An Integrated KD-Tree Cutter Size Determination Method for 3-Axis Finish Machining of Sculptured Surface Parts

Hai Qing Liang

A Thesis

in

the Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

October, 2008

Your file Votre référence
ISBN: 978-0-494-63227-7
Our file Notre référence
ISBN: 978-0-494-63227-7

ABSTRACT

An Integrated KD-Tree Cutter Size Determination Method for 3-Axis Finish Machining of Sculptured Surface Parts

Hai Qing Liang

In this research, an integrated KD-Tree cutter size search model is proposed to quickly determine the largest cutters and their accessible surface regions for 3-axis finish machining, without gouging and interference. By using these cutters, the highest material removal rate can be achieved, while maintaining the quality of the machined part. To overcome the problems of existing methods, such as long computational time and low accuracy, this model integrates the vertex KD-tree method with local gouging detection method. In this work, an imaginary cutter model is used to define a cutter, given a cutter contact point (CC point) and a testing point. All the testing points needed are derived from the part STL model. A simple and efficient algorithm is suggested to get rid of the redundant vertices in the STL file of the part. The local gouging detection method identifies the maximum local gouging-free cutter for a CC point as the initial cutter, and this cutter is used to determine the area of the cutter shadow. The cutter shadow is used to define a search range of the testing points to avoid any unnecessary search. Then the vertices covered by the shadow are quickly located by the KD-Tree algorithm from all the vertices, and used as test points to determine the final gouging and interference-free cutter size. The proposed model is tested with a hairdryer mould example. The results show that the model is not only computationally efficient, but also highly accurate. In

addition, the model is suitable for any types of milling cutters, and ready to implement in the CAD/CAM software.

Keywords: STL File, CATIA, Cutter Size Selection, KD-Tree search, Boundary, Gouging, Interference, Accuracy, and Computational Time.

ACKNOWLEDGMENTS

I would like to thank all of those who have given supports on the journey of my thesis research. Many professors, friends, and family members have assisted with techniques, ideas, and monetary resources to help me complete this research. While there were many people whose efforts bring this to fruition, a few people deserve special recognition. First I would like to thank my supervisor Dr. Chevy Chen for his guidance, patience and insight. Second, I would like to thank Qiang Fu for his programming supports and valuable ideas. Third, I would also like to thank Gang Liu, Maqsood Khan, Shuangxi Xie and Hong Zhen, whose encouragement, insights and comments have been the one of important driving factors for the advancement of this work.

Special thanks must go to my wife, Yu Hong Luan. She has provided support on every step of the way, emotionally, spiritually and even financially. Her support and encouragement have enabled me concentrate on my thesis program. I would like to thank my mother and my son for their understanding since I should have spent more time with them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Background

1.1.1 Sculptured surface machining

Today, there is a great demand for parts with complex geometric shapes due to the two main reasons. First, high aerodynamics, thermodynamics, or other performance requires a special shape design. Second, consumers are always in favour of products with better aesthetic appearance. These complex parts are characterized by their sculptured surfaces, which are challenging in design and manufacturing. A typical application of sculptured surface is the design of moulds and dies, as their geometries often include complex curved surfaces. Mathematically, sculptured surfaces can be expressed by non-periodic uniform rational B-Spline (NURBS) surface equations. Usually, many surface patches are required to form a part shape. The smoothness between patches is often defined as:

- Positional continuity (C^0): the end positions of two curves or surfaces are coincident.
- Tangential continuity (C^1): the end vectors of two curves or surfaces are in the same line.
- Curvature continuity (C^2): the curvatures of the two surface patches are equal.

To ensure aerodynamic requirements, it is very common that a sculptured part has to be designed by using NURBS patches with at least C^2 continuity. The typical examples include turbine blades, car bodies and boat hulls.

1.1.2 Gouging and interference

Gouging and interference are two main problems in sculptured surface machining as they could damage the workpiece and the machine tool. Gouging occurs on a part surface when the cutting tool overcuts the design surface near a cutter contact point or CC point (see Figure 1.1). In 3-axis machining, this is because the size of the selected cutter is not appropriate. In order to avoid gouging, the curvature of the selected cutter surface should not be smaller than that of the part surface. Interference happens when tool shank collides with the part surface during the machining (see Figure 1.2). This is usually because of the narrow open space of the part. The solution is to select a smaller cutter size. Frequently gouging is referred as local gouging as it happens in the vicinity of the CC point, while interference is referred as global gouging because it occurs away from the CC point.

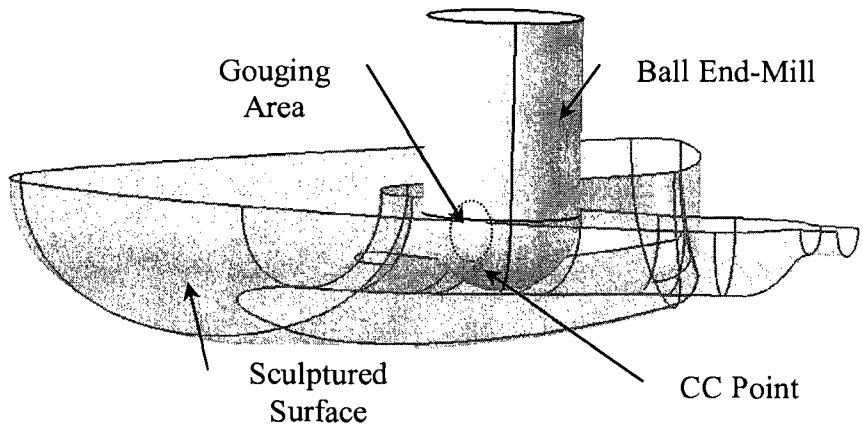


Figure 1.1 An illustration of gouging

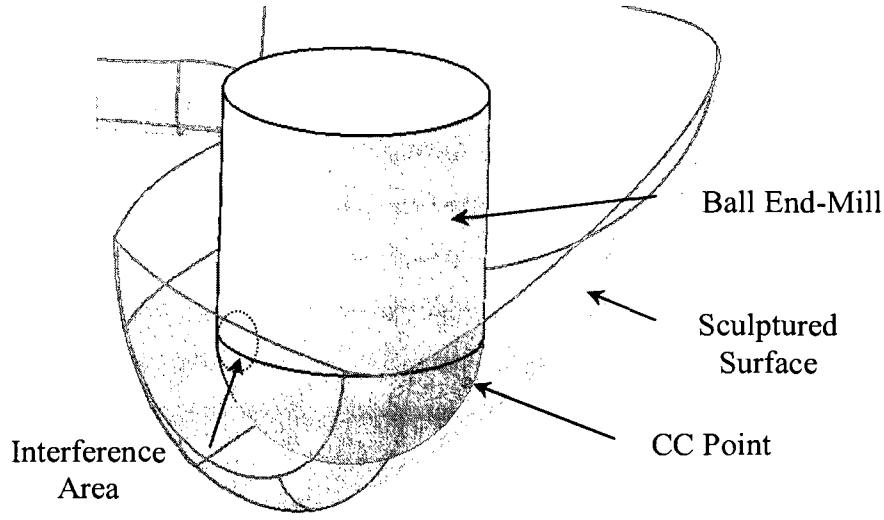


Figure 1.2 An illustration of interference

1.2 Problem Statement

In 3-axis sculptured surface machining, to meet the surface quality requirements, the part should not be gouged and interfered. This necessitates an appropriate cutter size for a given surface area. For a part with complex shapes, different surface areas may have different curvatures and open spaces for the cutter to access. Usually, to maximize the material removal rate, it is preferred to select the cutter as large as possible for each area. As a result, a high machining efficiency requires the use of multiple cutters with appropriate sizes. However, how to determine a group of optimal cutter sizes has been a challenge for a long time, and no mature solution has been found.

There is a commonly used method for this issue, which can be summarized as the following steps:

1) The cutter size is determined to avoid the local gouging through the curvature comparison or the distance calculation between the cutting surface and the designed surface at a CC point.

2) The interference is checked based on the obtained no-gouging cutter size. If the interference occurs, the cutter size is adjusted accordingly.

3) The check and the orientation adjustment may repeat several times until a satisfactory cutter size is chosen.

However, this method is time-consuming and the determined cutter size usually is not optimal. Besides, the machine efficiency is very low as only one small cutter is used. Consequently, a long lead time is required for the parts that have complex surfaces and requires a tight tolerance. For example, the average lead time for an American mould and die manufacturer is 20~30 weeks [1]. In our day, shortening the time to market is crucial for a business to succeed. As 10~15% of reduction in machining time would result in one week shorter in the lead time, it is important to seek solutions to this problem.

1.3 Literature Review

The main work of cutter size determination actually is gouging and interference detection. In surface machining, gouging and interference hinder the surface quality and production efficiency. To overcome these problems, many researches have been conducted in this topic. However, there is no effective solution to them so far. Generally,

two types of methods are comonly used to tackle this problems. One is curvature-related, while another one is non-curvature-related.

First, among the curvature-related methods, Glaeser et al. [2] and Pottmann et al. [3] introduced some concepts regarding exhaustive curvature comparison and discussed the local and global conditions for 3-axis collision-free milling of sculptured surfaces. But no feasible method was described for the implementation of these concepts and conditions in their work. Yoon et al.[4] proposed a local condition for 5-axis collision-free milling, using Taylor's quadratic approximation to represent the tool an part surfaces in the vicinity of a cutter contact point. They assumed that the approximation was accurate in a large area. As this assumption does not always hold, their work is unfeasible. Rao and Sarma [5] applied the curvature comparison method to detect local gouging for 5-axis machining using flat end-mill. However, no work has been done for bull-nose end-mill. Khan [6] proposed an improved approach to comprehensive curvature analysis for the engaged regions on both cutting surface and part surface in 3-axis finish machining. This method can be applied to flat, ball and bull-nose end-mills to determine the locally gouging-free cutter size. Unfortunately, his method is not able to solve the interference or global gouging problem, which plays the same important role as local gouging detection in cutter size determination.

Many non-curvature-related approaches also have been proposed for gouging detection and cutter size selection. Oliver et al. [7] located the regions with high curvatures in the first step, and then mainly performed the gouging and interference detection on these regions for 3-axis machining. Yang and Han [8] employed iso-phote curves on the

sculptured surfaces to determine the patches accessible in 3-axis machining, and optimized the cutting tool selection to minimize the cutting time. . Lee and Chang [9] used the maximum effective cutting radius to determine cutter sizes for 5-axis surface machining. George and Babu [10] applied the optimization techniques to determine the self-intersection curves of the cutter location surface, and then eliminated the locations causing local gouging. In order to get rid of the zones with potential interference, Hatna and Grieve [11] processed the surface first, and then generated interference-free tool paths in a simple sweeping process of the surface parametric space. The above methods have the inherent problems: low accuracy in gouging detection and /or tedious computation. Liu [12] proposed a close-form mathematical model (imaginary cutter size model) to determine the imaginary cutter size or the largest allowable cutter size, given a cutter contact point and any test point on a sculptured surface, and applied the particle swarm optimization (PSO) method to determine a group of gouging and interference-free cutter sizes, leading to the highest machining efficiency. The main problem of this method is its very large computational time required as any point on any surface of the part could be used to test the validity of the cutter size. Besides, his model is valid only for the surface whose normal at any CC point does not coincide with the machine tool axis, which is not true.

1.4 Research Objectives and General Approach

The objectives of this work are two folds. The first objective is to improve Liu's model so that it can apply to any surface conditions. To do that, a refined imaginary cutter size model will be derived to accommodate the special surface condition. The second

objective, as the ultimate goal, is to propose an integrated KD-Tree cutter size search method (iKD-Tree method), which can achieve the same result of Liu's method, but requires much less computational time for 3-axis finish machining of sculptured surface parts.

In the iKD-Tree method, the improved imaginary cutter model is adopted to determine a cutter size at a CC point and a test point. However, instead of random points identified by the PSO method in Liu's work, the vertices obtained from the part STL file are used as the test points. As we can imagine, the cutter can only be in contact with the surface area covered by the shadow of the maximum possible cutter or the projection of the maximum cutter area on the horizontal plane, assuming we look downward from the top of the cutter (see Figure 1.3). As a result, only the vertices within the shadow will be selected as test points.

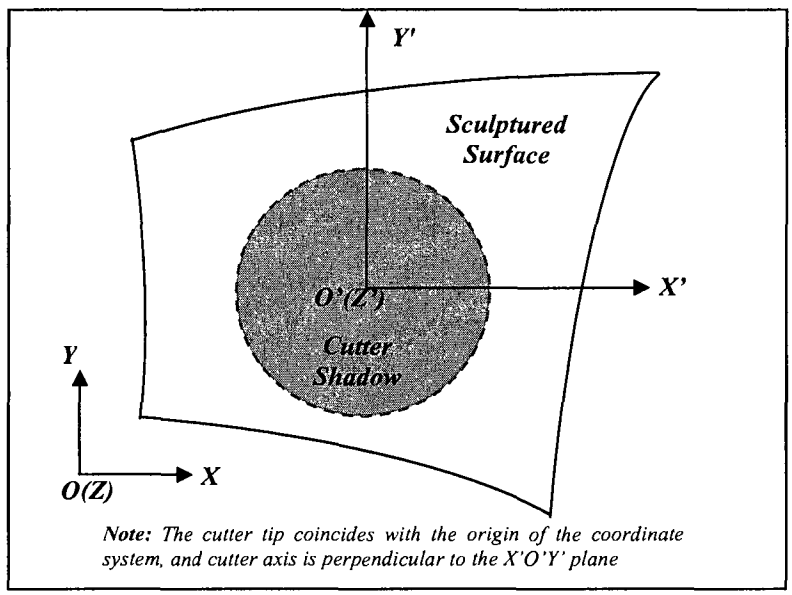


Figure 1.3 Cutter shadow on the surface

Theoretically, the maximum cutter could be the largest size available. However, the bigger size you choose, the higher computational time is needed. Thus Khan's comprehensive curvature analysis model will be adapted to determine a maximum local gouging-free cutter size as the initial input. This not only guarantees the cutter selected locally gouging-free, but also results in a great computational overhead saving.

After that, the KD-tree search, the most efficient region query method, is used to locate all the vertices falling within the shadow. At a CC point, a group of imaginary cutters can be obtained by using these vertices (testing points). The allowable cutter size for this CC point is simply the minimum cutter in the group.

1.5 Structure of the Thesis

This thesis contains eight chapters. In Chapter 1, some basic concepts concerning the surface machining are provided, followed by the problem statement, the literature review, research objectives and general approach. Chapter 2 introduces the imaginary cutter size model and its refinement. Chapter 3 presents the fundamental knowledge about the STL file and the method used to eliminate the vertex redundancy. Chapter 4 describes the approach to detecting the local gouging through comprehensive curvature analysis and the algorithm used to search the initial cutter size at a CC point. Chapter 5 explains the KD-tree search method and the procedures to determine the final cutter size at a cutting location. Chapter 6 deals with the structure and implementation of the integrated KD-tree model. In Chapter 7, the new model is applied to the hairdryer mould, and the results are compared with which from Liu's method and which from the CATIA rework function to

examine the efficiency and accuracy of the new model. Chapter 8 highlights the significance of this work and the future work directions.

CHAPTER 2. IMAGINARY CUTTER SIZE MODEL AND REFINEMENT

As Liu's imaginary cutter size model is a very good tool for defining a cutter size at a given CC point, it will continually be used to serve for the same purpose in this work. However, to accommodate all situations, some improvement has to be done on his model. First, a brief description of Liu's model is provided. Then, the refinement of Liu's model is proposed to overcome the limitation of his model.

2.1 Introduction of Liu's Imaginary Cutter Size Model

2.1.1 Compound surface

Due to the fact that the non-uniform rational B-splines (NURBS) surface can represent almost any smooth and curved surface with high accuracy, it becomes a trend to use the NURBS to model a complex surface or compound surface. Another advantage of the NURBS surfaces is that one can build a smooth curving surface or modify a surface to make it smooth with few control points. The parametric form NURBS surface is expressed as:

$$P(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}h_{i,j}\cdot P_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}h_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)},\ \textbf{with}\ \begin{cases}s_{k-1}\le u\le s_{n+1}\\ t_{l-1}\le v\le t_{m+1}\end{cases} \qquad (2.1)$$

Where, $N_{i,k}(u), N_{j,l}(v)$ are blending functions; u and v are the surface parameters; t_{l-1}, t_{m+1} and s_{k-1}, s_{n+1} are knot values limiting the finite intervals over which the blending functions have nonzero values; $P_{i,j}$ are the x, y, and z coordinates and $h_{i,j}$ are the homogeneous coordinates of the control points.

However, generally it is not practical to use only one NURBS equation or surface to model a complex surface in a part design. Instead, one should use a compound surface, a collection of NURBS surface patches connected with either position (C^0), tangent (C^1), or curvature (C^2) continuity between them.

For the purpose of convenience, the following notations are defined:

- $\mathbf{S}_i \ (i = 0,\ 1,\ ...,\ \mathrm{m})$: any patch of the compound surface
- $\mathbf{P}(u,v) \in \mathbf{S}_i,\ (i = 0,\ 1,\ ...,\ \mathrm{m})$: a point on $\mathbf{S}_i$
- $\Re_i^2 := \left[u_{i,\min}, u_{i,\max}; v_{i,\min}, v_{i,\min} \right]$: the domain of patch $\mathbf{S}_i$
- $\Re^2 = \Re_1^2 \cup \Re_2^2 \cup ... \Re_i^2 \cup ... \Re_\mathrm{m}^2$: the domain of the compound surface

Thus we can denote the compound surface as $\mathbf{S} := \mathbf{S}_1 \cup \mathbf{S}_2 \cup ... \mathbf{S}_i \cup ... \mathbf{S}_\mathrm{m}$. In CAD systems, the data of all the surface patches on a compound surface and the connectivity information between them are stored. As a result, for a given surface patch, one can use this information to identify its neighbouring surface patches in determining the maximum cutter size for CNC machining.

2.1.2 Representation of APT cutter geometry

The most common cutters for milling operations consist of standard flat, ball, and bull-nose end mills and they are special cases of the generic ISO APT cutter. In milling operations, the most commonly used cutting tools include standard flat, ball, or bull-nose end-mills, which are three special cases of the generic ISO APT cutter. Figure 2.1 shows an APT cutter and its three special cases are shown in Figure 2.2. Usually, the imaginary envelope formed by the spinning cutting edge of the APT cutter is defined as the cutting

surface, which can be further divided into three sub-surfaces: a tapered (A-B), a fillet (B-C), and a conical (C-O) surface. In practice, the taper angle θ is between zero to 20 degrees, and the conical angle ψ is small. Parameter r is the radius of the corner, R is the radial distance between the cutter axis and the center of the corner, and H is the cutter length.

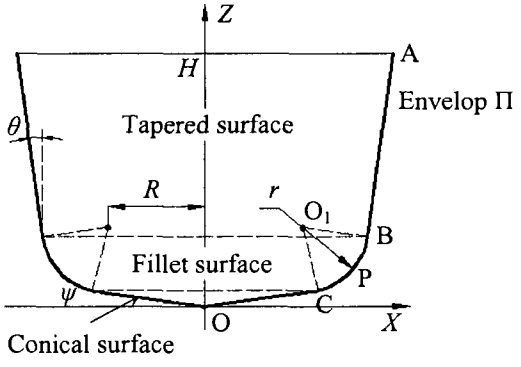


Figure 2.1 An illustration of APT cutter

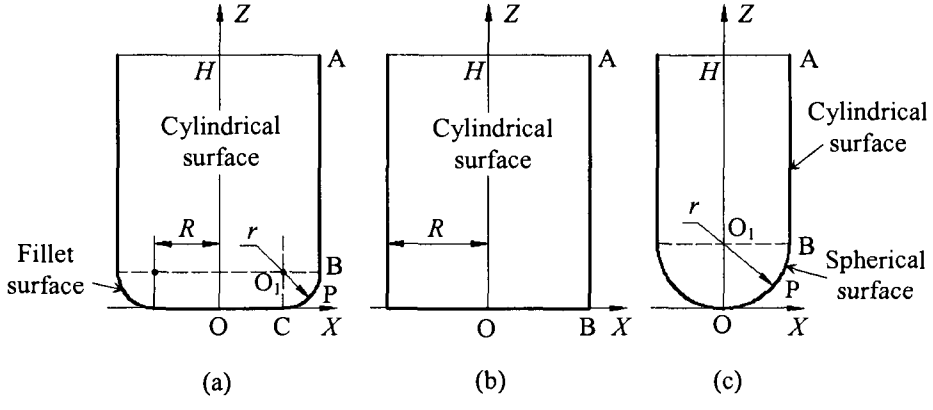


Figure 2.2 (a) a bull-nose end-mill, (b) a flat end-mill, and (c) a ball end-mill

A tool coordinate system is defined with its origin at the tip of the APT cutter and its Z-axis in line with the tool axis (refer to Figure 2.1). Using this coordinate system, the following expressions can be obtained:

- **The Coordinates of Key Points**

- Point C: $[x_c \quad z_c] = [R + r \cdot \sin\psi \quad (R + r \cdot \sin\psi)\tan\psi]$
- Point O_1: $[x_{o_1} \quad z_{o_1}] = [R \quad (R + r \cdot \sin\psi)\tan\psi + r \cdot \cos\psi]$ or

  $[x_{o_1} \quad z_{o_1}] = [R \quad R \cdot \tan\psi + r \cdot \sec\psi]$
- Point B: $[x_B \quad z_B] = [R + r \cdot \cos\theta \quad (R + r \cdot \sin\psi)\tan\psi + r \cdot \cos\psi - r \cdot \sin\theta]$ or

  $[x_B \quad z_B] = [R + r \cdot \cos\theta \quad R \cdot \tan\psi + r \cdot \sec\psi - r \cdot \sin\theta]$
- Point A: $[x_A \quad z_A] = [R + r \cdot \cos\theta + (H - z_B) \cdot \tan\theta \quad H]$, where,

  $z_B = (R + r \cdot \sin\psi)\tan\psi + r \cdot \cos\psi - r \cdot \sin\theta$

- **The surface equations of three sub-surfaces**
  - The conical surface: $T(z,\alpha) = \begin{bmatrix} \frac{\cos\alpha}{\tan\psi} \cdot z \\ \frac{\sin\alpha}{\tan\psi} \cdot z \\ z \end{bmatrix}, z \in [0, z_c]$, and $\alpha \in [0, 2\pi]$
  - The fillet surface: $T(z,\alpha) = \begin{bmatrix} [R + \sqrt{r^2 - (z_{o_1} - z)^2}]\cos\alpha \\ [R + \sqrt{r^2 - (z_{o_1} - z)^2}]\sin\alpha \\ z \end{bmatrix}$, and $\begin{cases} \alpha \in [0, 2\pi] \\ z \in [z_c, z_B] \end{cases}$
  - The tapered surface: $T(z,\alpha) = \begin{bmatrix} [x_B + (z - z_B) \cdot \tan\theta] \cdot \cos\alpha \\ [x_B + (z - z_B) \cdot \tan\theta] \cdot \sin\alpha \\ z \end{bmatrix}$, and

    $\begin{cases} \alpha \in [0, 2\pi] \\ z \in [z_c, z_B] \end{cases}$

Here, α is the angle measured from X-axis on XOY plane of the tool coordinate system.

