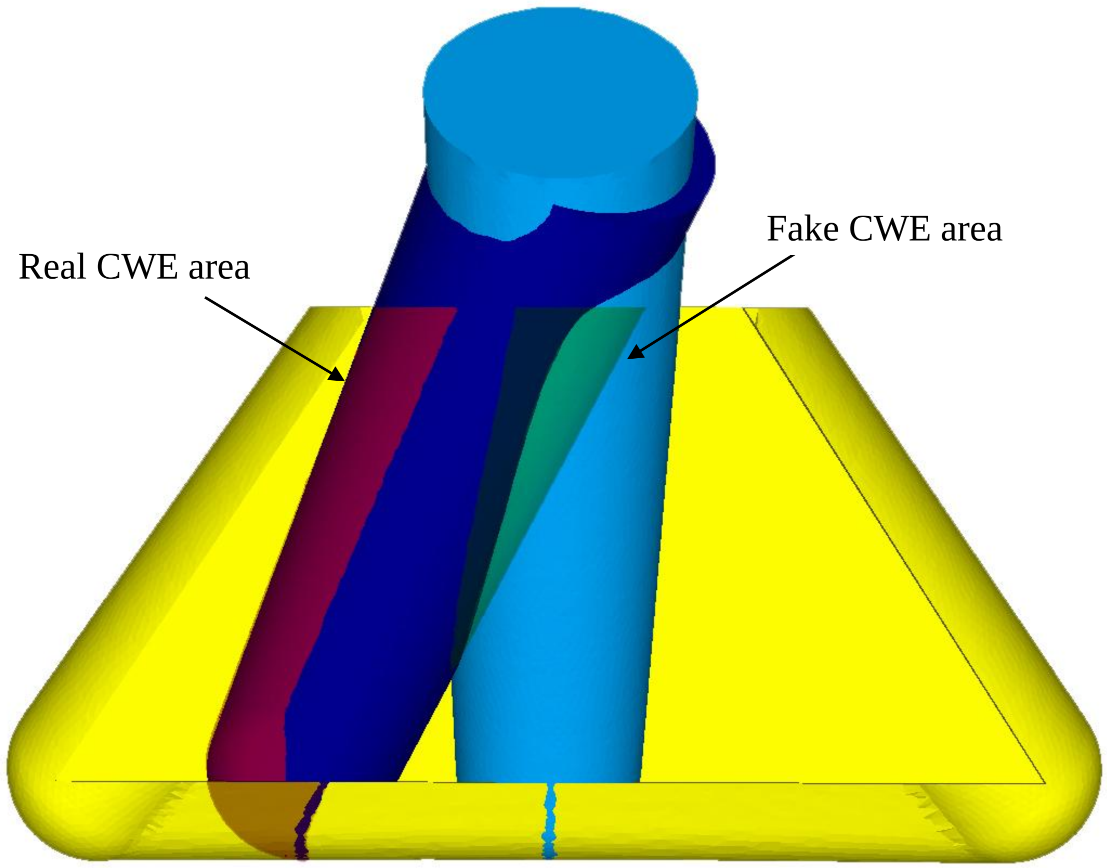


Figure 5.7: False CWE extraction without updating removal.

5.1.2 Error Analysis

As notice, the error mainly occurs around 0 ° and 180 °. When the angle is far from these two extremes, the error is negligible. To explore the influence of the expansion on the error of angle, especially the difference between different engagement angles, an error analysis is taken on simple situation where the boundary of RV and the tool path is a straight line.