Obviously, the APT cutter turns to be a bull-nose end-mill if $\theta = 0$ and $\psi = 0$, (see Figure 2.2(a)), a flat end-mill if $\theta = 0$, $\psi = 0$ and $r = 0$ (see Figure 2.2(b)), and a ball end-mill if $\theta = 0$, $\psi = 0$ and $R = 0$ (see Figure 2.2(c)).

For a standard cutter, its taper angle is limited to several degrees with discrete choices (e.g. 1°, 3°, 5°, 7° and 10°), and it also is common to set the conical angle $\psi = 0$. Similarly, for bull-nose end-mills, their corner radius r is often set to be $\frac{1}{32}$, $\frac{1}{16}$, or $\frac{1}{8}$ inch by tool manufacturers. For the simplicity purpose, the cutter size is optimized by varying only R for the bull-nose or flat end-mills, or only r for the ball end-mills, while fixing other parameters.

2.1.3 Theorem for allowable cutter size

It is common that a compound surface has complex shape and narrow open spaces. As a result, gouging and interference often happen during machining. To overcome this problem, Liu has proposed a theorem by taking both the part surface to be machined: $\mathbf{S}_0$, and its neighbouring surfaces or checking surfaces $\mathbf{S}_i$ $(i = 0,1,\ 2,\ ...,\ n)$ into account.

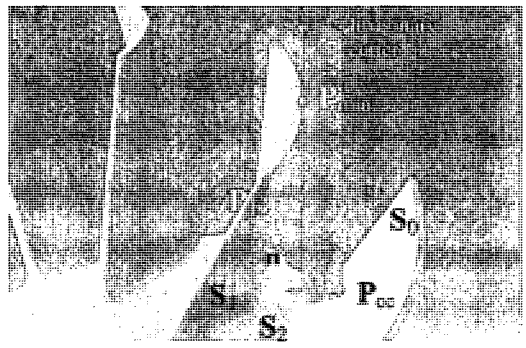


Figure 2.3 Illustration of the model of allowable cutter

As illustrated in Figure 2.3, a point $\mathbf{P}_{CC}$ on a bounded surface $\mathbf{S}_0$ to be cut is accessible in a part set-up, and the check surfaces $\mathbf{S}_i$ $(i = 1, 2, ..., n)$ are adjacent to this surface. An unique imaginary APT cutter in variable size R or r (for ball end mill) can be constructed, subject to the following constraints: (1) the fillet cutting surface of the cutter envelop $\Pi(r)$ is tangent to surface $\mathbf{S}_0$ at point $\mathbf{P}_{CC}$, and (2) this envelop passes through any other point **P** (called testing point) on one of these surfaces $\mathbf{S}_i$ $(i = 0, 1, 2, ..., n)$ ($\mathbf{S}_0$ is also a check surface). Thus, by changing the position of point **P** around all the testing surfaces, infinite imaginary cutters can be obtained. The smallest (called the allowable cutter) among all imaginary cutters can machine surface $\mathbf{S}_0$ at point $\mathbf{P}_{CC}$ without gouging and interference.

Let us denote the imaginary cutter size between $\mathbf{S}_0$ and $\mathbf{S}_i$ $(i = 0, 1, 2, ..., n)$ as $R_{\min}(\mathbf{S}_0, \mathbf{S}_i)$ if it is a flat or bull-nose end mill or $r_{\min}(\mathbf{S}_0, \mathbf{S}_i)$ if it is a ball end mill. The allowable cutter size can be obtained as:

$$R_{allowable} = \min[R_{\min}(S_0, S_0), \quad R_{\min}(S_0, S_1), \quad .... \quad R_{\min}(S_0, S_n),], \text{ or}$$

$$r_{allowable} = \min[r_{\min}(S_0, S_0), \quad r_{\min}(S_0, S_1), \quad .... \quad r_{\min}(S_0, S_n),]$$

Here, n is the total number of surface patches in the compound surface.

2.1.4 Imaginary cutter size model for the APT cutter

According to the above theorem, the imaginary cutter model can be developed. In Liu's research, he has developed two types of imaginary cutter model: the APT cutter model (for flat-end mill and bull-nose end mill), and the ball-end mill model. Since there is no need to refine the ball-end mill model in this work and both models share the similar idea, so only the APT cutter model is introduced here. The interested reader is encouraged to refer the Liu's work for details. Before the model development, some assumptions are needed:

- A 3-axis CNC milling machine is used to cut the part
- A CC point $\mathbf{P}_{cc} = \begin{bmatrix} x_{cc} & y_{cc} & z_{cc} \end{bmatrix}^T$ is on the surface $\mathbf{S}_0$ to be cut
- The testing point $\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is either on this surface or its check surfaces $\mathbf{S}_i (i = 0,1,2,...,n)$
- The unit surface normal of Point P $\boldsymbol{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ can be easily calculated, and the corner radius r, taper angle θ and conical angle ψ are specified, according to the industry standard choices
- R is the only parameter has to be determined as the imaginary APT cutter size.

Depending on the size of R, there are three cases for modeling a general APT cutter, as shown in Figure 2.4.

- When $0 < R < \infty$, the cutting envelop of the APT cutter consists of three portions: a tapered surface (TS), a fillet surface (FS) and a conical surface (CS).
- When $R = 0$, the three portions of cutting envelop shrinks to TS0, FS0 and CS0

- When $R = \infty$, the three portions of cutting envelop extends to TS∞, FS∞ and CS∞

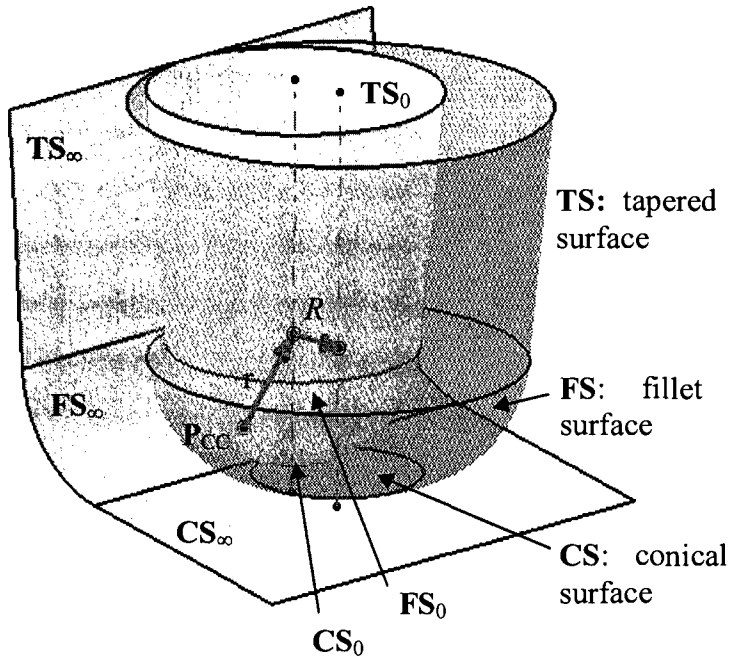


Figure 2.4 Example of imaginary cutter and its two extreme case: R=0 or R=∞

The cutting envelop of any feasible imaginary cutter should be confined in the volume bounded by $\mathbf{TS}_0$, $\mathbf{FS}_0$ & $\mathbf{CS}_0$ and $\mathbf{TS}_\infty$, $\mathbf{FS}_\infty$ and $\mathbf{CS}_\infty$. Specifically, no feasible imaginary cutter can be found if a testing point **P** is outside this volume, while there always exists a feasible cutter if P is inside this volume.

Two planes $\mathbf{PL}_1$ and $\mathbf{PL}_2$ are constructed to determine which portion of the cutting envelops will be in touch with the testing point **P**, as shown in Figure 2.5. $\mathbf{PL}_1$ is the plane where the intersection circle between **TS** and **FS** locates. $\mathbf{PL}_2$ is a plane where the intersection circle between **FS** and **CS** locates. Because only R is the variable in this model, $\mathbf{PL}_1$ and $\mathbf{PL}_2$ will remain unchanged regardless of the size of R. If **P** is above $\mathbf{PL}_1$,

only the tapered surface has chance to touch **P**. If **P** is between $\mathbf{PL}_1$ and $\mathbf{PL}_2$, **P** can only locate on the fillet surface. If **P** is below $\mathbf{PL}_2$, only the conical surface will pass **P**. Then, different formulas can be developed to calculate the imaginary cutter size, depending on the space location of **P** relative to two new constructed planes.

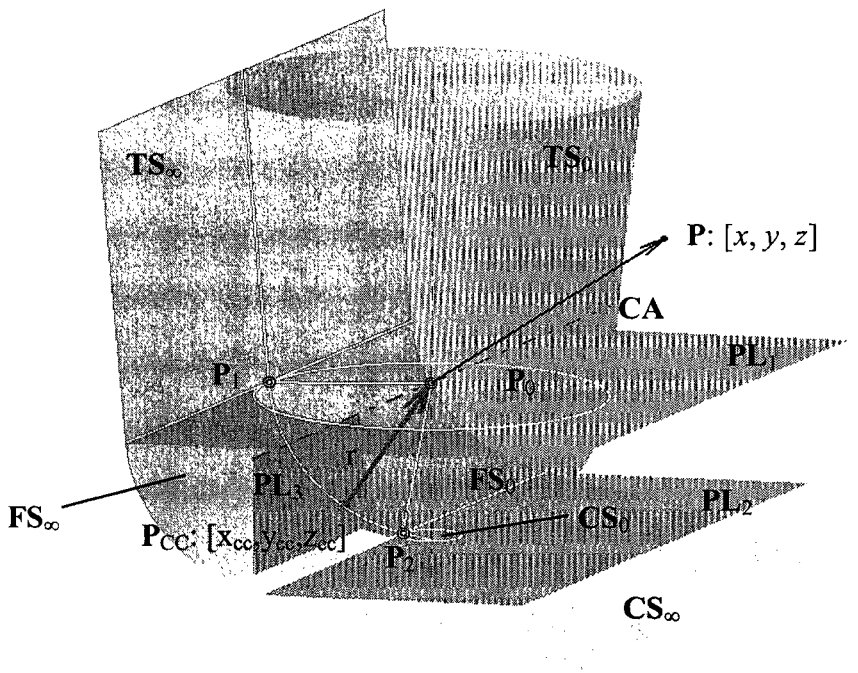


Figure 2.5 Some geometry features used in imaginary cutter model

As previously mentioned, given a CC point and a testing point, a unique imaginary APT cutter can be determined. Since in this model, θ, ψ and r of the cutter are specified as constants, the only variable needs to be determined is R. For the purpose of derivation, let $\Delta x = x - x_{cc} - r \cdot n_x$, $\Delta y = y - y_{cc} - r \cdot n_y$, $\Delta z = z - z_{cc} - r \cdot n_z$ and $\lambda = 1/\sqrt{n_x^2 + n_y^2}$. Different mathematical equations are derived to compute R, based on the relative Z coordinate of the testing point **P**.

Scenario 1: testing point **P** is above the plane $\mathbf{PL}_1$

Under this circumstance, only the tapered surface **TS** can touch **P** and we have $\Delta z \geq -r \cdot \sin\theta$. Then, three cases need to be further considered based on the location of the test point.

- **Case 1**

If **P** falls into the volume bounded by $\mathbf{TS}_0$, gouging will happen at P. As a result, no feasible cutter size can be found; that is, R = 0 Then, all satisfied testing points can be expressed as:

$$\Delta x^2 + \Delta y^2 \leq \left[(\Delta z + r\sin\theta)\cdot\tan\theta + r\cos\theta\right]^2 \text{ or}$$

$$\Delta x^2 + \Delta y^2 \leq (\Delta z \cdot \tan\theta + r\sec\theta)^2 \tag{2.2}$$

- **Case 2**

If **P** locates on or behind the plane $\mathbf{TS}_\infty$, any size of R will not cause gouging. Thus, R can be set as ∞. Surface normal of $\mathbf{TS}_\infty$ is $\overline{P_1P_0}$, so **P** should satisfy the inequality equation: $\overline{P_1P}\cdot\overline{P_1P_0} \leq 0$.

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} + r\cdot n_x - r\cdot\lambda\cdot n_x\cdot\cos\theta \\ y_{cc} + r\cdot n_y - r\cdot\lambda\cdot n_y\cdot\cos\theta \\ z_{cc} + r\cdot n_z - r\cdot\sin\theta \end{bmatrix}\right)\cdot\begin{bmatrix} r\cdot\lambda\cdot n_x\cdot\cos\theta \\ r\cdot\lambda\cdot n_y\cdot\cos\theta \\ r\cdot\sin\theta \end{bmatrix} \leq 0 \tag{2.3}$$

Inequality equation (2.3) can be simplified as:

$$\lambda\cdot n_x\cdot\Delta x\cdot\cos\theta + \lambda\cdot n_y\cdot\Delta y\cdot\cos\theta + \Delta z\cdot\sin\theta + r \leq 0 \tag{2.4}$$

- **Case 3**

If **P** is in the volume bounded by $\mathbf{TS}_0$ and $\mathbf{TS}_\infty$, there always exists a feasible imaginary cutter whose cutting surface tangent to the $\mathbf{P}_{CC}$ passing **P**, the following equation can be obtained:

$$\begin{aligned}&\left[x-(x_{cc}+r\cdot n_x+R\cdot\lambda\cdot n_x)\right]^2+\left[y-(y_{cc}+r\cdot n_y+R\cdot\lambda\cdot n_y)\right]^2\\&=\left\{R+r\cdot\cos\theta+\left[z-(z_{cc}+r\cdot n_z-r\cdot\sin\theta)\right]\cdot\tan\theta\right\}^2\end{aligned} \tag{2.5}$$

From equation(2.5), parameter R can be calculated:

$$R=\frac{1}{2}\cdot\frac{\Delta x^2+\Delta y^2-(\Delta z\cdot\tan\theta+r\cdot\sec\theta)^2}{\lambda\cdot n_x\cdot\Delta x+\lambda\cdot n_y\cdot\Delta y+\Delta z\cdot\tan\theta+r\cdot\sec\theta} \tag{2.6}$$

Scenario 2: P is between the planes $\mathbf{PL}_1$ and $\mathbf{PL}_2$

In this scenario, only the fillet surface can touch **P** and we have $-r\cdot\sin\theta\geq\Delta z\geq-r\cdot\cos\psi$. Similarly, three cases needs to be analyzed separately according to the location of the testing point **P**.

- **Case 1**

If **P** is in the volume bounded by $\mathbf{FS}_0$, gouging is unavoidable. This means that R has to be 0. Mathematically, all points must be satisfied the following condition:

$$\Delta x^2+\Delta y^2+\Delta z^2\leq r^2 \tag{2.7}$$

- **Case 2**

If **P** locates on or behind the surface of $\mathbf{FS}_\infty$, gouging will never happen. Consequently, R can be set as ∞. The criteria can be derived as follows:

Criterion one: **P** should be away enough from the axis of the cylinder surface $\mathbf{FS}_\infty$.

In Figure 2.5, $\mathbf{FS}_\infty$ is a cylinder surface, and let **CA** to represent its axis. $\mathbf{PL}_3$ is a reference plane passing through **CA** and perpendicular with the horizontal plane XY. If there exists a point $\mathbf{P}_p$ on the cylinder axis **CA** and $\overrightarrow{\mathbf{P}_p\mathbf{P}}$ is perpendicular with **CA**. Then, one condition to ensure gouging free is $\left|\overrightarrow{\mathbf{P}_p\mathbf{P}}\right| \geq r$.**CA** can be written as:

$$\overrightarrow{CA} = \begin{bmatrix} x_{cc} + n_x \cdot r - n_y \cdot t \\ y_{cc} + n_y \cdot r + n_x \cdot t \\ z_{cc} + n_z \cdot r \end{bmatrix}, \textit{ where } t \textit{ is a variable} \tag{2.8}$$

Because $\overrightarrow{\mathbf{P}_p\mathbf{P}} \perp \mathbf{CA}$, we have

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} + n_x \cdot r - n_y \cdot t \\ y_{cc} + n_y \cdot r + n_x \cdot t \\ z_{cc} + n_z \cdot r \end{bmatrix} \right) \cdot \begin{bmatrix} -n_y \\ n_x \\ 0 \end{bmatrix} = 0 \tag{2.9}$$

From equation(2.9), t can be obtained:

$$t = \frac{n_x \Delta y - n_y \Delta x}{n_x^2 + n_y^2} \tag{2.10}$$

Substituting t into(2.8) gives $\mathbf{P}_p$:

$$P_p = \begin{bmatrix} x_{pp} \\ y_{pp} \\ z_{pp} \end{bmatrix} = \begin{bmatrix} x_{cc} + n_x \cdot r - \dfrac{n_x n_y \Delta y - n_y{}^2 \Delta x}{n_x^2 + n_y^2} \\ y_{cc} + n_y \cdot r + \dfrac{n_x{}^2 \Delta y - n_x n_y \Delta x}{n_x^2 + n_y^2} \\ z_{cc} + n_z \cdot r \end{bmatrix} \tag{2.11}$$

Plugging $\mathbf{P}_p$ into $\overline{P_p P} \geq r$, we have:

$$\left[x - x_{cc} - n_x \cdot r + \frac{n_x n_y \Delta y - n_y{}^2 \Delta x}{n_x^2 + n_y^2} \right]^2 + \left[y - y_{cc} - n_y \cdot r - \frac{n_x{}^2 \Delta y - n_x n_y \Delta x}{n_x^2 + n_y^2} \right]^2 , \text{ or}$$
$$+ \left[z - z_{cc} - n_z \cdot r \right]^2 \geq r^2$$

$$\left[\frac{n_x n_y \Delta y + n_x{}^2 \Delta x}{n_x^2 + n_y^2} \right]^2 + \left[\frac{n_y{}^2 \Delta y + n_x n_y \Delta x}{n_x^2 + n_y^2} \right]^2 + \Delta z^2 \geq r^2 \tag{2.12}$$

Simplifying the In-eq.(2.12), we get the ***first criterion***:

$$\left[\lambda^2 \cdot \left(n_x n_y \Delta y + n_x{}^2 \Delta x \right) \right]^2 + \left[\lambda^2 \cdot \left(n_y{}^2 \Delta y + n_x n_y \Delta x \right) \right]^2 + \Delta z^2 \geq r^2 \tag{2.13}$$

Criterion two: **P** should be on the side of the plane $\mathbf{PL}_3$ where $\mathbf{P}_1$ locates.