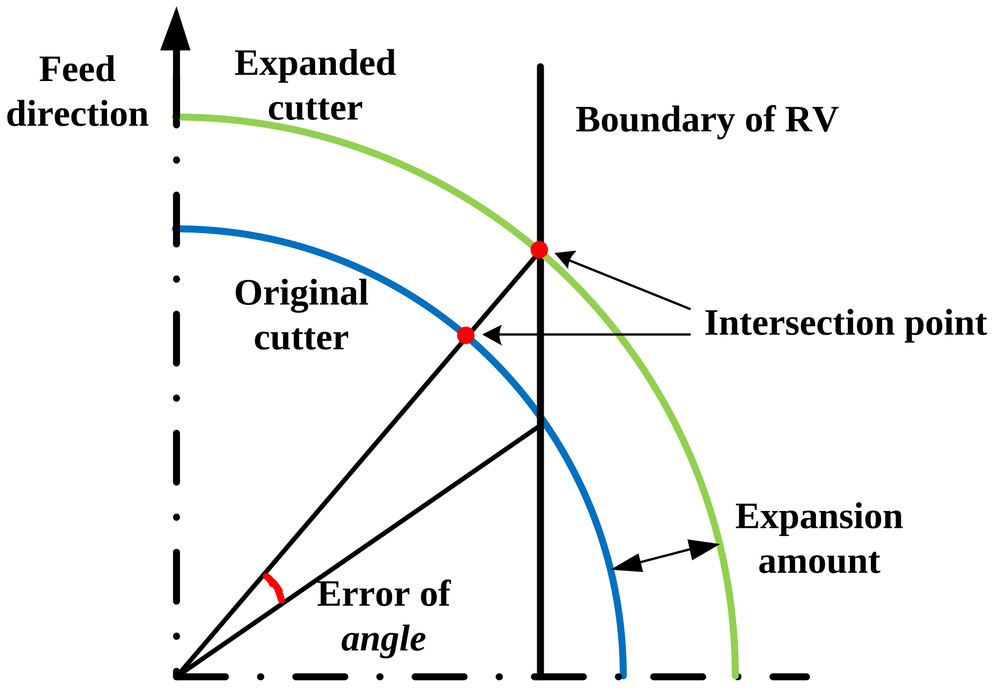


Figure 5.8: Error analysis interpretation.

In Fig. 5.8 a part of the cutter and a straight RV boundary are plotted. The blue circle is the true cutter and the green one is the expanded cutter. As the two cutters will intersection with RV at two intersection points which lead to different engagement angles, the error is introduced. Fig. 5.9 plots the error versus various engagement angle (with true cutter) from 0 ° to 180 ° and expansion amount from 0.1% to 2% of the radius of the cutter. The trend of the error is obvious in this figure. For the same amount of expansion, the error increases greatly near 0 ° and 180 ° while for other angles the error is very small. For the same engagement angle, the error increases with expansion with no doubt.