Since $\mathbf{N}_{\mathbf{PL}_3}$, the surface normal of $\mathbf{PL}_3$, is $\left[n_x, n_y, 0 \right]^T$, from $\overrightarrow{\mathbf{P_p P}} \cdot \mathbf{N}_{\mathbf{PL}_3} \leq 0$, we have:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} + n_x \cdot r - \dfrac{n_x n_y \Delta y - n_y{}^2 \Delta x}{n_x^2 + n_y^2} \\ y_{cc} + n_y \cdot r + \dfrac{n_x{}^2 \Delta y - n_x n_y \Delta x}{n_x^2 + n_y^2} \\ z_{cc} + n_z \cdot r \end{bmatrix} \right) \cdot \begin{bmatrix} n_x \\ n_y \\ 0 \end{bmatrix} \leq 0 \qquad (2.14)$$

Finally, the ***second criterion*** can be obtained by simplifying(2.14):

$$n_x \cdot \left(\Delta x \cdot n_x{}^2 + n_x n_y \Delta y \right) + n_y \cdot \left(\Delta y \cdot n_y{}^2 + n_x n_y \Delta x \right) \leq 0 \qquad (2.15)$$

So, when both inequality equation (2.13)and (2.15) hold, $R = \infty$.

- **Case 3**

Otherwise, a feasible imaginary cutter exists, which can be obtained by the following equation:

$$\left[x - (x_{cc} + r \cdot n_x + R \cdot \lambda \cdot n_x) \right]^2 + \left[y - (y_{cc} + r \cdot n_y + R \cdot \lambda \cdot n_y) \right]^2 = \left(R + \sqrt{r^2 - \Delta z^2} \right)^2 \qquad (2.16)$$

From(2.16), we have:

$$R = \frac{1}{2} \cdot \frac{\Delta x^2 + \Delta y^2 + \Delta z^2 - r^2}{\lambda \cdot n_x \cdot \Delta x + \lambda \cdot n_y \cdot \Delta y + \sqrt{r^2 - \Delta z^2}} \qquad (2.17)$$

Scenario 3: **P** is below the plane $\mathbf{PL}_2$

In this condition, only conical surface **CS** can touch **P** as shown inFigure 2.5, and we have $\Delta z \le -r \cdot \cos\psi$. Likewise, three cases need to be handled one by one, based on the location of the testing point **P**.

- **Case 1**

If **P** is in the volume bounded by $\mathbf{CS}_0$, there is no doubt that gouging will happen. Therefore, R can be set as 0. The testing point **P** should meet the following inequality equation:

$$\sqrt{\Delta x^2 + \Delta y^2} - \left(\frac{\Delta z + r \cdot \cos\psi}{\tan\psi} + r \cdot \sin\psi \right) \le 0, \text{ or}$$

$$\Delta x^2 + \Delta y^2 \le \left(\frac{\Delta z + r \cdot \sec\psi}{\tan\psi} \right)^2, \text{ and } \Delta z > -r \cdot \sec\psi \qquad (2.18)$$

- **Case 2**

In Figure 2.4, if **P** is below $\mathbf{CS}_\infty$, no gouging will happen. Hence, R can be set as ∞. From Figure 2.5, we know the surface normal of $\mathbf{CS}_\infty$ is $\overrightarrow{\mathbf{P}_2\mathbf{P}_0}$ ($\mathbf{P}_2$ corresponds to Point C in Figure 2.1), and $\overrightarrow{\mathbf{P}_2\mathbf{P}_0} = \left[\lambda \cdot n_x \cdot \sin\psi, \quad \lambda \cdot n_y \cdot \sin\psi, \quad \cos\psi \right]^T$. So **P** has to meet the following inequality equation if $R = \infty$:

$$\overrightarrow{\mathbf{P}_2\mathbf{P}} \cdot \overrightarrow{\mathbf{P}_2\mathbf{P}_0} \le 0 \qquad (2.19)$$

Then, by substituting the specific expressions into(2.19), we have:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} + r \cdot n_x - r \cdot \lambda \cdot n_x \cdot \sin\psi \\ y_{cc} + r \cdot n_y - r \cdot \lambda \cdot n_y \cdot \sin\psi \\ z_{cc} + r \cdot n_z - r \cdot \cos\psi \end{bmatrix}\right) \cdot \begin{bmatrix} \lambda \cdot n_x \cdot \sin\psi \\ \lambda \cdot n_y \cdot \sin\psi \\ \cos\psi \end{bmatrix} \leq 0 \qquad (2.20)$$

Inequality equation (2.20) can be simplified as:

$$\lambda \cdot n_x \cdot \Delta x \cdot \sin\psi + \lambda \cdot n_y \cdot \Delta y \cdot \sin\psi + \Delta z \cdot \cos\psi + r \leq 0 \qquad (2.21)$$

- **Case 3**

Otherwise, **P** should be on the conical surface **CS** of a feasible imaginary cutter. Then, we have the following equation:

$$\begin{aligned} &\left[x - (x_{cc} + r \cdot n_x + R \cdot \lambda \cdot n_x)\right]^2 + \left[y - (y_{cc} + r \cdot n_y + R \cdot \lambda \cdot n_y)\right]^2 \\ &= \left[\frac{z - (z_{cc} + r \cdot n_z - r \cdot \cos\psi)}{\tan\psi} + r \cdot \sin\psi + R\right]^2 \end{aligned} \quad \text{, or}$$

$$\left(\Delta x - R \cdot \lambda \cdot n_x\right)^2 + \left(\Delta y - R \cdot \lambda \cdot n_y\right)^2 = \left(\frac{\Delta z + r \cdot \cos\psi}{\tan\psi} + r \cdot \sin\psi + R\right)^2 \qquad (2.22)$$

Simplifying (2.22) gives:

$$R = \frac{1}{2} \cdot \frac{\Delta x^2 + \Delta y^2 - \left(\frac{\Delta z}{\tan\psi} + \frac{r}{\sin\psi}\right)^2}{\lambda \cdot n_x \cdot \Delta x + \lambda \cdot n_y \cdot \Delta y + \left(\frac{\Delta z}{\tan\psi} + \frac{r}{\sin\psi}\right)} \qquad (2.23)$$

2.2 Refinement of Liu's Model

2.2.1 Limitation of Liu's model

When Liu defined the tool coordinate system in his model, implicitly he assumed that the X axis of that coordinate system coincides with the projection of the CC point normal (N_{cc}) on the XOY plane in the part coordinate system. As illustrated in Figure 2.6, P_{cc}, O, and n_{xy} are the projections of CC point, tool center, and N_{cc} on the XOY plane, respectively. In addition, n_y and n_y are the X and Y components of N_{cc}, in that order. Thus $\overrightarrow{P_{cc}O_1}$ and $\overrightarrow{O_1O}$ are the projections of corner radius r and radius R on the XOY plane, correspondingly, and they have to be in line with the direction of n_{xy}. We can uniquely define the tool center O if N_{cc} is not perpendicular to the XOY plane or at least one of n_x and n_y is not zero.

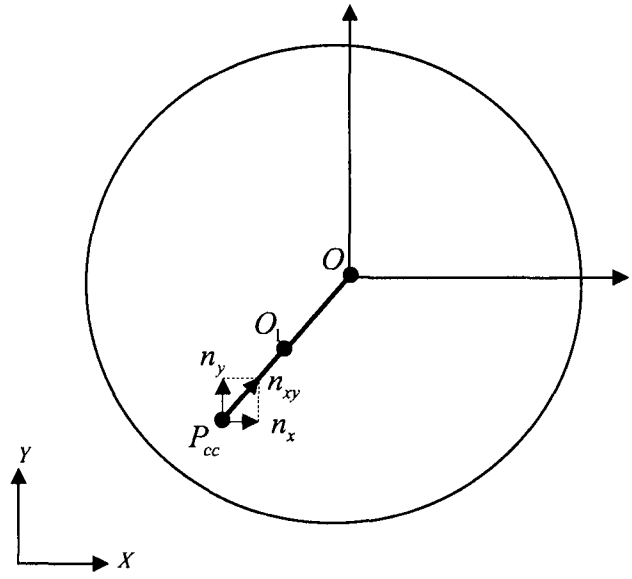


Figure 2.6 Determination of the cutter center O by the normal of CC Point

However, when N_{cc} is parallel to the Z axis, N_{cc} has neither x component, nor y component; that is, $n_x = n_y = 0$. As a result, the tool center cannot be uniquely defined if the cutter is a flat end mill or torus mill (APT cuter case in Liu's model). In this case, Liu's model is no longer valid. Thus, a revised new model is provided to address this special case.

2.2.2 Upgrading of Liu's model for the special case

To develop the new imaginary cutter size model for the special case where the surface normal N_{cc} is perpendicular to the XOY plane, we need to examine the trajectory of the cutter center O. In this case, $n_x = n_y = 0$, the trajectory of O is a circle centered at P_{cc} with a radius of R' as shown in Figure 2.7. Here $R' = r \cdot t + R$, where $t = \sqrt{n_x^2 + n_y^2}$ (r, and R are defined in Figure 2.1). For any cutter center location O_i on the circle, an angle φ_i is formed by the radius $O_i P_{cc}$ and the horizontal line that passes P_{cc}.

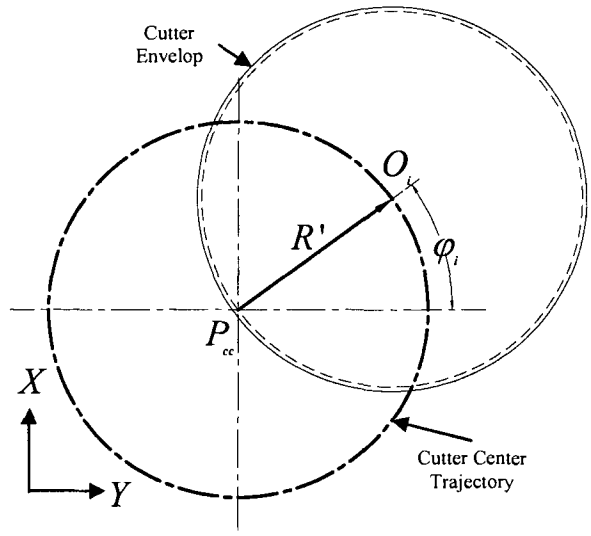


Figure 2.7 Center trajectory of the cutter

For a given angle φ_i ($0 \le \phi_i \le 360^0$), the x and y coordinates of O_i are calculated as:

$$\begin{cases} x_{O_i} = x_{cc} + R'\cos\varphi_i \\ y_{O_i} = y_{cc} + R'\sin\varphi_i \end{cases}$$

As $n_x = n_y = 0$, we have $\Delta x = x - x_{cc}$, $\Delta y = y - y_{cc}$ and $\Delta z = z - z_{cc} - r$. Since the cutter center location O_i depends on angle φ_i, it will become a new variable in the new revised model. Like Liu's original model, the revision of model also needs to address the same three scenarios, each of which has three cases. Here only the portions involving modification are described as the derivations of unchanged cases can be obtained from the original model. If the angle φ_i is given, the new model can be derived as follows.

Scenario 1: testing point **P** is above the plane $\mathbf{PL}_1$

- **Case 1**

(Remains unchanged)

- **case 2**

Here, $P_1P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} - r\cdot\cos\varphi_i\cdot\cos\theta \\ y_{cc} - r\cdot\sin\varphi_i\cdot\cos\theta \\ z_{cc} + r - r\cdot\sin\theta \end{bmatrix}$ and $P_1P_0 = \begin{bmatrix} r\cdot\cos\varphi_i\cdot\cos\theta \\ r\cdot\sin\varphi_i\cdot\cos\theta \\ r\cdot\sin\theta \end{bmatrix}$, so $\overrightarrow{P_1P}\cdot\overrightarrow{P_1P_0} \le 0$

becomes:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} - r\cdot\cos\varphi_i\cdot\cos\theta \\ y_{cc} - r\cdot\sin\varphi_i\cdot\cos\theta \\ z_{cc} + r - r\cdot\sin\theta \end{bmatrix}\right)\cdot\begin{bmatrix} r\cdot\cos\varphi_i\cdot\cos\theta \\ r\cdot\sin\varphi_i\cdot\cos\theta \\ r\cdot\sin\theta \end{bmatrix} \le 0 \qquad (2.24)$$

So, the inequality equation(2.4) is replaced by (2.24) or its simplified form (2.25):

$$\cos\varphi_i \cdot \Delta x \cdot \cos\theta + \sin\varphi_i \cdot \Delta y \cdot \cos\theta + \Delta z \cdot \sin\theta + r \le 0 \tag{2.25}$$

- **Case 3**

In case 3 of the new model, $\cos\varphi_i$ and $\sin\varphi_i$ are used to determine the center location of the cutter, which play the same roles as n_x and n_y, respectively. Therefore equation (2.5) is replaced with the following equation:

$$\begin{aligned}&\left[x-(x_{cc}+r\cdot\cos\varphi_i+R\cdot\cos\varphi_i)\right]^2+\left[y-(y_{cc}+r\cdot\sin\varphi_i+R\cdot\sin\varphi_i)\right]^2\\&=\left\{R+r\cdot\cos\theta+\left[z-(z_{cc}+r\cdot n_z-r\cdot\sin\theta)\right]\cdot\tan\theta\right\}^2\end{aligned} \tag{2.26}$$

From (2.26) parameter R can be calculated:

$$R=\frac{1}{2}\cdot\frac{\Delta x^2+\Delta y^2-(\Delta z\cdot\tan\theta+r\cdot\sec\theta)^2}{\cos\varphi_i\cdot\Delta x+\sin\varphi_i\cdot\Delta y+\Delta z\cdot\tan\theta+r\cdot\sec\theta} \tag{2.27}$$

For flat-end mill, (2.27) can be further simplified as:

$$R=\frac{1}{2}\cdot\frac{\Delta x^2+\Delta y^2-(\Delta z\cdot\tan\theta)^2}{\Delta x\cdot\cos\phi_i+\Delta y\cdot\sin\phi_i+\Delta z\cdot\tan\theta} \tag{2.28}$$

Scenario 2: P is between the planes $\mathbf{PL}_1$ and $\mathbf{PL}_2$

- **Case 1**

(Remains unchanged)

- **Case 2**

In case 2 of the new model, the equation of CA: $\overrightarrow{CA} = \begin{bmatrix} x_{cc} - n_x \cdot t \\ y_{cc} + n_y \cdot t \\ z_{cc} + r \end{bmatrix}$ (where t is a variable)

is replaced by:

$$\overrightarrow{CA} = \begin{bmatrix} x_{cc} - \sin\phi_i \cdot t \\ y_{cc} + \cos\phi_i \cdot t \\ z_{cc} + r \end{bmatrix} \tag{2.29}$$

Accordingly, $\overrightarrow{\mathbf{P}_p\mathbf{P}} \perp \mathbf{CA}$ becomes:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} - \sin\phi_i \cdot t \\ y_{cc} + \cos\phi_i \cdot t \\ z_{cc} + r \end{bmatrix} \right) \cdot \begin{bmatrix} -\sin\phi_i \\ \cos\phi_i \\ 0 \end{bmatrix} = 0 \tag{2.30}$$

From(2.30), we get:

$$t = \cos\phi_i \Delta y - \sin\phi_i \Delta x \tag{2.31}$$

Substitute t into(2.29), $\mathbf{P}_p$ can be calculated:

$$P_p = \begin{bmatrix} x_{pp} \\ y_{pp} \\ z_{pp} \end{bmatrix} = \begin{bmatrix} x_{cc} - \cos\phi_i \sin\phi_i \Delta y + \sin\phi_i^2 \Delta x \\ y_{cc} + \cos^2\phi_i \Delta y - \cos\phi_i \sin\phi_i \Delta x \\ z_{cc} + n_z \cdot r \end{bmatrix} \tag{2.32}$$

Substituting $\mathbf{P}_p$ into $\overline{P_pP} \geq r$, we have the following expression:

$$\left[x - x_{cc} + \cos\phi_i \sin\phi_i \Delta y - \sin^2\phi_i \Delta x\right]^2 + \left[y - y_{cc} - \cos^2\phi_i \Delta y + \cos\phi_i \sin\phi_i \Delta x\right]^2 + \left[z - z_{cc} - n_z \cdot r\right]^2 \geq r^2, \text{ or}$$

$$\left[\cos\phi_i \sin\phi_i \Delta y + \cos^2\phi \Delta x\right]^2 + \left[\cos\phi_i \sin\phi_i \Delta x + \sin^2\phi \Delta y\right]^2 + \Delta z^2 \geq r^2 \qquad (2.33)$$

Simplifying(2.33), we have the new **first criterion**,

$$\left[\cos^2\phi_i \Delta x + \cos\phi_i \sin\phi_i \Delta y\right]^2 + \left[\sin^2\phi_i \Delta y + \cos\phi_i \sin\phi_i \Delta x\right]^2 + \Delta z^2 \geq r^2 \qquad (2.34)$$

Here, the surface normal of $\mathbf{PL}_3$ has to be changed into $[\cos\varphi, \sin\varphi, 0]^T$. Therefore, $\overline{P_p P} \cdot [\cos\varphi, \sin\varphi, 0]^T \leq 0$ is expressed as:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} - \cos\phi_i \sin\phi_i \Delta y + \sin\phi_i^2 \Delta x \\ y_{cc} + \cos^2\phi_i \Delta y - \cos\phi_i \sin\phi_i \Delta x \\ z_{cc} + n_z \cdot r \end{bmatrix}\right) \cdot \begin{bmatrix} \cos\phi_i \\ \sin\phi_i \\ 0 \end{bmatrix} \leq 0 \qquad (2.35)$$

Simplifying(2.35), we obtain the new **second criterion**:

$$\cos\phi_i \left(\cos^2\phi \Delta x + \cos\phi_i \sin\phi_i \Delta y\right) + \sin\phi_i \left(\sin^2\phi \Delta y + \cos\phi_i \sin\phi_i \Delta x\right) \leq 0 \qquad (2.36)$$

In this case, when both Eq. (2.18) and Eq. (2.20) hold true, $R = \infty$.

- **Case 3**

In this case of the new model, the equation used to define R is modified as:

$$\left[x - (x_{cc} + r \cdot n_x + R \cdot \cos\varphi_i)\right]^2 + \left[y - (y_{cc} + r \cdot n_y + R \cdot \sin\varphi_i)\right]^2 = \left(R + \sqrt{r^2 - \Delta z^2}\right)^2 \qquad (2.37)$$

From(2.37), we have:

$$R = \frac{1}{2} \cdot \frac{\Delta x^2 + \Delta y^2 + \Delta z^2 - r^2}{\cos\varphi_i \cdot \Delta x + \sin\varphi_i \cdot \Delta y + \sqrt{r^2 - \Delta z^2}} \tag{2.38}$$

Scenario 3: **P** is below the plane $\mathbf{PL}_2$

- **Case 1**

(Remains unchanged)

- **Case 2**

In case 2, the surface normal of $\mathbf{CS}_\infty$ is $\overrightarrow{\mathbf{P}_2\mathbf{P}_0} = \left[\cos\varphi_i \cdot \sin\psi,\ \sin\varphi_i \cdot \sin\psi,\ \cos\psi\right]^T$.

Recall $\overrightarrow{\mathbf{P}_2\mathbf{P}} \cdot \overrightarrow{\mathbf{P}_2\mathbf{P}_0} \le 0$ if $R = \infty$. Thus we can get the following equation by substituting the two vectors:

$$\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} x_{cc} + r \cdot n_x - r \cdot \cos\varphi_i \cdot \sin\psi \\ y_{cc} + r \cdot n_y - r \cdot \sin\varphi_i \cdot \sin\psi \\ z_{cc} + r \cdot n_z - r \cdot \cos\psi \end{bmatrix} \right) \cdot \begin{bmatrix} \cos\varphi_i \cdot \sin\psi \\ \sin\varphi_i \cdot \sin\psi \\ \cos\psi \end{bmatrix} \le 0 \tag{2.39}$$

The inequality equation (2.39) can be simplified as:

$$\cos\varphi_i \cdot \Delta x \cdot \sin\psi + \sin\varphi_i \cdot \Delta y \cdot \sin\psi + \Delta z \cdot \cos\psi + r \le 0 \tag{2.40}$$

- **Case 3**

In case 3, to determine a feasible imaginary cutter, the equation for determining a feasible imaginary cutter is altered as:

$$\left[x - (x_{cc} + r \cdot n_x + R \cdot \cos\varphi_i)\right]^2 + \left[y - (y_{cc} + r \cdot n_y + R \cdot \sin\varphi_i)\right]^2$$
$$= \left[\frac{z - (z_{cc} + r \cdot n_z - r \cdot \cos\psi)}{\tan\psi} + r \cdot \sin\psi + R\right]^2$$

, or

$$\left(\Delta x - R\cdot\cos\varphi_i\right)^2 + \left(\Delta y - R\cdot\sin\varphi_i\right)^2 = \left(\frac{\Delta z + r\cdot\cos\psi}{\tan\psi} + r\cdot\sin\psi + R\right)^2 \qquad (2.41)$$

Simplifying (2.41) gives:

$$R = \frac{1}{2}\cdot\frac{\Delta x^2 + \Delta y^2 - \left(\frac{\Delta z}{\tan\psi} + \frac{r}{\sin\psi}\right)^2}{\cos\varphi_i\cdot\Delta x + \sin\varphi_i\cdot\Delta y + \left(\frac{\Delta z}{\tan\psi} + \frac{r}{\sin\psi}\right)} \qquad (2.42)$$

Since the angle ϕ ranges from 0^0 to 360^0, the cutter radius R for this case is the maximum of all the radii resulting from all the angles φ_i (i=1,2,….n) . Mathematically, we have:

$$R = \max(R_1, R_2, ..., R_i, ..., R_n)\ (i = 1,2,....n)$$

Two ways can be considered to obtain the radius R. one way is that we can use PSO optimization method to search for the maximum possible cutter radius among all the possible angles. The second way is that we can discretize the angle domain of 360^0 into a series of angles such that the difference between two consecutive angles is constant. For example, if we set $\varphi_i = (1^0, 2^0, ...i^0.., 360^0)$, the desired cutter is the maximum cutter among all the radii calculated from these 360 angles. The first method is more likely optimal but at the expense of computational time, while the second method may not be always the best but good enough. Plus, it is simple and fast. In this work, the second approach is used.

CHAPTER 3. STL FORMAT AND DATA PROCESSING

In Liu's work, he has optimized the cutter size search using the PSO method. Theoretically, the testing point can be taken from any surface and any location of this surface to ensure the accuracy. However, this good accuracy is at the expense of a huge computational time. In the STL file, due to the adaptive triangulation mesh, the number of triangles used to approximate the surface is proportional to the surface curvature. In fact, this matches our needs, as in the surface area with a high curvature, more testing points are demanded; while, in the surface area with a small curvature, less testing points are required. Therefore, high efficiency and reasonable accuracy can be achieved if we use the vertices of triangles extracted from the STL file as the testing points. Due to the advantages of STL file, this work uses the discrete vertices extracted from the STL as the testing points for the cutter size search. In this chapter, a general picture of the STL format is described. Then the method of filtering the redundant STL vertices is provided.

3.1 Introduction to the STL Format

3.1.1 Basics of the STL format

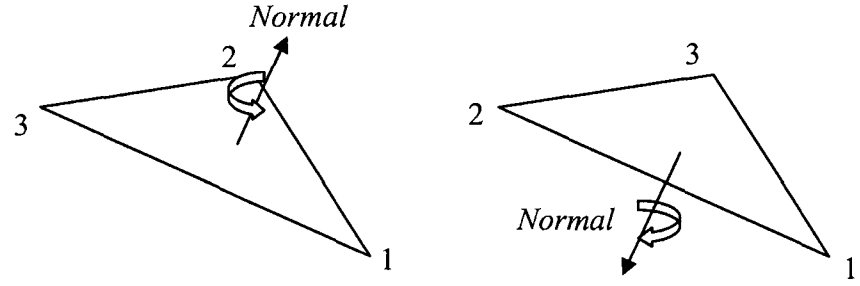
- **Definition of STL file**

STL is the abbreviation of stereoligthography. An STL file is a triangular representation of a 3D surface geometry. In an STL file, the surface is approximated by a series of oriented triangles (facets). Each facet is defined by a unit outward normal and three vertices listed in a specified order. The size of the facet is controlled by the surface

curvature and the tolerance that controls the surface quality. The STL format must respect the following rules [13]:

1) In each facet, its normal and each of three vertices are specified by three coordinates, so a total of 12 numbers is stored for each facet.
2) The vertices are listed in counter-clockwise order, if the viewing direction is from the outside of the object (using right-hand rule).
3) The normal of a facet must be the same as the facet orientation derived from its three vertices using the above mentioned right-hand rule.
4) Each triangle must share two vertices with each of its adjacent triangles, which is known as the vertex-to-vertex rule.

An example of facet is shown in Figure 3.1. In Figure 3.1a), three vertices (1, 2, and 3) are listed in counter-clockwise order, so the facet orientation obtained by right-hand rule is pointing up, which is the same as the normal direction. However, in Figure 3.1b), the three vertices are listed in clockwise order if we view it in the same direction; as a result, the normal direction of the facet is reversed or pointing down.



a) Normal pointing up *b*) Normal pointing down

Figure 3.1 Example of a facet and its normal

- **Two data formats**

The standard STL data can be categorized into two formats: ASCII and binary. The ASCII form is more descriptive and easy to read, but takes much more space to store the same CAD geometry, compared to the binary form. The structures of the two formats will be explained one by one.

An ASCII file starts with a description line "solid solid_name" , where solid_name refers the name of your design. The last line is the keyword "endsolid". Between the first line and last line, the facets are listed one by one. Each facet uses a consistent data structure, including the facet normal (e.g. N_x, N_y *and* N_z) and coordinates of three vertices (e.g. X_1, Y_1, Z_1; X_2, Y_2, Z_2; and X_3, Y_3, Z_3). The structure is illustrated in Figure 3.2 [14].

solid solid_name

<facet list>
facet normal Nx Ny Nz
outer loop
vertex X1 Y1 Z1
vertex X2 Y2 Z2
vertex X3 Y3 Z3
endloop
endfacet

...

endsolid

Figure 3.2 Structure of a STL file

A binary STL file also has an 80 byte header line containing the comment of the design. Then the next four bytes is a long integer, which stores the total number of facets. After that, all the facet information is listed. The data of each facet consists of a normal and

three vertices. Between each facet is a 2 byte spacer. Each facet requires 50 bytes, 12 for the normal, 36 for the 3 vertices, and 2 for the spacer. Therefore, the binary STL file is more compact and more efficient for data processing.

3.1.2 Advantages of STL file

STL format has the topologically simple and robust nature. First, it contains only one type of element, a triangular facet, which makes the geometry description homogenous, CAD-kernel independent, modeling history independent. In addition, the calculations involved in the generation and slicing of STL triangular facets are easy, fast, and accurate enough to satisfy the rapid prototyping industry. Furthermore, it is reasonably suitable to be the interface between a object model and the layer-by-layer fabrication[15]. As a result, it is widely used by most commercial CAD and CAM software and rapid prototyping equipment. Figure 3.3 shows both the CAD model and STL model of a complex part.

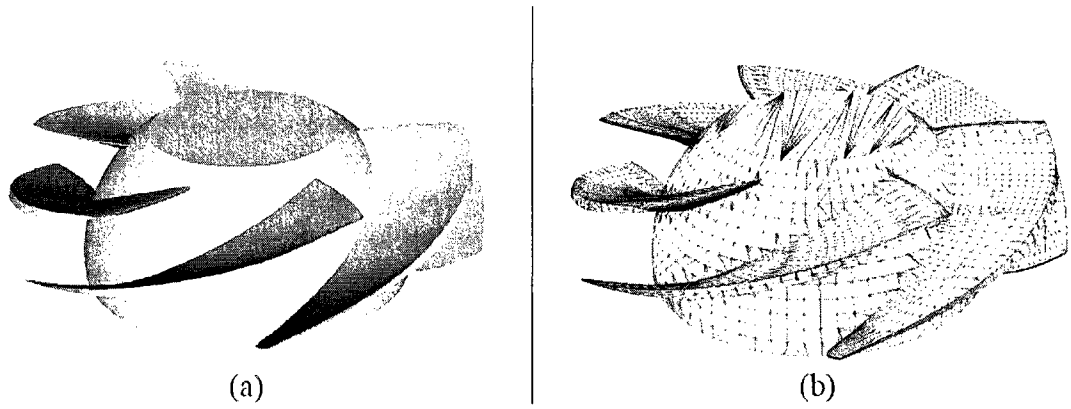


Figure 3.3 Example of a complex shaped part: a) CAD model, b) STL model

However, the STL format does have some disadvantages. First, inconsistency of normal happens when the facet normal generated by the CAGD system is different from that derived from the facet vertices, or when the normals of adjacent triangular facets are inconsistent. Another flaw is that the facet may collapse to be a gap when it is too thin to

keep its triangular shape. The third problem is illegal overlap. This happens when a vertex is located on the edge of another facet or when two adjacent facets are partially overlapping, which breaks the STL rule that each triangular facet must share two vertices with each adjacent facet. Finally, the depiction of geometric elements in STL format is redundant. For example, on average, each vertex of a facet is recorded repeatedly four times, which takes extra computational time and memory. In this work, the part STL file used is assumed to be perfect.

3.1.3 STL model accuracy control

In STL format, the model accuracy is determined by the number of the facets used. The more facets we use, the more accurate the model is. The number of facets is controlled by the sag values set in the CAD software [16]. The sag value is the chordal deviation for curves or surfaces. The curve chordal deviation is the maximum distance between a polyline ("chord") whose end points lie on a curve and a point on this curve, as shown in Figure 3.4. Similarly, the surface chordal deviation is the maximum distance between the tessellated triangles and the surface. A low sag value means that a very fine triangular mesh is used to render surfaces due to the small distance between the geometry and triangles in tessellation. On the other hand, a high sag value means that a very coarse mesh is used due to a high deviation between the geometry and the triangles.

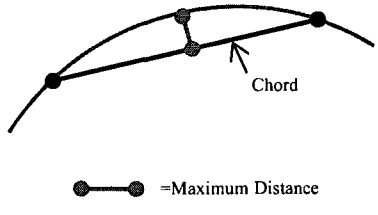
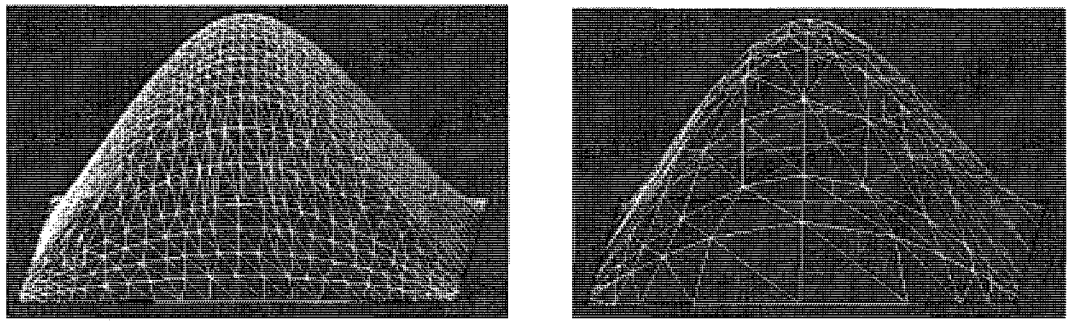


Figure 3.4 Curve chordal deviation

There are two ways to set up the sag values: fixed and proportional to element size. The fixed sag value does not change with the object size, while the sag value proportional to element size varies with the size of the object. That is, for the same sag value, the larger the object, the coarser the tessellation. Figure 3.5 shows the effect of different sag values. The sag value is fixed at 0.20 in Figure 3.5a), and at 8.5 in Figure 3.5b). It is clear that the small sag value results in denser triangles in the model. If you save your model with different sag values, the sizes of the file will not be the same.



a) Model with a fixed sag value of 0.20 b) Model with a fixed sag value of 8.5

Figure 3.5 The effect of sag values

3.2 STL Data Processing

Due to the huge redundancy of data in the STL file, it greatly reduces the computational efficiency if we use the data without pre-processing it. For example, for a part STL file with 20,180 facets, it uses 60,540 vertices (each fact has three vertices). In fact, only 10,086 vertices are not repeated, which means that more than 80% of vertices are redundant. In general, the number of the facets n_f and the necessary number of the vertices n_v have this relationship: $n_f / n_v \approx 2$. The number of redundant vertices is estimated as:

$$n_f \cdot 3 - n_f / 2 \approx 2.5 n_f$$

In a real application, it is very important to take out the redundant vertices from the data, not only because the repeated vertices consumes unnecessary memory and computational time, but also because they could destroy the topological relation between two adjacent facets. Many methods can be used to filter the redundant vertices. Fast redundancy search efficiency can be achieved by sequentially sorting the vertices in all three dimensions and dividing the sorted points into small groups [17]. The much more efficient search method is using hashtable data structure [18]. However, both methods involve relatively complex programming. In this work, a straightforward but efficient method, called Norm method is proposed. The algorithm of norm method is explained as follows.

First, read all vertices from the STL files. Then, calculate the norm of each vertex and sort all the vertices based on the values of their norms. The norm of any vertex *p (x, y, z)* can be calculated by: $Norm_p = \sqrt{x^2 + y^2 + z^2}$. The sorted vertices are stored in a temporary vertex array, called V_{temp} . Meanwhile, define a new array V_{new} to store the filtered vertices. Next, put the first record of V_{temp} in V_{new} and copy it to temporary point variable P_{curr} . After that, check if next record P_{next} of V_{temp} exists. Add P_{next} to the end of V_{new} and set $P_{curr} = P_{next}$, ***only if*** P_{next} exists and meets either of the following two conditions:

- The norm of P_{next} is not equal to that of P_{curr} , or
- The norm of P_{next} is equal to that of P_{curr} , but their coordinates of two points are not identical.

Otherwise, P_{next} is redundant and it will be omitted. Continue the process until reach the end of V_{temp}. The algorithm flowchart is shown in Figure 3.6:

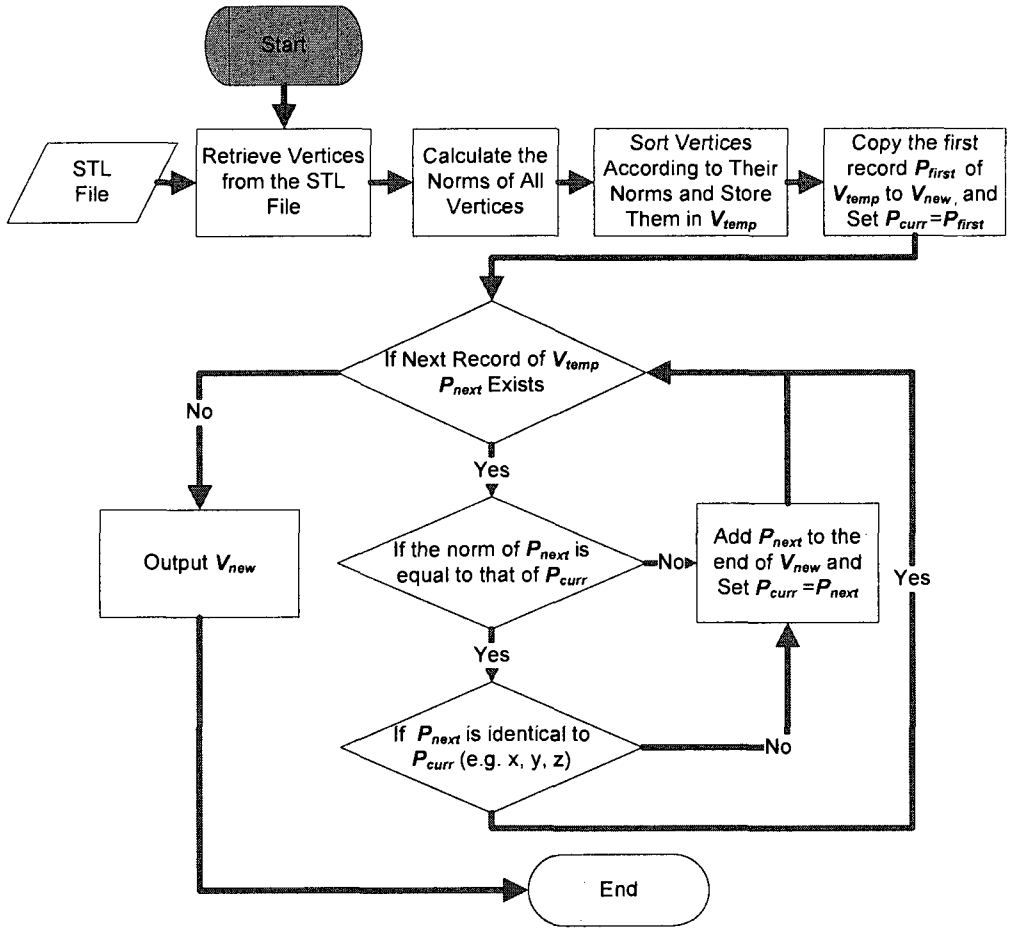


Figure 3.6 Algorithm of filtering the STL redundant vertices

CHAPTER 4. INITIAL CUTTER SIZE DETERMINATION

As mentioned earlier, the search will only focus on the area covered by the cutter shadow or cutter shadow area (refer to Figure 1.3). The question is we do not know the cutter size in advance as it is the final result we are looking for. One way is that we could use the maximum standard cutter size available to define the search range. However, the computational time would be very costly. This is because in most cases we search the area much bigger than needed, which is a huge waste of time. Fortunately, Khan's curvature local gouging detection model can be used to determine the initial cutter size at a CC point[6], which is the maximum cutter size without local gouging. Thus, this initial cutter can be used to define the cutter shadow area. In this chapter, the Khan's model is described first. Next, an algorithm for the initial cutter size determination is proposed.

4.1 Overview of Khan's Model

In Khan's model, he used the comprehensive curvature analysis to detect the local gouging for compound surface patches in 3-axis surface machining. His model provides a set of close-form equations to calculate the normal curvatures along all directions. At a CC point, a local gouging free cutter can be ensured by checking if the curvatures of the cutter are not smaller than that of the surface in any normal direction.

To find the largest allowable cutter size, he tested all standard cutter sizes at hand from big to small until a gouging free cutter is obtained. For each cutter, he compared the curvatures of the cutter with which of the part surface at the engaged area along any tangent directions. If there is no curvature rule violation, this cutter is chosen; otherwise,

this cutter is discarded and a smaller cutter is tested again. The process is repeated until a local gouging-free cutter is found.

In this chapter, the equations related Khan's models are described without detailed derivation. These equations involve part principal curvatures, cutting surface curvatures and gouging check at the engaged area.

4.2 Principal Curvatures for the NURBS Surfaces

The maximum and minimum values of the normal curvatures are known as the principal curvatures at a given point on the surface. The directions in which the curvature takes the maximum and minimum values are called the principal directions of the normal curvatures. The related the equations are provided step by step as follows.

4.2.1 NURBS surface equation

First, let us define a NURBS surface in the part coordinate system as:

$$\mathbf{S}(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}\cdot\mathbf{P}_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)} \qquad (t_{k-1}\le u\le t_{n-1}, t_{l-1}\le v\le t_{m-1}) \quad (4.1)$$

Where, the parameters are defined as:

- $\mathbf{P}_{i,j}$ are the controls points;
- $w_{i,j}$ are their corresponding weights;

- $N_{i,k}(u) = \begin{cases} \dfrac{(u-t_i)N_{i,k-1}(u)}{t_{i+k-1}-t_i} - \dfrac{(t_{i+k}-u)N_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}} & \text{if k>1} \\ \begin{cases} 1 & t_i \le u \le t_{i+1} \\ 0 & \textit{Otherwise} \end{cases} & \text{if k=1} \end{cases}$ is the basis function with an order k;

- $N_{j,l}(v) = \begin{cases} \dfrac{(v-t_j)N_{j,l-1}(v)}{t_{j+l-1}-t_j} - \dfrac{(t_{j+l}-v)N_{j+1,l-1}(v)}{t_{j+l}-t_{j+1}} & \text{if } l>1 \\ \begin{cases} 1 & t_j \le v \le t_{j+1} \\ 0 & \textit{Otherwise} \end{cases} & \text{if } l=1 \end{cases}$ is the basis function with an order l;

- u and v are the parameters of the base functions;

- t_i and t_j are the knot values.

4.2.2 The first and second derivatives of base function

The first derivatives of $\mathbf{S}(u,v)$ in terms of u and v are denoted as $\mathbf{S}_u(u,v)$ and $\mathbf{S}_v(u,v)$, respectively. The second derivative in terms of u is denoted as $\mathbf{S}_{uu}(u,v)$, of v as $\mathbf{S}_{vv}(u,v)$, and of u and v as $\mathbf{S}_{uv}(u,v)$. By differentiating the base function with respect to it parameter (e.g., u), we have the first derivative:

$$N'_{i,k} = (k-1)\left(\frac{N_{i,k-1}}{t_{i+k-1}-t_i} - \frac{N_{i+1,k-1}}{t_{i+k}-t_{i+1}}\right) \tag{4.2}$$

and the second derivative:

$$N''_{i,k} = (k-1)\left(\frac{N'_{i,k-1}}{t_{i+k-1}-t_i} - \frac{N'_{i+1,k-1}}{t_{i+k}-t_{i+1}}\right) \tag{4.3}$$

4.2.3 The first and second derivatives of NURBS surfaces

Let us express the NURBS surface as:

$$S(u,v)=\frac{N(u,v)}{D(u,v)} \tag{4.4}$$

Where,

$$N(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\cdot \mathbf{P}_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)$$

$$D(u,v)=\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\cdot N_{i,k}(u)\cdot N_{j,l}(v)$$

Note, to simplify an expression, occasionally this work uses S, N, and D to represent $S(u,v)$, $N(u,v)$, and $D(u,v)$, respectively.