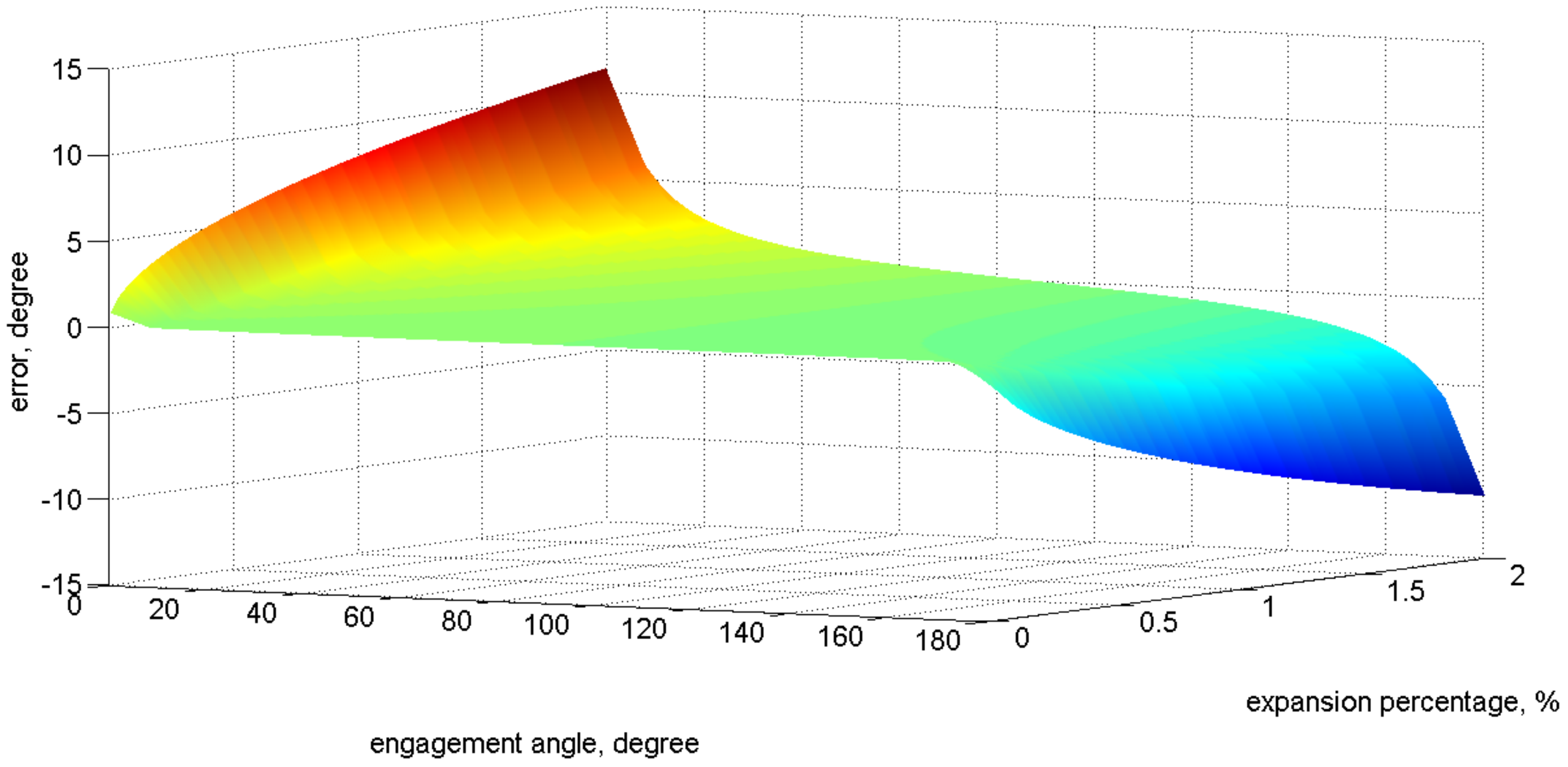


Figure 5.9: Error versus angle and expansion percentage.

Chapter 6: Conclusion

In modern manufacturing industry, a single complex part could be costly and required lots of time to machine it. For better accuracy and performance, people tend to simulate the cutting conditions in the virtual environment to predict and optimize the machining process. Cutter-workpiece engagement is key parameter for physical simulation and can only be obtained through accurate and robust geometric modeling. This thesis presents a novel geometry modeling methodology for cutter-workpiece engagement extraction.

Our methodology is based on triangle mesh models throughout three major procedures including cutter swept volume generation, workpiece update and CWE extraction. The cutter SV is used to simulate the space that the cutter possessed during machining. It is used to remove the material from the workpiece. Ball-pivoting algorithm is adopted in the generation of the triangle mesh model of SV. The workpiece is updated after each tool path segment to create the in-process workpiece for cutting condition simulation. Boolean operation between meshes is explained in details. Octree space partition algorithm is applied to speed up the triangle-to-triangle intersection calculations by reducing the number of triangles. In conclusion, our method is able to simulate the geometry information of general milling process in virtual environment meanwhile generate fairly accurate cutter-workpiece engagement for the physical simulation.

Several case studies revealed the advantage of our methodology that it can be applied for general milling process simulation. The main contribution of our research is that it breaks the limitation of cutter geometry and tool path complexity for CWE extraction by adopting discrete triangle mesh modeling method. Accompanied with the advantage is the sacrifice of execution time. Currently BPA for SV generation and triangle-to-triangle intersection calculation in Boolean operation cost a majority of the execution time. Further optimization is required to shorten the execution time so that this methodology is applicable in practice.

Bibliography

[1] Altintas, Y., 2012, *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*, Cambridge university press.

[2] Yip-Hoi, D., and Huang, X., 2006, "Cutter/Workpiece Engagement Feature Extraction from Solid Models for End Milling," Journal of Manufacturing Science and Engineering, **128**, pp. 249-260.