Then, the first derivatives of the NURBS surface are obtained as:

$$S_u(u,v)=\frac{\partial}{\partial u}\left(N(u,v)/D(u,v)\right)=\left(\frac{N_u(u,v)}{D(u,v)}\right)-\left(\frac{N(u,v)\cdot D_u(u,v)}{D^2(u,v)}\right) \tag{4.5}$$

$$S_v(u,v)=\frac{\partial}{\partial v}\left(N(u,v)/D(u,v)\right)=\left(\frac{N_v(u,v)}{D(u,v)}\right)-\left(\frac{N(u,v)\cdot D_v(u,v)}{D^2(u,v)}\right) \tag{4.6}$$

Where,

- $N_u(u,v)=\frac{\partial}{\partial u}\left[\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\mathbf{P}_{i,j}N_{i,k}(u)N_{j,l}(v)\right]=\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\cdot \mathbf{P}_{i,j}\cdot N'_{i,k}(u)\cdot N_{j,l}(v)$

- $D_u(u,v) = \frac{\partial}{\partial u}\left[\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot N_{i,k}(u) \cdot N_{j,l}(v)\right] = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot N'_{i,k}(u) \cdot N_{j,l}(v)$

- $N_v(u,v) = \frac{\partial}{\partial v}\left[\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}\mathbf{P}_{i,j}N_{i,k}(u)N_{j,l}(v)\right] = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N_{i,k}(u) \cdot N'_{j,l}(v)$

- $D_v(u,v) = \frac{\partial}{\partial v}\left[\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}N_{i,k}(u)N_{j,l}(v)\right] = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot N_{i,k}(u) \cdot N'_{j,l}(v)$

Then, the second derivatives of the NURBS surface are found as:

$$S_{uv}(u,v) = \frac{\partial^2}{\partial u \cdot \partial v}\left(\frac{N}{D}\right) = -\left(\frac{D_v \cdot N_u}{D^2}\right) + \left(\frac{N_{uv}}{D}\right) + 2 \cdot \left(\frac{N \cdot D_v \cdot D_u}{D^3}\right) - \left(\frac{D_{uv} \cdot N}{D^2}\right) - \left(\frac{N_v \cdot D_u}{D^2}\right) \quad (4.7)$$

$$S_{uu}(u,v) = \frac{\partial^2}{\partial u^2}\left(\frac{N}{D}\right) = -\left(2\frac{D_u \cdot N_u}{D^2}\right) + \left(\frac{N_{uu}}{D}\right) + 2\left(\frac{N \cdot D_u \cdot D_u}{D^3}\right) - \left(\frac{D_{uu} \cdot N}{D^2}\right) \quad (4.8)$$

$$S_{vv}(u,v) = \frac{\partial^2}{\partial v^2}\left(\frac{N}{D}\right) = -\left(2\frac{D_v \cdot N_v}{D^2}\right) + \left(\frac{N_{vv}}{D}\right) + 2\left(\frac{N \cdot D_v \cdot D_v}{D^3}\right) - \left(\frac{D_{vv} \cdot N}{D^2}\right) \quad (4.9)$$

Where,

- $N_{uv}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N'_{i,k}(u) \cdot N'_{j,l}(v) = \sum_{i=0}^{n} N'_{i,k}(u)\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N'_{j,l}(v)$
- $D_{uv}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot N'_{i,k}(u) \cdot N'_{j,l}(v) = \sum_{i=0}^{n} N'_{i,k}(u)\sum_{j=0}^{m} w_{i,j} \cdot N'_{j,l}(v)$
- $N_{uu}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N''_{i,k}(u) \cdot N_{j,l}(v) = \sum_{i=0}^{n} N''_{i,k}(u)\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N_{j,l}(v)$
- $D_{uu}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j}N''_{i,k}(u)N_{j,l}(v) = \sum_{i=0}^{n} N''_{i,k}(u)\sum_{j=0}^{m} w_{i,j}N_{j,l}(v)$

- $N_{vv}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N_{i,k}(u) \cdot N''_{j,l}(v) = \sum_{i=0}^{n} N_{i,k}(u) \sum_{j=0}^{m} w_{i,j} \cdot \mathbf{P}_{i,j} \cdot N''_{j,l}(v)$
- $D_{vv}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} N_{i,k}(u) N''_{j,l}(v) = \sum_{i=0}^{n} N_{i,k}(u) \cdot w_{i,j} \sum_{j=0}^{m} N_{j,l}(v)$

The first fundamental matrix of this surface **G** can be defined as:

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{S}_u(u,v)^T \cdot \mathbf{S}_u(u,v) & \mathbf{S}_u(u,v)^T \cdot \mathbf{S}_v(u,v) \\ \mathbf{S}_v(u,v)^T \cdot \mathbf{S}_u(u,v) & \mathbf{S}_v(u,v)^T \cdot \mathbf{S}_v(u,v) \end{bmatrix} \tag{4.10}$$

, and the second fundamental matrix of this surface **D** is defined as:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{n} \cdot \mathbf{S}_{uu}(u,v) & \mathbf{n} \cdot \mathbf{S}_{uv}(u,v) \\ \mathbf{n} \cdot \mathbf{S}_{uv}(u,v) & \mathbf{n} \cdot \mathbf{S}_{vv}(u,v) \end{bmatrix} \tag{4.11}$$

, where the unit surface normal $\mathbf{n} = \left[n_x, n_y, n_z\right]^T = \mathbf{S}_u(u,v) \times \mathbf{S}_v(u,v) \Big/ \sqrt{g_{11} \cdot g_{22} - g_{12}^2}$

4.2.4 Equations of principal curvatures

Let us denote the maximum and the minimum curvatures of the part surface as $k_{s,\max}$ and $k_{s,\min}$, respectively. The Gauss curvature K and mean curvature H can be expressed as:

$$K = k_{s,\max} \cdot k_{s,\min} = \frac{d_{11} \cdot d_{22} - d_{12}^2}{g_{11} \cdot g_{22} - g_{12}^2} \tag{4.12}$$

and

$$H = \frac{1}{2} \cdot (k_{s,\max} + k_{s,\min}) = \frac{1}{2} \cdot \frac{g_{11} \cdot d_{22} - 2 \cdot g_{12} \cdot d_{12} + g_{22} \cdot d_{11}}{g_{11} \cdot g_{22} - g_{12}^2} \tag{4.13}$$

Then, we have $k_{s,\max} = H + \sqrt{H^2 - K}$ and $k_{s,\min} = H - \sqrt{H^2 - K}$. In addition, the principal direction of the maximum curvature $\mathbf{T}_{s,\max}$ can be calculated by:

$$\mathbf{T}_{s,\max} = du_{\max} \cdot \mathbf{S}_u(u,v) + dv_{\max} \cdot \mathbf{S}_v(u,v) \tag{4.14}$$

Where,

$$\begin{bmatrix} du_{\max} \\ dv_{\max} \end{bmatrix} = \begin{bmatrix} k_{s,\max} \cdot g_{21} - d_{21} \\ d_{11} - k_{s,\max} \cdot g_{11} \end{bmatrix}$$

Similarly, the principal direction of the minimum curvature $\mathbf{T}_{s,\min}$ can be determined by:

$$\mathbf{T}_{s,\min} = du_{\min} \cdot \mathbf{S}_u(u,v) + dv_{\min} \cdot \mathbf{S}_v(u,v) \tag{4.15}$$

Where,

$$\begin{bmatrix} du_{\min} \\ dv_{\min} \end{bmatrix} = \begin{bmatrix} k_{s,\min} \cdot g_{21} - d_{21} \\ d_{11} - k_{s,\min} \cdot g_{11} \end{bmatrix}$$

According to Euler's equation, the normal curvature in any tangent direction at an interior surface point can be expressed as:

$$k_s(\beta) = k_{s,\max} \cdot \cos^2 \beta + k_{s,\min} \cdot \sin^2 \beta \qquad \beta \in [0, 2\pi] \tag{4.16}$$

, where angle β represents this tangent direction in terms of direction $\mathbf{T}_{s,\max}$ in counter-clockwise. At a convex point, all normal curvatures are negative; at a concave point, they are positive; and at a saddle point, they are either positive or negative.

4.3 Principal Curvatures and Directions of the Cutting Surface

If $R > r (r \neq 0)$, an APT cutter becomes a bull-nose end mill, which is the most complex and general case, compared to ball end mill and flat end mill. When $R = r$ $(R \neq 0)$ and $r = 0$, it represents a ball end mill and flat end mill respectively. Thus, the model for bull-nose mills is derived first. The models for the other two types of cutters can be easily obtained thereafter. Here, the part coordinate system (*X*-*Y*-*Z*) is assumed to be the reference coordinate system, on which any CC point P_0 on the NURBS surface $\mathbf{S}(u, v)$ lies. Meanwhile, the tool axis is defined as $[0, \quad 0, \quad 1]^T$ in this coordinate system.

4.3.1 Tool coordinate system

In order to find the principal curvatures and principal directions of the cutting surface of the tool, a tool coordinate system is defined for the bull-nose end mill (see Figure 4.1), subject to the following constraints: 1) the tool tip (the center of the bottom circle) is set as the origin of this coordinate system; 2) its z-axis is aligned with the tool axis; 3) its x-axis is perpendicular to its z-axis and is on the plane formed by this z-axis and the surface normal n at this CC point; and 4) its y-axis is the cross-product of these z- and x-axes (see Figure 4.2).

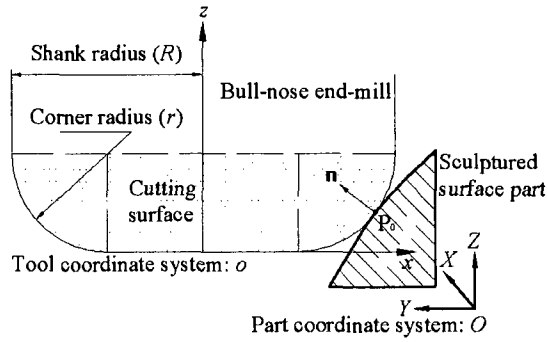


Figure 4.1 Cutting action of bull-nose end-mill in the NURBS surface

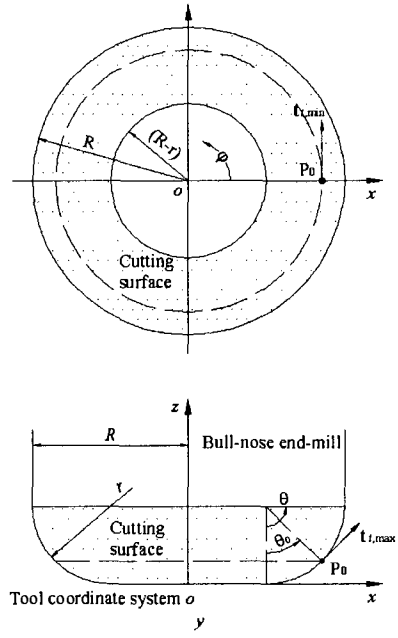


Figure 4.2 Cutting surface of the bull-nose end-mill in the tool coordinate system

4.3.2 Principal curvatures and directions of the cutting surface

- **Bull-nose end mill**

The maximum curvature of this cutting surface at the CC point can be found as: $k_{t,\max} = 1/r$, and its direction $\hat{\mathbf{t}}_{t,\max}$ is $\begin{bmatrix}\cos\theta_0 & 0 & \sin\theta_0\end{bmatrix}^T$ in the tool coordinate system (see Figure 4.2). Meanwhile, the minimum curvature is: $k_{t,\min} = \sin\theta_0/(R-r+r\cdot\sin\theta_0)$, and its direction $\hat{\mathbf{t}}_{t,\min}$ is $\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ in the same system.

- **Ball end mill**

For ball-nose end-mills ($R = r$), the principal curvatures are identical: $k_{\max} = k_{\min} = 1/R$, and the normal curvatures in all directions are all equal to $1/R$ as well.

- **Flat end mill**

By inspection, the maximum curvature of the flat end mill cutting surface at the CC point can be found as $k_{t,\max} = 1/R$, and its direction $\hat{\mathbf{t}}_{t,\max}$ is $[0 \quad 1 \quad 0]^T$ in the tool coordinate system (see Figure 4.2). Meanwhile, the minimum curvature is $k_{t,\min} = 0$, and its direction $\hat{\mathbf{t}}_{t,\min}$ is $[0 \quad 0 \quad 1]^T$ in the same system.

4.4 Gouging Check in All Tangent Directions

Only comparing the principal curvatures of the part surface and the cutting surface is not helpful for local gouging detection due to the fact that their principal directions frequently do not coincide with each other. To remedy this problem, we have to transform the principal curvature directions of the cutting surface from the tool coordinate system to the part coordinate system. Then we can compare the curvatures in all directions to detect any possible gouging at a CC point.

4.4.1 Gouging check between the part surface and bull-nose end mill

The maximum curvature direction $\hat{\mathbf{t}}_{t,\max}$ of the cutting surface can be transformed to the part coordinate system as $\mathbf{T}_{t,\max}$, through the following formula $\mathbf{T}_{t,\max} = [R] \cdot \hat{\mathbf{t}}_{t,\max}$, or

$$\mathbf{T}_{t,\max} = \begin{bmatrix} \left(-n_x \cdot \cos\theta_0\right) \Big/ \sqrt{n_x^2 + n_y^2} \\ \left(-n_y \cdot \cos\theta_0\right) \Big/ \sqrt{n_x^2 + n_y^2} \\ \sin\theta_0 \end{bmatrix} \tag{4.17}$$

Where, $[R] = \dfrac{1}{\sqrt{n_x^2 + n_y^2}} \cdot \begin{bmatrix} -n_x & n_y & 0 \\ -n_y & -n_x & 0 \\ 0 & 0 & \sqrt{n_x^2 + n_y^2} \end{bmatrix}$ and $\hat{\mathbf{t}}_{\text{t,max}} = \begin{bmatrix} \cos\theta_0 \\ 0 \\ \sin\theta_0 \end{bmatrix}$

Similarly, the minimum curvature direction $\hat{\mathbf{t}}_{\text{t,min}}$ of the cutting surface can be transformed to the part coordinate system as $\mathbf{T}_{\text{t,min}}$, through the following formula $\mathbf{T}_{t,\min} = [R] \cdot \hat{\mathbf{t}}_{t,\min}$ *where* $\hat{\mathbf{t}}_{t,\min} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$; that is:

$$\mathbf{T}_{t,\min} = \begin{bmatrix} n_y \Big/ \sqrt{n_x^2 + n_y^2} \\ -n_x \Big/ \sqrt{n_x^2 + n_y^2} \\ 0 \end{bmatrix} \tag{4.18}$$

The values of the principal curvatures remain unchanged in the two coordinate systems. The direction of the maximum curvature of the cutting surface $\mathbf{T}_{t,\max}$ is set as a reference to determine the tangent directions as shown in Figure 4.3. The tangent direction is specified by α, which is the angle measured from the direction of the maximum curvature $\mathbf{T}_{t,\max}$. Here, α_0 is the angle between the directions of the maximum curvatures of these two surfaces, and can be calculated as $\alpha_0 = \arccos\left(\dfrac{\mathbf{T}_{s,\max} \cdot \mathbf{T}_{t,\max}}{\left|\mathbf{T}_{s,\max}\right| \cdot \left|\mathbf{T}_{t,\max}\right|}\right)$, where

$0 \le \alpha_0 \le \pi$. The angle β of a tangent direction is measured from the maximum curvature direction $\mathbf{T}_{s,\max}$ of the part surface

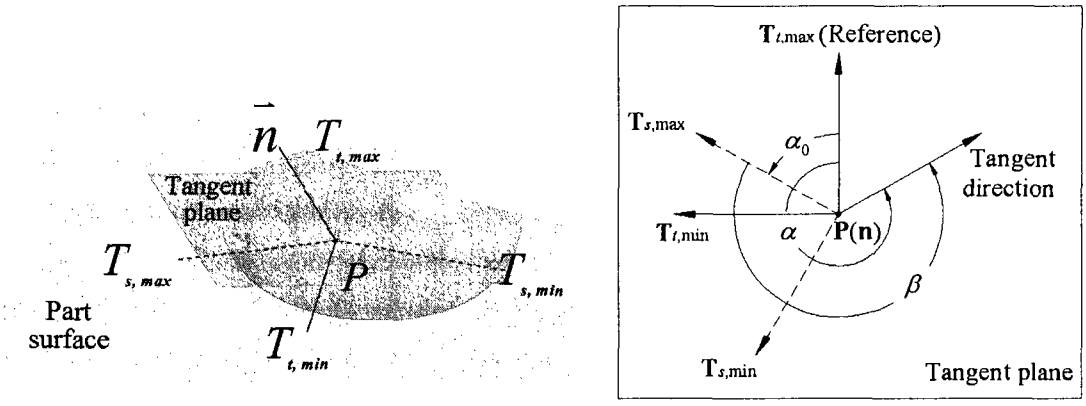


Figure 4.3 Principal directions of the toroidal cutting surface and the part surface

According to Euler's formula, the cutting-surface normal curvature along any tangent direction can be represented as:

$$k_t(\alpha) = \frac{1}{r} \cdot \cos^2 \alpha + \frac{\sin \theta_0}{R - r + r \cdot \sin \theta_0} \cdot \sin^2 \alpha \qquad \alpha \in [0,\ 2\pi] \tag{4.19}$$

Obviously, the relationship between α and β is $\beta = 2\pi + \alpha - \alpha_0$ $(0 \le \alpha \le \alpha_0)$ or $\beta = \alpha - \alpha_0$ $(\alpha_0 < \alpha \le 2\pi)$, and the normal curvature of the part surface in any direction measured from $\mathbf{T}_{t,\max}$ can be calculated as:

$$k_s(\alpha) = k_{s,\max} \cdot \cos^2(\alpha - \alpha_0) + k_{s,\min} \cdot \sin^2(\alpha - \alpha_0) \qquad \alpha \in [0,\ 2\pi] \tag{4.20}$$

If $k_s(\alpha) \leq k_t(\alpha) \quad \alpha \in [0,\ 2\pi]$, then the part surface at the CC point will be gouging free; otherwise, local gouging will occur.

4.4.2 Gouging check between the part surface and flat end mill

In a flat end mill, the cutting surface becomes a planar cutting circle, so the curvature analysis is completely different from that of bull-nose end mill. To conduct the curvature analysis between the cutting circle and the local part surface at the CC point, first, identify the intersection curve between this circle and the part surface; then, calculate the curvature of the this curve at the CC point; finally, compare this curvature with that of the cutting circle, that is, $1/R$ (see Figure 4.4).

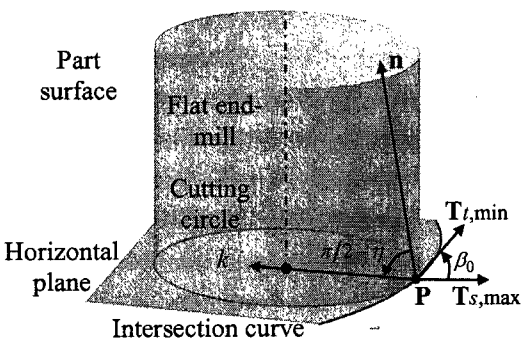


Figure 4.4 Curvature analysis using a flat end-mill

The curvature of the intersection curve is not the same as the normal curvature of the part surface along the tangent direction of this curve. By using Meusnier's theorem, we can calculate the curvature of the intersection curve as

$$k = \frac{1}{\cos\eta}\left(k_{s,\max} \cdot \cos^2 \beta_0 + k_{s,\min} \cdot \sin^2 \beta_0\right) \tag{4.21}$$

, where angle β_0 is the angle between $\mathbf{T}_{s,\max}$ and $\mathbf{T}_{t,\min}$, and angle η is the angle between the tool axis and the surface normal **n** (see Figure 4.4).

If this curvature k is not greater than $1/R$, the local surface is free of gouging; otherwise, it will be gouged.

4.4.3 Gouging check between the part surface and ball end mill

Since the surface curvatures of a ball end mill are the same in all tangent directions, we only need to consider the maximum principal curvature $k_{s,\max}$ of the part. If $k_{s,\max} \leq 1/R$, no local gouging will happen; otherwise, the part surface will be damaged.

4.5 Algorithm for Quick Initial Tool Size Determination

In Khan's model, one needs to know the cutter size before judging if the curvatures between part surface and the cutting surface match or not. In order to find the largest safe cutter, we need to check all the cutter sizes in the standard cutting tool list. However, this would cost a significant amount of computational time as we need to compare the curvatures in all the tangent directions for every cutter in the list. To overcome this problem, an efficient algorithm is proposed to search the maximum possible cutter size at a given CC point, where only few cutter sizes are tested.

In this algorithm, an initial cutter size R_0 (*most likely non-standard size*) is found by setting $k_t(\alpha^*) = k_s(\alpha^*)$ at a specific curvature direction defined by angle α^* (e.g., $\alpha^* = 0$). Then, find the cutter size R_i^* in the tool list, which is the size closest to, but not bigger than R_0. Next, test if the curvatures of this cutter are not smaller than those of the part in all the normal tangent direction. If the test is true, this size is gouging free for this CC point and output this cutter; otherwise, choose the next smaller size in the list, and test it again. Repeat the process until the cutter is gouging free. The algorithm is shown below:

Step 1. Initialize the available cutting tool list:

$$\{R_1, R_2, ..., R_i, ...R_n\}, \text{ where } R_i < R_{i+1}$$

Step 2. Check if $k_t(\alpha = 0) < k_s(\alpha = 0)$. If not, set R=0, and then go to **Step 6**; else go to **Step 3**.

Step 3. Find out the possible cutter size R_0 by setting $k_t(\alpha = 0) = k_s(\alpha = 0)$, and then find the cutter R_{i^*} (i^* is the cutter index in the list) from the list, which is nearest to, but not bigger than R_0

Step 4. Test if R_{i^*} satisfies the condition of $k_t(\alpha) \geq k_s(\alpha)$, where $\alpha \in [0,\ 2\pi]$. If yes, set $R = R_{i^*}$, go to **Step 6**; else go to **Step 5**.

Step 5. If $i^* - 1 > 0$, set $i^* = i^* - 1$ and go to **Step 4;** else, set R=0, and go to **Step 6.**

Step 6. End the search (Return R to main program).

After the maximum local gouging-free cutter is found, this cutter is used as the initial cutter to determine the cutter shadow area.

CHAPTER 5. VERTEX KD-TREE SEARCH

To identify all the points or the triangle vertices falling within the shadow of the cutter (see Figure 1.3), an efficient region query method is needed. The query speed is highly dependent on the efficiency of the data structure. In this work, the KD-tree is chosen as the data structure to identify the vertices or testing points under the cutter shadow, due to its super efficiency in region query[19]. The fundamentals of KD-tree are described, followed by the procedure for final cutter size determination.

5.1 Introduction of KD-Tree

- **Advantages of KD tree**

Multidimensional binary tree or KD-tree was first coined in 1975 in a theoretical paper by Bentley [20], where K is the number of the dimensions in the search space and D refers to dimension. As a special data structure, the KD-tree enables us highly efficiently retrieve the data with the required conditions using multidimensional search keys. According to Bentley, a typical insertion and record lookup in a KD-tree with a size of n records will examine about $1.386\log_2 n$ nodes, which is very efficient. The big advantage of this structure is that a single data structure can be used to perform many types of queries at a very fast speed, such as nearest point search, region query, fixed-radius near neighbour search [21, 22]

Among three typical data structures, namely linked list, adaptive quad tree, and KD-tree, the KD tree offers the best performance in region query, though it requires the largest memory. On the other hand, the linked list has worst efficiency in region query, yet it

uses the smallest memory. The quad tree falls in between the KD-tree and the quad tree regarding both querying efficiency and memory usage. In our case, the main concern is the computational time and the memory usage is not a problem.

- **Applications of KD-tree**

Binary search tree is a special case of KD-tree, where we can only find the items within certain range in one dimension. However, for items with multiple keys with each key corresponding to a different dimension, the KD-tree would be an appropriate data structure to serve this purpose. For example, given a set of points in three spaces (K=3), one can quickly find all the points whose X, Y, and Z coordinates satisfying the specified criteria:

$$\begin{cases} X_{\min} \le X \le X_{\max} \\ Y_{\min} \le Y \le Y_{\max} \\ Z_{\min} \le Z \le Z_{\max} \end{cases}$$

The application of KD-tree is not limited to geometric problems. As a typical example, one may wish to find all employees whose salary is between $12/hour and 18$/hour, whose family has three to five children, and who have worked more than five years. The salaries could be viewed as points along the X axis, the number of children as points along the Y axis, and the years as points along the Z axis.

5.2 KD-Tree Data Structure

5.2.1 Components of KD-tree

According to the Bentley's theory, in a KD-tree, each record of a given file is stored in a form of a node. Typically, a node includes the following components:

- **K keys**, called K_0 (P), K_1 (P),..., K_{k-1}(P), where P is any node. These keys represent the values of P at different dimensions, e.g. X, Y, and Z coordinates of P if P is a point in 3-D space.
- **Two pointers**: LOSON (P) and HISON (P), which are either null or point to another node in next level in the KD-tree. (Note that each pointer can be considered as specifying a sub-tree)
- A **discriminator** DISC (P) (or splitting dimension) associated with each node. The discriminator is an integer between 0 and k – 1 inclusively
- **A splitting value** K_J(P) at which the data set is cut into two subsets such that the points in the LOSON will be not greater than it, and the points in the HISON will be not less than it against its corresponding dimension (J=DISC(P)).
- **A boundary array** B, which bounds the node position by certain values. B has 2k entries: *B(0), B(1),...,B(k-1)*, corresponding the maximum and minimum boundaries in each dimension.

5.2.2 Construction procedure

To build a KD tree, the common construction procedure involves the below steps:

- First, index all the dimensions by 0, 1, 2...K-1 (e.g. for a 3D tree or K=3, its dimensions are represented by index 0, 1, and 2, which corresponds to X, Y, and Z axis, respectively). These indices are called as discriminators.
- Then, select a splitting plane (or line if in 2-D space) perpendicular to *discriminator **0*** (e.g. X axis). This plane cuts the entire points into two subsets

(called ***LOSON or HISON***), each of which is split into two subsets using a splitting plane perpendicular to *discriminator 1* (e.g. Y axis).

- The cutting action continues until the *discriminator* reaches ***K-1***(e.g. Z axis) at which time the *discriminator* to which the splitting plane will be perpendicular is reset to ***0*** (e.g. X axis).
- The process repeats in the same fashion until the size of the point set is small enough to meet certain criterion.

To guarantee a balanced tree or the optimal search efficiency, at each internal node, the splitting plane or line has to pass the median value of the points in its corresponding dimension. Specifically, suppose we have n points that are sorted in a given dimension, for example, such that $X_i < X_i+1$ (i = 0, …, n – 2). Let m =integer portion of (n-1)/2.Then the vertical splitting plane is at $X = X_m$ or $(X_m + X_{m+1})/2$, depending on the type of KD-tree. To implement this, the points should be stored in K sorted vectors in all the dimensions (e.g. X, Y, Z…) before the splitting operation proceeds, for example, for a three dimensional KD-tree, three sorted vectors are needed for points: one ordered by x coordinates, one ordered by y coordinates, and one ordered by z coordinates.

5.2.3 Main properties of KD-tree

- **Discriminators at different levels**

At a given level i of the tree, all the nodes use the same discriminator. Generally, to calculate the discriminator for next level, a function NEXTDISC can be defined as:

NEXTDISC(i)=(i+1) mod k

Also, if the two sons are non-null, we have:

NEXTDISC(DISC(P)) = DISC(LOSON(P))= DISC(HISON(P))

- **Criteria for searching directions**

For any node P, let j be DISC (P).A function SUCCESSOR (P, Q) is defined to tell which son of P to visit while searching for node Q. This function returns either LOSON or HISON. As mentioned before, we can easily identify which son to go if $K_j(Q)$ and $K_j(P)$ are not equal, according to the property of KD-tree. However, if the two keys are the same, a superkey of P has to be defined as a product of all the keys (Note: the key starts with by K_j and cyclically concatenates with each other. The superkey of P is expressed as:

$$S_j(P) = k_j(P) \cdot k_{j+1}(P)...k_{k-1}(P) \cdot k_0(P) \cdot ...k_{j-2}(P) \cdot k_{j-1}(P)$$

By comparing the superkeys, three cases are identified:

- If $S_j(Q) < S_j(P)$, it returns LOSON;
- if $Sj(Q) > Sj(P)$, it returns HISON.
- If $Sj(Q) = Sj(P)$ it indicates that the two nodes are identical.

- **Property of boundary box**

If Q is a descendant of P, then:

$$B(2j) \le K_j(Q) \le B(2j+1) \text{, where } j \in [0, k-1]$$

5.3 Types of KD-Tree

There are many variations of KD-trees. However, here only two typical ones will be presented. In this work, the second type of KD-tree is employed as the data structure.

The first version is the earliest one, which was described by Bentley in his KD-tree theory. In this version, every node of a KD-tree, from the root to the nodes before their leaves, stores a point. Leaf nodes, called null nodes, store no point. As a result, each splitting plane or line must go through one of the points in the tree. An example is shown in Figure 5.1, where the data or records are stored as nodes in 2D space. Figure 5.2 is planar graph form of the same 2-D tree. Note that LOSONs are represented by the left branches, HISONs by the right branches, and null sons by boxes.

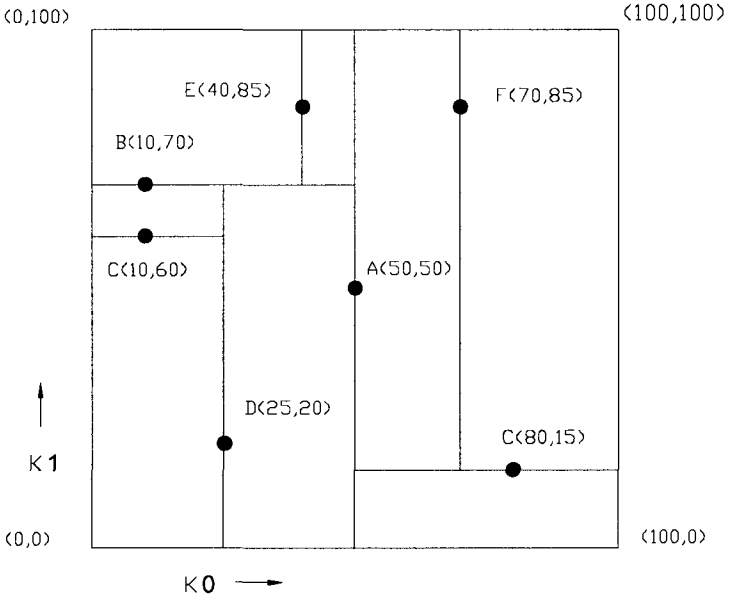


Figure 5.1 2D tree in box form (boxes represent range of sub-tree)

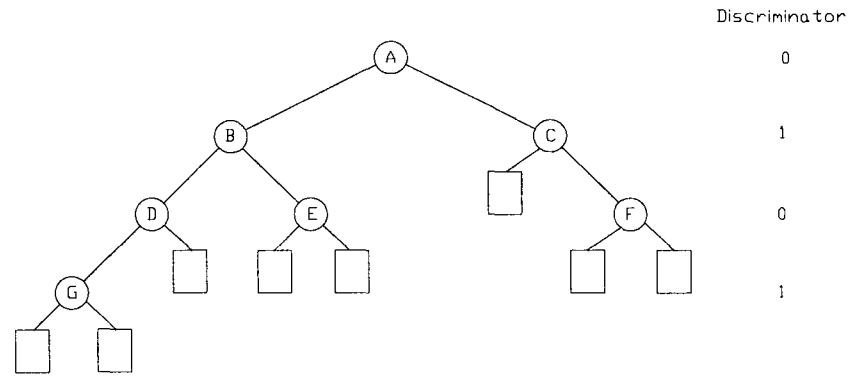


Figure 5.2 Planar graph representation of the same 2-D tree

Opposite to the first version, the second type of KD-tree stores the points in its leaf nodes only [23]. Note that each splitting plane or line may still go through one of the points and each leaf node stores as least one point. The structure of this type is illustrated in Figure 5.3. The left side of the figure shows the given points(*P1, P2, ..., P9*) and the subdivision of the data set with splitting lines(*l1, l2, ..., l9*), and the right side illustrates the corresponding KD-tree. Interior nodes are marked by circles, and leaf nodes, which contain individual points, are marked by squares. Each node contains the information of its corresponding rectangular region.

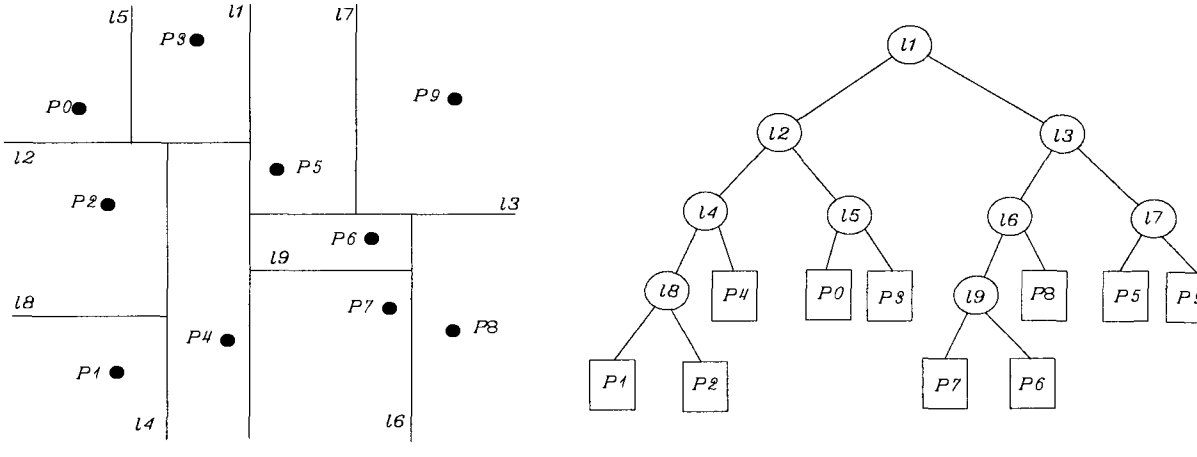


Figure 5.3 Second version of KD–Tree

5.4 KD-Tree Algorithm

The major procedures for KD-tree involve KD-tree creation, and region query. The corresponding algorithms will be presented in the following sections:

- **Creating KD-tree**

The following is the algorithm for the KD-tree creation, which is combination of knowledge from [20, 23]. This work only concerns the points in 3D space, so K=3. The discriminator values 0, 1, and 2 correspond to X, Y, and Z axis, respectively. Here P and B represent input data points and boundary array of these points, correspondingly. C is the maximum size of points contained in a leaf node.

Create KDTree (P, discriminator, B, C)

{

if (the size of P $\leq$ C), then return a leaf node T storing P and the node rectangle B

else if (discriminator=0)

{

Then divide P into two halves with a plane Φ passing through the median X-coordinate of the points in P and perpendicular to X axis.

Let P1 be the set of points to the left of Φ,

and let P2 be the set of points to the right of or on Φ.

Let B1 be the part of B to the left of Φ,

and let B2 be the part of B to the right of Φ.

}

else if (discriminator=1)

{

Then divide P into two halves with a plane Φ passing through the median Y-coordinate of the points in P and perpendicular to Y axis.

Let P1 be the set of points below Φ,

and let P2 be the set of points above or on Φ.

Let B1 be the part of B below Φ,

and let B2 be the part of B above Φ.

}

else

{

Then divide P into two halves with a plane Φ passing through the median Z-coordinate of the points in P and perpendicular to Z axis.

Let P1 be the set of points behind Φ,

and let P2 be the set of points before or on Φ.

Let B1 be the part of B behind Φ,

and let B2 be the part of B before Φ.

}

Tleft= Create KdTree (P1, (discriminator + 1)%K, B1, C)

Tright = Create KdTree (P2, (discriminator + 1)%K, B2, C)

Store pl, Tleft, Tright, and the node array B in a node T

(Tleft is the left child of current nodeT, and Tright is the right child of T.)

return T

}

- **Querying the KD-tree**

A region query searches the points of a KD-tree within a cubic region R limited by minimum and maximum values of X, Y, and Z and provides the results to the user. It identifies all points under the node T falling in region R. The algorithm is shown below:

```
Region_Search_KdTree (T, R)
{
    if (T is a leaf)
    {
        then report the points stored at T whose coordinates are contained in R
    }
    else if (node cubic (Tleft) is fully contained in R)
    {
        then return the points in Tleft
    }
    else if (node cubic (Tleft) intersects R)
    {
        then Region_Search_ KdTree (Tleft, R)
    }
    if (node rectangle(Tright) is fully contained in R)
```

```
        {
            then return the points in (Tright)
        }
        else if (node rectangle(Tright) intersects R)
        {
            then Region_Search_KdTree (Tright, R)
        }
}
```

5.5 Determination of the Cutter Size

As previously mentioned, to find all the testing points or vertices by KD-Tree search, we have to specify the criteria for their X, Y, and Z coordinates in a cubic form:

$$\begin{cases} X_{\min} \le X \le X_{\max} \\ Y_{\min} \le Y \le Y_{\max} \\ Z_{\min} \le Z \le Z_{\max} \end{cases} \quad where, \quad \begin{cases} X_{min} \text{ and } X_{max} \text{ are the lower and upper limits for } X \\ Y_{min} \text{ and } Y_{max} \text{ are the lower and upper limits for } Y \\ Z_{min} \text{ and } Z_{max} \text{ are the lower and upper limits for } Z \end{cases}$$

However, the shadow of a cutter is a circle (refer to Figure 1.3), so we have to transform the circle into a rectangle. Figure 5.4 illustrates how to define the search range for X and Y axis, based on the cutter shadow. Here, the circle centered at ***O'*** represents the cutter shadow with a radius $\boldsymbol{R_s}$. If we know the $X_{o'}$ and $Y_{o'}$ (the coordinates of ***O'***), the lower and upper limits for X and Y can be defined as:

$$\begin{cases} X_{\min} = X_{o'} - R_s \text{ and } X_{\max} = X_{o'} + R_s \\ Y_{\min} = Y_{o'} - R_s \text{ and } Y_{\max} = Y_{o'} + R_s \end{cases}$$

For a given APT cutter with a radius R, corner radius r, and taper angle θ, the radius R_s can be calculated by $R_s = R + r \cdot \cos\theta + H \cdot \tan\theta$, where ***H'*** is the length of the cutter

tapered portion. Since the conical angle usually is very small or zero, we can assume it is zero. Thereby, ***H'*** is determined by $H' \approx H - r(1-\sin\theta)$ (refer to Figure 2.1).

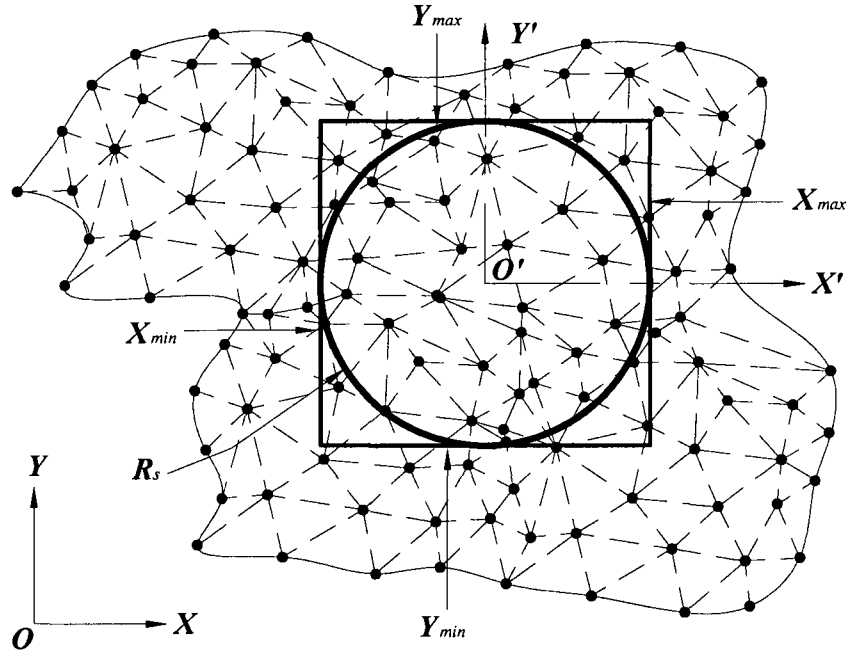


Figure 5.4 The search range determination

Given a CC point and its normal N_{cc}, $X_{o'}$ and $Y_{o'}$ can be easily obtained by Liu's model or the refined model for the case where the N_{cc} only has Z component. Meanwhile, from all the vertices extracted from the part STL file, we can find the minimum value Z'_{min} and the maximum value Z'_{max} of all those vertex points. Therefore, the lower and upper limits for Z coordinate are expressed as: $\begin{cases} Z_{min} = Z'_{min} \\ Z_{max} = Z'_{max} \end{cases}$.

Using the above criteria, the KD-Tree search algorithm can quickly identify all the vertices falling within the cutter shadow area. Then, through the imaginary cutter model, a group of imaginary cutters can be obtained by using the vertices as testing points. The

final cutter size or the maximum allowable cutter size for the given CC point can be easily found by selecting the minimum cutter from the imaginary cutter group.

CHAPTER 6. INTEGRATED KD-TREE CUTTER SIZE SEARCH MODEL

The local gouging detection method is capable of providing an upper bound of the cutter size for the vertex KD-tree cutter size search and ensuring that any size smaller than this bound is local gouging-free. The interference or global gouging check, which is not able to be solved by Khan's gouging detection model, is taken care by the vertex KD-tree search. Thus, integration of the two methods is an ideal choice. The integrated model is called iKD-tree model, and its structure and implementation are discussed as follows.

6.1 Structure of the Integrated Model

The integrated model can be organized into three main modules, namely Data Preparation, Cutter Search Engine, and Data Postprocessing. The flowchart of the model is shown in Figure 6.1. The details of these three modules are described in the following sections.