[3] El-Mounayri, H., Elbestawi, M. A., Spence, A. D., and Bedi, S., 1997, "General Geometric Modelling Approach for Machining Process Simulation," International Journal of Advanced Manufacturing, **13**, pp. 237–247.

[4] Imani, B. M., Sadeghi, M. H., and Elbestawi, M. A., 1998. "An Improved Process Simulation System for Ball-End Milling of Sculptured Surfaces," International Journal of Machine Tools and Manufacture, **38**, pp. 1089–1107.

[5] Bailey, T., Elbestawi, M. A., El-Wardany, T. I., and Fitzpatrick, P., 2002, "Generic Simulation Approach for Multi-Axis Machining, Part 1: Modeling Methodology," Journal of Manufacturing Science and Engineering, **124**, pp. 624-633.

[6] Sadeghi, M. H., Haghighat, H., and Elbestawi, M. A., 2003, "A Solid Modeler Based Ball-End Milling Process Simulation," The International Journal of Advanced Manufacturing Technology, **22**, pp. 775–785.

[7] Fussell, B. K., Jerard, R. B., and Hemmett, J. G., 2003, "Modeling of Cutting Geometry and Forces for 5-Axis Sculptured Surface Machining," Computer-Aided Design, **35**, pp. 333–346.

[8] Yao, Z., and Joneja, A., 2010, "Computing Cutter Engagement Values in Milling Tessellated Free-Form Surfaces," Journal of Computing and Information Science in Engineering, **10**, 041005.

[9] Larue, A., and Altintas, Y., 2005, "Simulation of Flank Milling Processes," International Journal of Machine Tools and Manufacture, **45**, pp. 549–559.

[10] Aras, E., and Yip-Hoi, D., 2008, "Geometric Modeling of Cutter/Workpiece Engagements in Three-Axis Milling Using Polyhedral Representations," Journal of Computing and Information Science in Engineering, **8**, 031007.

[11] Ferry, W., and Yip-Hoi, D., 2008, "Cutter-Workpiece Engagement Calculations by Parallel Slicing for Five-Axis Flank Milling of Jet Engine Impellers," Journal of Manufacturing Science and Engineering, **130**, 051011.

[12] Wang, W. P., and Wang, K. K., 1986, "Geometric Modeling for Swept Volume of Moving Solids," Computer Graphics and Applications, **6**, pp. 8-17.

[13] Chiou, C-J., and Lee Y-S., 1999, "A Shape-Generating Approach for Multi-axis Machining *G*-Buffer Models," Computer-Aided Design, **31**, pp. 761-776.

[14] Chiou, C-J., and Lee Y-S., 2002, "Swept Surface Determination for Five-Axis Numerical Control Machining," International Journal of Machine Tools and Manufacture, **42**, pp. 1497-1507.

[15] Du, S., 2004, *Simulation and Tool Path Optimization for the Hexapod Milling Machine*, Vulkan-Verlag GmbH.

[16] Du, S., Surmann, T., Webber, O., and Weinert, K., 2005, "Formulating Swept Profiles for Five-Axis Tool Motions," International Journal of Machine Tools and Manufacture, **45**, pp. 849-861.

[17] Sheltami, K., Bedi, S., and Ismail, F., 1998, "Swept Volumes of Toroidal Cutters Using Generating Curves," International Journal of Machine Tools and Manufacture, **38**, pp. 855-870.

[18] Roth, D., Bedi, S., and Ismail, F., 1999, "Generation of Swept Volumes of Toroidal Endmills in Five-Axis Motion Using Space Curves," In Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications, pp. 306-311.

[19] Roth, D., Bedi, S., Ismail, F., and Mann, S., 2001, "Surface Swept by a Toroidal Cutter During 5-Axis Machining," Computer-Aided Design, **33**, pp. 57-63.

[20] Mann, S., and Bedi, S., 2002, "Generalization of the Imprint Method to General Surfaces of Revolution for NC Machining," Computer-Aided Design, **34**, pp. 373-378.