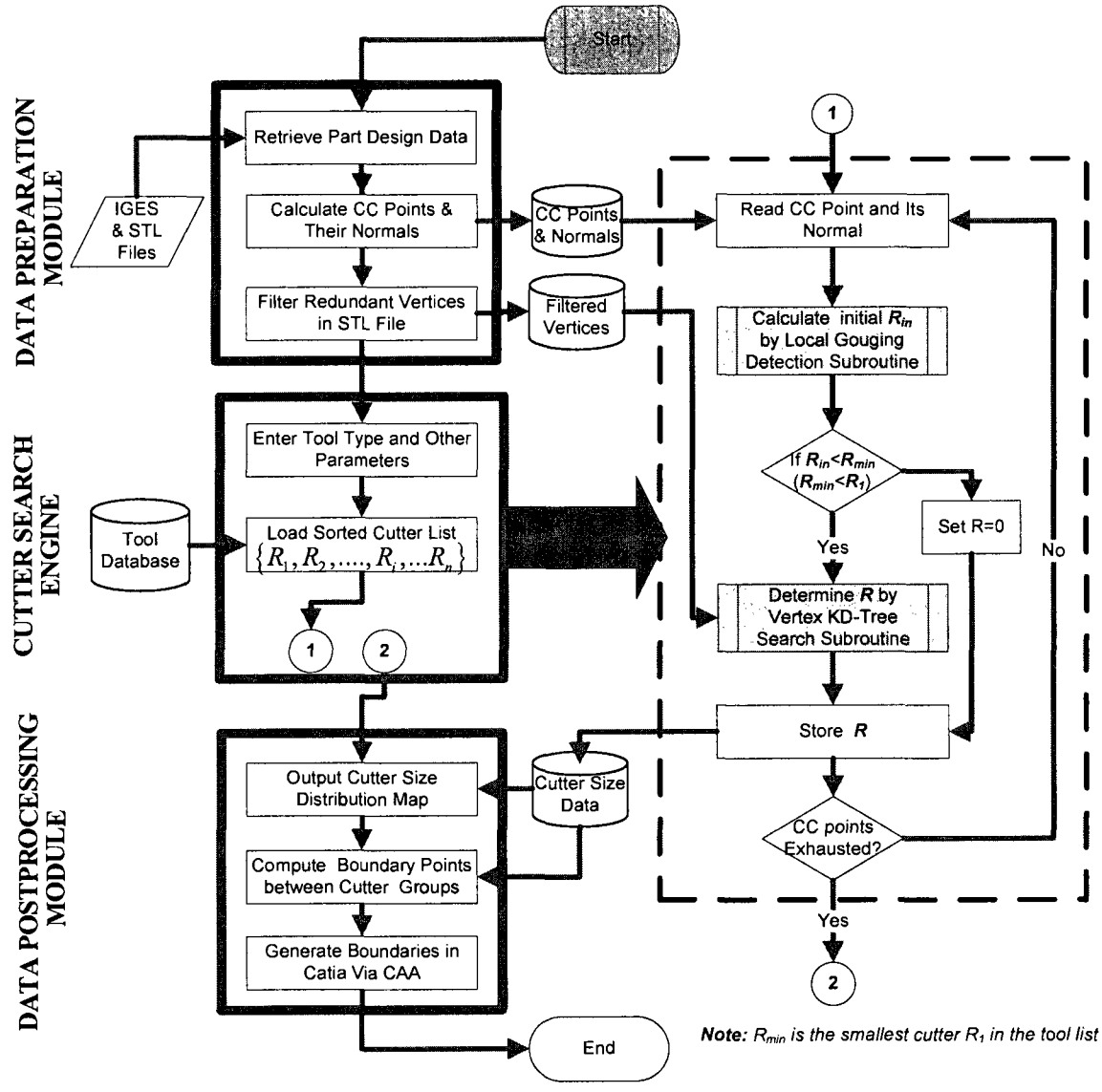


Figure 6.1 Flowchart of the integrated model

6.1.1 Data preparation module

In data preparation module, the NURBS surface equations and other parameters are retrieved from the part IGES file, which will be used to generate CC points, their normals, and the boundary information between two adjacent surface patches. Meanwhile, the vertices of triangles are read from the part STL file. Then vertices are filtered to remove

all the redundant points, which will be used as the testing points to determine the cutter size in the vertex KD-tree search.

6.1.2 Cutter search engine module

Cutter search engine module is the core of the integrated model, which performs the initial cutter search and the final cutter radius determination. In the beginning, the tool type and other parameters (i.e., corner radius) are prompted to enter the system. Then, the cutter list of the specified type is loaded into the system. After that, the initial cutter size R_{in} is determined by the local gouging detection subroutine. The flowchart of the local gouging detection subroutine is shown in Figure 6.2. Using the obtained initial cutter size, the system performs the vertex KD-tree cutter size search to locate the final largest possible cutter size R for each CC point. The subroutine flowchart of the vertex KD-tree search is shown Figure 6.3.

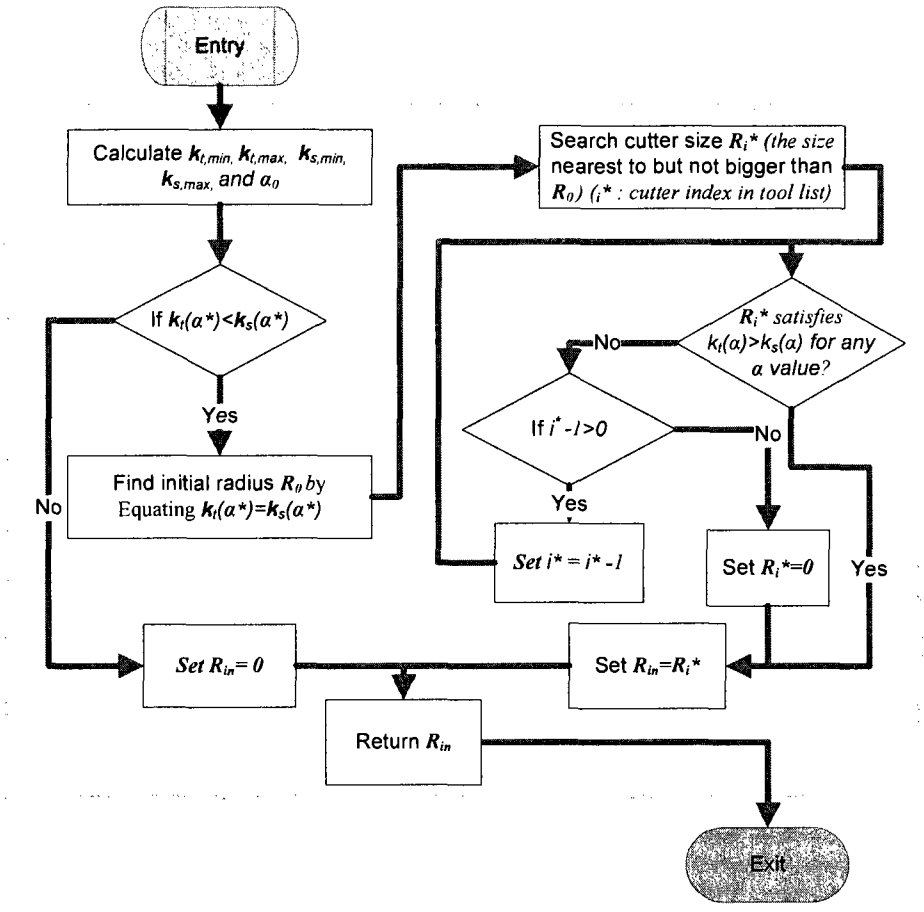


Figure 6.2 Flowchart of local gouging detection subroutine

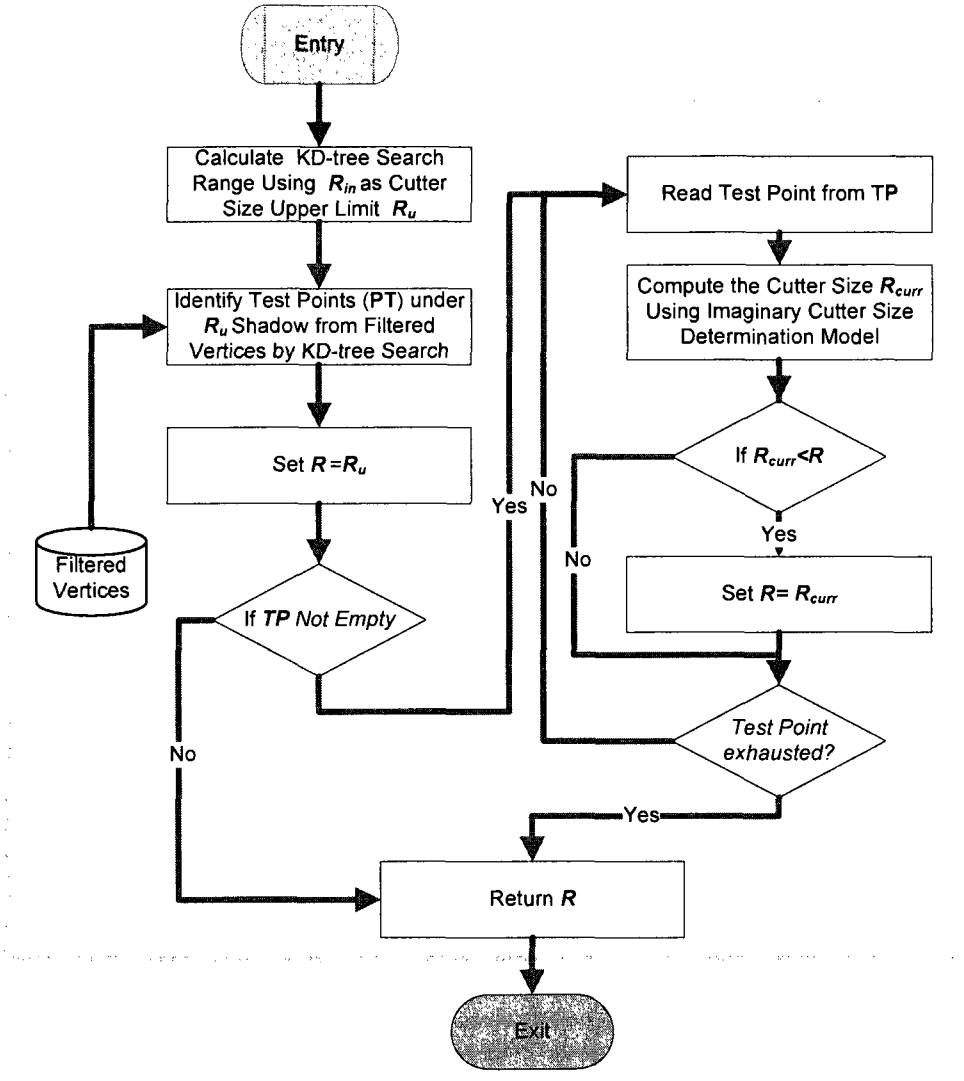


Figure 6.3 Flowchart of vertex KD-Tree search subroutine

6.1.3 Data postproccessing module

The main functions of data postprocessing Module are to present cutter size distribution in a color map form for a quick visual check, and to generate the cutter size boundaries in the CAD/CAM software CATIA for the machining process planning. The concepts of color cutter map and cuter size boundary are described as follows.

- **Color cutter map**

The standard cutter distribution can be effectively represented by a color cutter map. In the map, the color at a given CC point indicates the largest standard cutter that can be used at that point. The CC points with the same color forms a region that can be cut by the same cutter size. The procedure of constructing a cutter map is described as follows:

- Build a standard cutter list according to the all the cutter available
- Assign different colors to different cutter sizes in the list
- Calculate the allowable cutter at each CC point using the integrated model
- Find the closest standard cuter in the list for each allowable cutter, but it should not be bigger than the allowable cutter.
- Plot the cutter size at each CC point using the corresponding color to obtain the cutter map.

Using the above method, we can obtain a combination of standard candidate cutters that can be used to machine the part with the least machining time. Obviously, this standard cutter group can be used to cut the part without gouging and interference due to the fact that each of them is not bigger than the corresponding imaginary cutter. From the cutter map, one can roughly identify the largest possible standard cutter size that can be employed at each surface region. In some case, when the surface curvature is very large and the open space at an area is very small, one can clearly see if there is a standard cutter can access that area using the cutter map. If not, a design change may be required.

- **Generating the cutter size boundary in CATIA**

As mentioned earlier, in the part surface, each cutter forms a cutter size region at which the highest cutting efficiency can be achieved by this cutter. Between any two cutter size

regions, there exists a boundary. The cutter map is generated by the MATLAB program, and it is just a rough presentation of the cutter size regions. As a result, there is not boundary in a cutter map that we can directly use to divide the compound surface into different regions for the corresponding cutter sizes. Furthermore, even if there is a boundary in the map, it is usually not cost-effective to plan machining process on the map. If we can generate these boundaries in the part CATIA design, we could perform machining process planning in CATIA by assigning the largest standard cutter size for each region, which is the most efficient way for material removal. Since whenever there is standard cutter size jump between two CC points, we can always find out the parametric middle point, whose u and v are the u average and the v average of the two CC points, respectively. Using the u and v of the parametric middle point, we can obtain any boundary Curves between any two neighbouring cutter regions in CATIA by the following steps:

- First calculate all these middle points through their u and v values.
- Then output the points obtained in the first step to CATIA.
- Finally connect those points by B-Spline Curves.

6.2 Implementation of Integrated Model

To transform the model into real applications, the model is implemented in MATLAB and a snapshot of the user interface is shown in Figure 6.4. The system requires the user to input the surface patch information and the cutter parameters. In addition, the user has to load the STL file of the part before computing. If the user wants to generate the cutter

size boundary in CATIA, he or she has to make the CATIA Generative Shape Design Workbench ready before pressing the button.

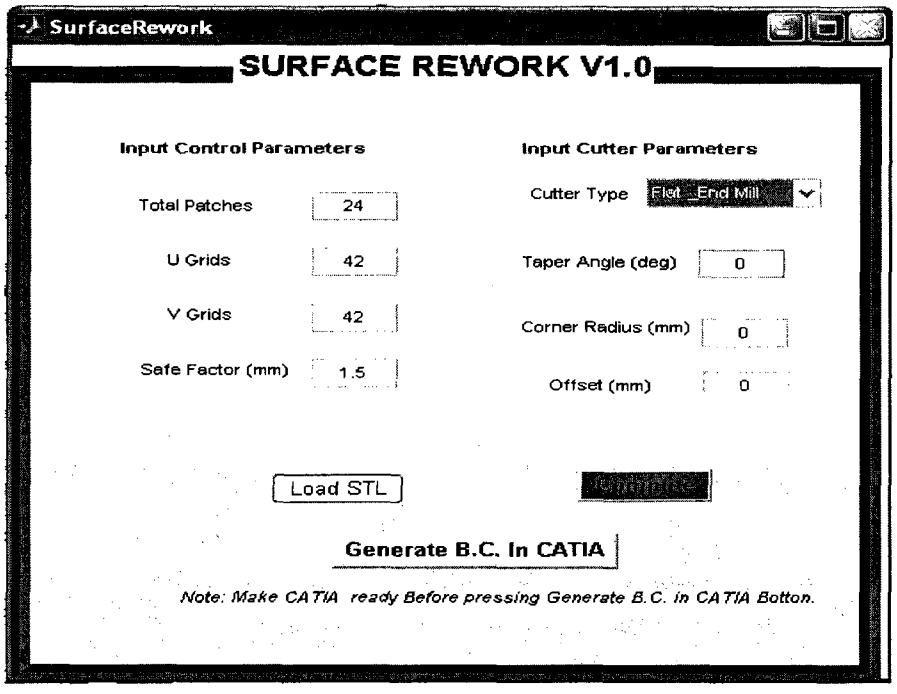


Figure 6.4 The interface of the system

CHAPTER 7. APPLICATION AND COMPARISON

7.1 Introduction

Compound surfaces are widely used in the parts with a complex shape, for instance, dies/moulds. Normally, a compound surface consists of a number of free-form surfaces connected with C^1 or/and C^2 continuity. When we plan the machining processes, we always want to use the largest possible, yet gouging-free cutter for at a given surface region. In other words, the curvature of the cutter should match that of the surface as much as possible. Thereby, we can maintain the highest material removal rate. To cut a compound surface on 3-axis CNC machines, using only one cutter to machine all the surfaces is not efficient. This is not only because the surface curvature changes from CC point to CC point, but also because the open space for the cutter varies from location to location. As a result, this would be far away from the rule of curvature match.

Ideally, all imaginary cutters calculated from the model are the perfect candidate sizes to cut the compound surface. However, most of the imaginary cutters are non-standard and it would be very costly to make them. As a result, all the imaginary cutters will be converted into standard cutters as the output in this work. The conversion is simply a process of finding the available standard cutter size closest to the imaginary cutter.

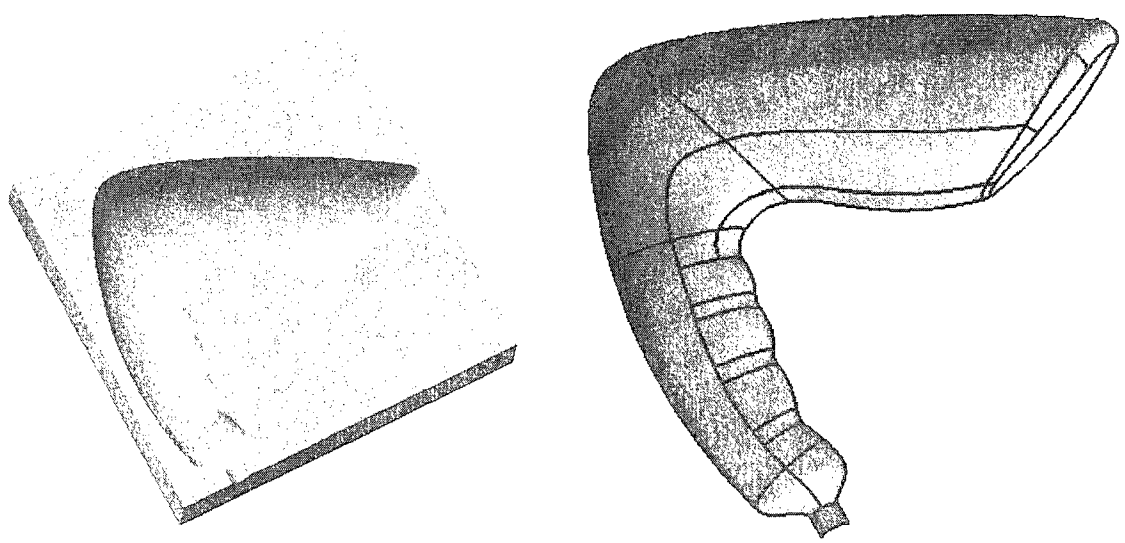
In this chapter, a specific application of hairdryer is introduced. Then, comparisons between iKD-Tree method, Liu's PSO cutter search method, and CATIA rework function

are made to show the advantages of the proposed model, regarding the computational time, accuracy, and cutter selection efficiency.

7.2 Application of a Hairdryer Mould

7.2.1 Part CATIA design

Due to its complexity, this work still uses the hairdryer mould as an example, which is the same part design that Liu used in his PSO cutter size method. This hairdryer mould consists of 24 surface patches, which are connected with C^0, C^1 or C^2 continuity. The CATIA design of this part and its 24 patches are shown in Figure 7.1. This hairdryer surfaces are very complex as it has both concave and convex regions. It is extremely difficult to determine the combination of cutter sizes to realize the maximum production efficiency, while ensuring the no gouging and interference happen.



(a) (b)

Figure 7.1 (a) CATIA design of the hair dryer mould and (b) Its 24 surface patches

7.2.2 Surface patch parameters and cutter data

In this work, for each surface patch, a mesh of 21×21 iso-parametric CC points is used. Since there are 24 patches for the whole hairdryer, there are 10584 CC points in total. Assume the standard cutter sizes available are 25.4 mm (1 inch), 12.7 mm (1/2 inch), 6.35 mm (1/4 inch) and 3.175 mm (1/8 inch) for both flat end-mill and ball end-mill. The STL file is generated in CATIA by setting the 3D accuracy and 2D accuracy to be .01, which is the finest setting in this software. Unless mentioned otherwise, these parameter settings are used throughout this chapter to run the integrate model system. The results are described in the following sections.