[21] Pottmann, H., Peternell, M., 2000, "Envelopes—computational theory and applications," Proceedings of Spring Conference on Computer Graphics and its Applications, Budmerice, Slovakia, pp. 3–23.

[22] Aras, E., 2009, "Generating Cutter Swept Envelopes in Five-Axis Milling by Two-Parameter Families of Spheres," Computer-Aided Design, **41**, pp. 95-105.

[23] Gong, H., and Wang, N., 2009, "Analytical Calculation of the Envelope Surface for Generic Milling Tools Directly from CL-Data Based on the Moving Frame Method," Computer-Aided Design, **41**, pp. 848-855.

[24] Lee, S. W., and Nestler, A., 2011, "Complete Swept Volume Generation, Part I: Swept Volume of a Piecewise C^1-Continuous Cutter at Five-Axis Milling via Gauss Map," Computer-Aided Design, **43**, pp. 427-441.

[25] Lee, S. W., and Nestler, A., 2011, "Complete Swept Volume Generation—Part II: NC Simulation of Self-Penetration via Comprehensive Analysis of Envelope Profiles," Computer-Aided Design, **43**, pp. 442-456.

[26] Blackmore, D., Leu, M. C., and Wang, L. P., 1997, "The Sweep-Envelope Differential Equation Algorithm and its Application to NC Machining Verification," Computer-Aided Design, **29**, pp. 629-637.

[27] Chung, Y. C., Park, J. W., Shin, H., and Choi, B. K., 1998, "Modeling the Surface Swept by a Generalized Cutter for NC Verification," Computer-Aided Design, **30**, pp. 587-594.

[28] Aras, E., and Feng, H. Y., 2011, "Vector Model-Based Workpiece Update in Multi-Axis Milling by Moving Surface of Revolution," The International Journal of Advanced Manufacturing Technology, **52**, pp. 913-927.

[29] Jerard, R. B., Hussaini, S. Z., Drysdale, R. L., and Schaudt, B., 1989, "Approximate Methods for Simulation and Verification of Numerically Controlled Machining Programs," The Visual Computer, **5**, pp. 329-348.

[30] Park, J. W., Shin, Y. H., and Chung, Y. C., 2005, "Hybrid Cutting Simulation via Discrete Vector Model," Computer-Aided Design, **37**, pp. 419-430.

[31] Jang, D., Kim, K., and Jung, J., 2000, "Voxel-based virtual multi-axis machining," The International Journal of Advanced Manufacturing Technology, **16**, pp. 709-713.

[32] Lee, S. W., and Nestler, A., 2012, "Virtual Workpiece: Workpiece Representation for Material Removal Process," The International Journal of Advanced Manufacturing Technology, **58**, pp. 443-463.

[33] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., and Taubin, G., 1999, "The Ball-Pivoting Algorithm for Surface Reconstruction," Visualization and Computer Graphics, **5**, pp. 349-359.

[34] Möller, T., 1997, "A Fast Triangle-Triangle Intersection Test," Journal of Graphics Tools, **2**, pp. 25-30.

Appendix

		Cutter Diameter (mm)	Average edge length (mm)	Number of Triangles (cutter)	Number of Triangles (workpiece)	Number of Triangles (SV)	Execution time
Case #1	*Normal Performance*	100	1.54	44K	42K	160K	5min
	High Performance		0.77	181K	198K	1382K	2hr 30min
Case #2	*Normal Performance*	20	0.78	32K	86K	149K	10min
	High Performance		0.30	215K	343K	1129K	3hr
Case #3	*Normal Performance*	15	0.31	28K	125K	82K	3min
	High Performance		0.07	332K	499K	729K	9hr
Case #4	*Normal Performance*	15	1.25	3K	8K	15K	1min
	High Performance		0.31	52K	132K	240K	2hr
Case #5	*Normal Performance*	15	0.78	8K	40K	139K	6min
	High Performance		0.15	200K	1013K	1368K	5hr
Case #6	*Normal Performance*	10	1.25	9K	41K	16K	3min
	High Performance		0.31	113K	655K	205K	4hr 30min