7.3 Color Cutter Map of Hairdryer

As previously mentioned, the results of cutter sizes are represented by the color cutter maps. By running the system with the above parameter settings, the cutter maps of flat end-mill and ball end-mill are shown in Figure 7.2 and Figure 7.3, respectively. In the two cutter maps, different colors signify the different cutter radius regions:

- dark brown: 1 inch
- light brown: ½ inch
- green: ¼ inch
- light blue: ⅛ inch
- dark blue: cutter not available

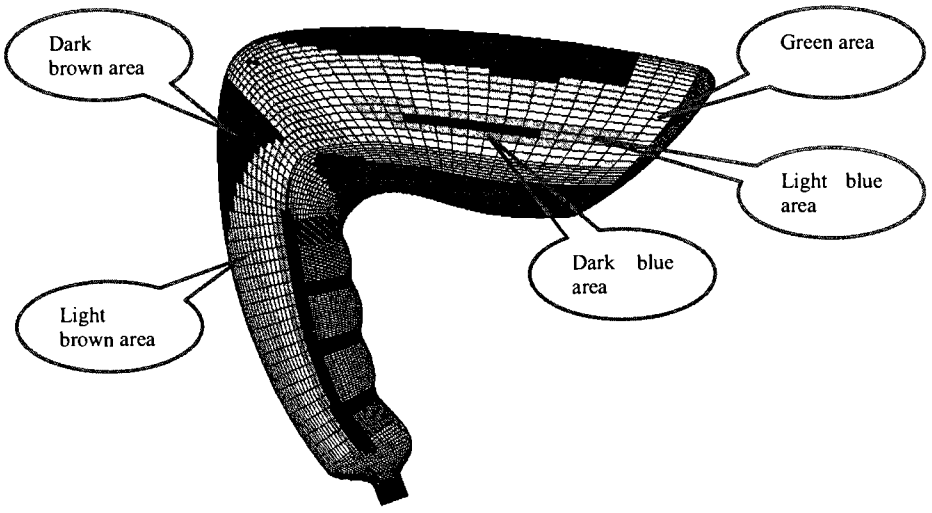


Figure 7.2 Standard cutter radii map for flat end-mills

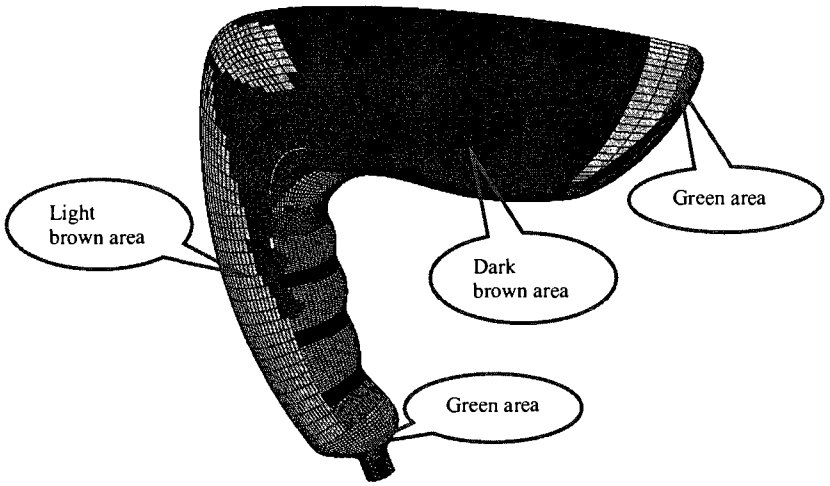


Figure 7.3 Standard cutter radii map for ball end-mills

The color cutter maps are generated by MATLAB program. By visual inspection, a rough cutter distribtuion can be obtained. The small dark blue area in Figure 7.2 indicates that no flat cutter can access that area.Therefore, to cut this area, one option is to use a smaller

flat-mill cutter or other type of cutter. Another option is to modify the design of that region so that the available flat end-mill cutter can access it.

7.4 Comparison between PSO Method and iKD-Tree Method

In order to demonstrate the computational efficiency and cutter size accuracy, this work uses both the Liu's PSO method and iKD-Tree method to search the cutter size combination for the hairdryer mould. The results from both methods are compared.

7.4.1 Computational time

As expected, the iKD-Tree cutter search method is much more efficient than the PSO cutter search method in terms of cutter size search speed. This is not only because the iKD-Tree method greatly narrows its searching scope to the area covered by the cutter shadow, but also because the search is on the limited discrete points of that area, while the PSO method has to continuously search entire compound surfaces. Indeed, the efficiency of iKD-Tree is proved by the hairdryer mould example.

To demonstrate the computational efficiency, the same settings are employed as previously mentioned, except the cutter type. This time, only torus end-mill, the most complex cutter among the three types of APT cutters, is selected, and assume that it has a corner radius of 1mm and a taper angle of 2^0. In the part STL file, there are 21931 vertices. The time required to compute all cutter sizes of all the CC points is 4.2636 hours, which is only 18% of the time that are required by the PSO method to do the same job. Please keep in mind that in this work, the program is coded by MATLAB, which is much

slower than the other programming languages, e.g. C++. Therefore, this is really a huge improvement on the search efficiency. The detailed data are shown in Table 7.1.

Table 7.1 Computational time comparisons

Methods	Total CC Points	Computational Time(hr)
iKD-Tree	10,584	4.2636
PSO	10,584	23.4646

7.4.2 Cutter size accuracy

Generally speaking, the PSO cutter search method has a very high accuracy in determining the cutter size due to its continuous search nature. Therefore, a quick and simple way to check the accuracy of the proposed method is to compare the cutter sizes obtained from the iKD-Tree method with which obtained from the PSO method under the same conditions. From the results, it is found that the cutter sizes computed by both methods are identical for 99.5% of the CC points. Among the 10,584 CC points, only 50 CC points have slightly cutter size differences between two methods. The deviations are mostly about 1 mm. Please keep in mind that those values are imaginary cutter sizes. In reality, we need to convert them into the corresponding standard cutter sizes.

Suppose that the available standard cutter sizes are *(1) 3.175, (2) 6.35, (3) 8.0, (4) 10.0, (5) 12.7, (6) 14.0, (7) 15.0, (8) 16.0, (9) 20.0, (10) 25.4, (11) 30.0, (12) 32.0,* and *(13) 40.0*. Those cutter sizes are in mm and listed in the ascending order. After converting all the cutter sizes into the standard cutter sizes, the CC points that have different cutter sizes between two methods drop to 10 (see Table 7.2). From Table 7.2, it is seen that the

9 CC points have a cutter size index difference of 1, and only one CC point has a cutter size index difference of 2. That is being said, the difference is insignificant.

Table 7.2 Different cutter sizes between PSO method and iKD-Tree method

No. of CC Point	PSO Method		iKD-Tree Method		Difference of Index
	Standard size in mm	Index in the cutter list	Standard size in mm	Index in the cutter list	
3,582	3.175	(1)	6.35	(2)	1
3,902	0	(0)	3.175	(1)	1
4,618	3.175	(1)	6.35	(2)	1
5,259	10	(4)	12.7	(5)	1
5,260	12.7	(5)	14	(6)	1
5,261	12.7	(5)	16	(7)	1
5,262	12.7	(5)	16	(7)	1
5,263	14	(6)	16	(7)	1
5,264	14	(6)	20	(8)	2
10,107	0	(0)	3.175	(1)	1
10,108	0	(0)	3.175	(1)	1
10,113	6.35	(2)	8	(3)	1
10,115	10	(4)	12.7	(5)	1
10,116	14	(6)	16	(7)	1
10,117	16	(7)	20	(8)	1

***Note**: Index 0 means the cutter size is not available*

Further investigation reveals that all the 10 CC points with different cutters between two methods are from the patch boundaries. The main reason is because the local gouging detection approach is only valid under the assumption that all the patches are connected with C^2 continuity. In this hairdryer mould design, not all of the patches are connected with C^2 continuity. Thus, if a better local gouging detection model that can apply to the surfaces connected with C^1 continuity is provided, we could completely solve the problem.

After converting all the imaginary cutter sizes into standard ones, the result matching between two methods becomes to 99.9% of CC points. On the other hand, there is no guarantee that the PSO method will produce 100% of correct results due to its stochastic nature. Therefore, the accuracy of iKD-Tree method is comparable to that of the PSO method, and is acceptable.

7.5 Comparison between iKD-Tree Method and CATIA Rework Function

In CATIA V5, there is machining rework function, which can be used to determine the rework boundaries that cannot be accessed by a given cutting tool. If we know all the possible candidate cutters that can be used to cut a part, we can also identify the boundaries between any cutter size regions by using the CATIA rework function. In order to give the reader a clear picture about the strength of this work and the limitation of the CATIA rework function, a comparison between them is provided by assuming all the cutter size are given for CATIA rework.

7.5.1 Efficiency of cutter size selection

In reality, it is almost impossible to guess all the candidate cutter sizes that can be used to cut a part by experience, especially when the part surface becomes very complex. If you want to test all the available cutters in the list to obtain the boundaries of different cutter regions, it would be very time-consuming. Suppose we still use the previously mentioned cuter list with 13 cutters available. In CATIA rework function, if we set the tolerance to be 7.874e-006 in, one given cutter size takes the CATIA about 45 minutes to create the rework boundaries. So a total of 13 cutters would take 9.75 hours to obtain all

the boundaries, which is more than twice of the time required by the proposed model. Remember that tolerance in the proposed model is set to be zero, and CATIA uses programming language for its system. If this work is implemented by a programming language, for example, C++, the computational time can be shrunk to half an hour. Thus, the iKD-Tree method overshadows the CATIA rework function, considering the efficiency of cutter size selection.

7.5.2 Boundary accuracy

In order to make the comparison simple and easy, it is assumed that the available ball end-mill sizes for the hairdryer mould are only 1 inch, ½ inch, and ¼ inch in radius. Thus, the time required for CATIA to generate the rework boundaries for those cutter sizes will be tolerable. As CATIA is the widely accepted CAD/CAM software, the rework boundary accuracy is trustable. Here we need to check if the boundaries generated from iKD-Tree method match the results from the CATIA rework function.

The boundaries of the available cutter sizes obtained from the iKD-Tree method are shown in Figure 7.4. Here the colors of dashed lines indicate the types of boundary. The definitions of boundary types are described as follows:

- Type 1(red): the boundary between 1 inch cutter region and ½ inch cutter region
- Type 2(green): the boundary between ½ inch cutter region and ¼ inch cutter region
- Type 3(yellow): the region of the boundary that no cuter in the list can access

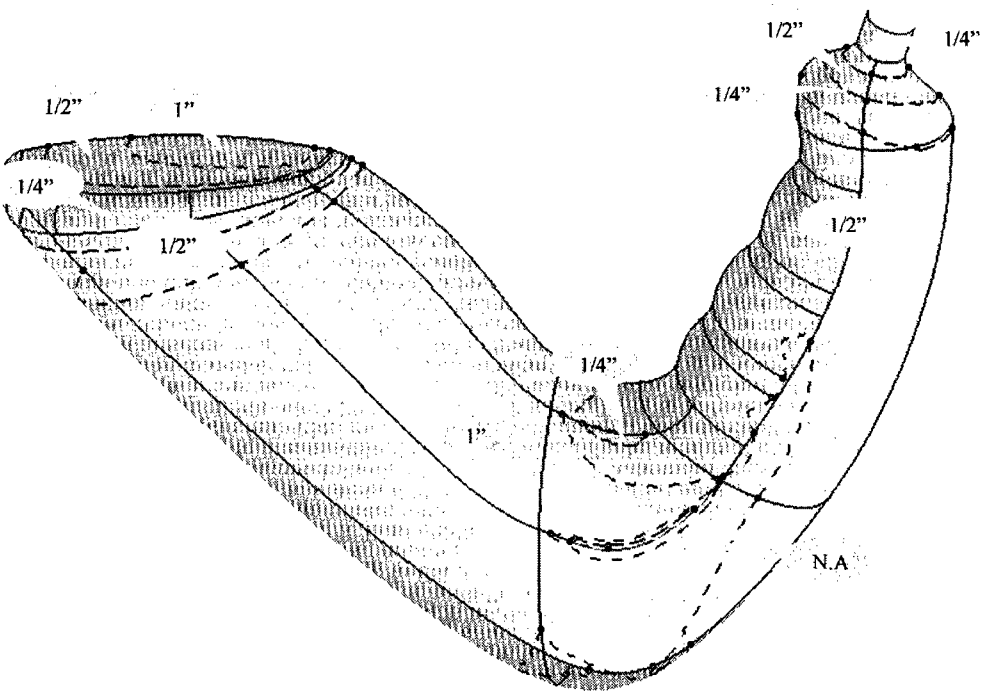


Figure 7.4 Boundaries generated by iKD-Tree method

The orange callouts in Figure 7.4 also indicate the largest available cutter sizes (in inch) that can be used to cut the corresponding regions. The callout with 'N.A.' signifies that no available cutter can be used to cut the region enclosed by the yellow dashed boundary.

In CATIA rework function, set the tolerance to be 7.874e-006 in and then generate the same three types of boundaries, using the same three cutters. The results are shown in Figure 7.5, Figure 7.6, and Figure 7.7. Note that CATIA rework function always uses the same type of red thin line to represent the boundary, regardless of boundary types. In Figure 7.5, it is seen that the type 1 boundary obtained from the CATIA rework exactly coincides with the red dashed line obtained from the iKD-Tree method. Similarly, from Figure 7.6, one can see that the type 2 boundaries from both CATIA rework function and the proposed method match quite well. In Figure 7.7, we can see that boundaries of type

3 from both methods match also very well, except that the area enclosed by the boundary from the proposed method is slightly larger than that from the CATIA rework function. This means the proposed method is somewhat conservative or safer. Therefore, we can conclude that boundary accuracy from the proposed method is equivalent to that from the CATIA.

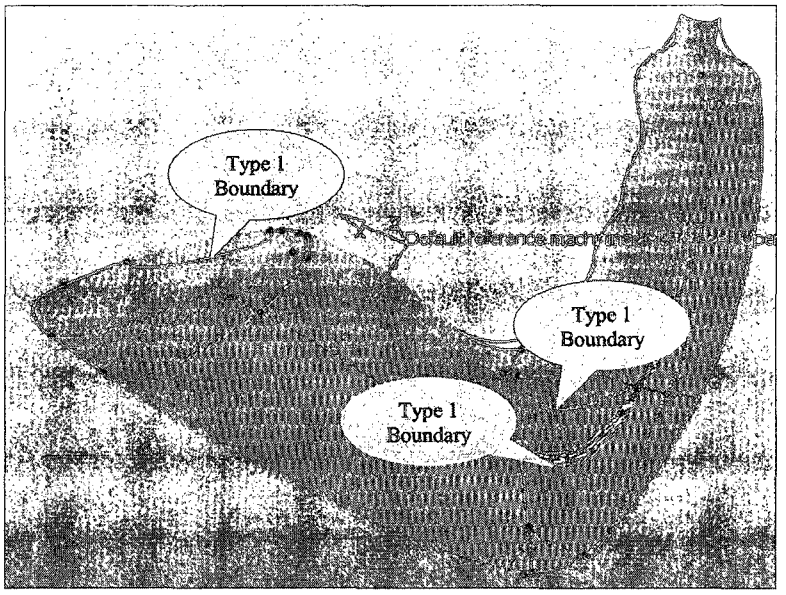


Figure 7.5 Comparison of boundary type 1

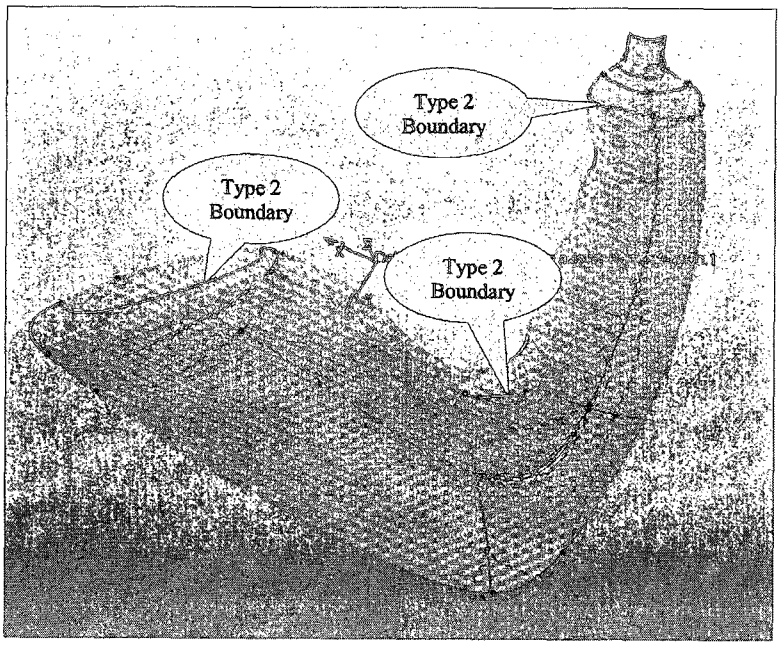


Figure 7.6 Comparison of boundary type 2

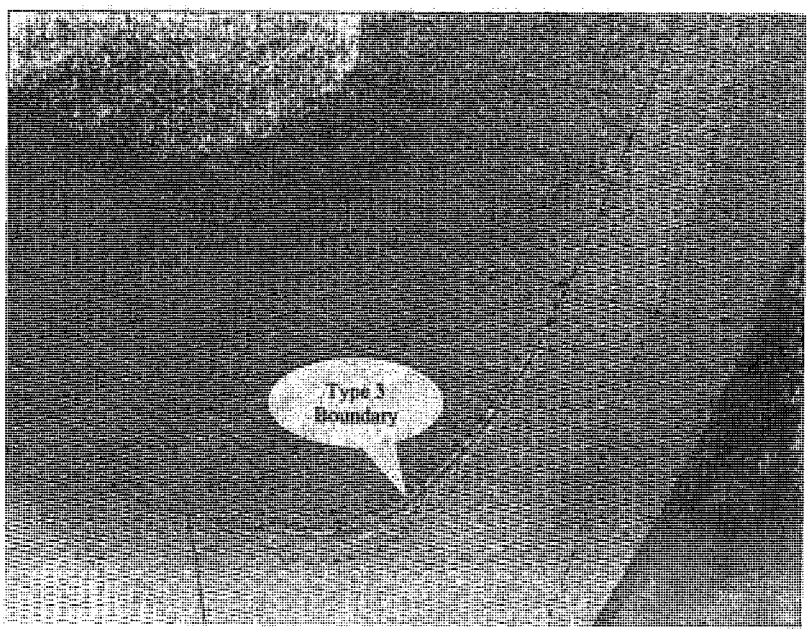


Figure 7.7 Comparison of boundary type 3

7.6 Machining Time Comparison

In order to see the machining efficiency, the previously determined three cutters (1/2 inch, 1 inch, and 2 inch) are used to plan the tool path in their corresponding boundaries for the hairdryer mould. At the same time, the ½ inch cutter, the smallest cutter among them, is also used to cut the same part. Their machining times and total times used are shown in Table 7.3.

Table 7.3 Machining time and total time comparison

Three cutter approach				One cutter approach
Individual cutter machining time (min)			Total machining time (min)	Machining time (min)
2 inch	1 inch	½ inch		½ inch
9.40	12.93	2.62	24.95	54.23

From Table 7.3, it is seen that the total machining time by using three ball end-mills is less than the half of the machining time by using one ball end-mill. The results are the same as those from the Liu's work. Indeed, the material removal rate can be significantly increased through the iKD-Tree model.

CHAPTER 8. CONTRIBUTIONS AND FUTURE RESEARCH

8.1 Contributions

In this research, an integrated KD-Tree cutter size search method is proposed to quickly determine the largest combination of gouging and interference-free cutters and their corresponding boundary regions, for 3-axis finish machining of sculptured surfaces. Those cutters can be used to achieve a high material removal rate, while maintaining the quality of the machined part. The main contributions of this work are summarized as follows.

First, an improvement has been made in Liu's imaginary cutter model to overcome its drawback. Since Liu's original model becomes invalid in the case when surface normal at a CC point is parallel to the tool axis, a new model has been developed for this particular case. As a result, the improved imaginary cutter size model can be applied to any surface conditions.

The introduction of the STL discrete point search concept is a methodological breakthrough of cutter size determination. In this work, the vertices extracted from the part STL file are employed as the testing points to define the cutter size, which is the radical way to accelerating the searching speed due to the discrete and finite nature of those vertices. Furthermore, the search accuracy is guaranteed by the adaptability of the STL format: the number of the test points is proportional to the curvature of the surface where those points lie. Additionally, to eliminate the chance of repeated search, an

efficient and straightforward algorithm has been developed to filter the large percentage of redundant vertices in the STL file.

Further, the suggested Vertex KD-Tree search method enables an extremely purposeful and high-speed search. First, the cutter shadow has been used to define the search area that the cutter would likely touch. Therefore, the search will be performed only in the necessary region. Second, adopting the efficient region query algorithm, the KD-tree search allows us to quickly identify all vertices within area covered by the cutter shadow.

More importantly, an integrated KD-tree model has been established to search the desired cutter sizes at an acceptable computational time and accuracy. The model incorporates the vertex KD-Tree method and Khan's local gouging detection method. Based on the Khan's method, an efficient initial cutter search algorithm has been developed. This initial cutter size serves as an upper limit of the allowable cutter to determine the radius of the cutter shadow for the vertex KD-Tree method.

Finally, the integrated model can achieve the comparable accuracy as the Liu's PSO cutter size method, but only requires 18% of the computational time. Besides, the cutter size boundaries generated by this model have the same accuracy as that generated by the CATIA rework function, but requires less than half of the running time needed by the CATIA to so the same job.

8.2 Future Research

For the future research, several directions are suggested:

- Develop the improved local gouging detection model that can be applied to the surfaces that are connected with any kind of continuity, including C^0, C^1, and C^2, so that the accuracy of the local gouging detection becomes more dependable.
- Conduct research on the re-meshing of STL facets to generate testing points other than the vertices to augment the accuracy of the cutter size search.
- Extend this work to 5-axis surface machining since many complex parts have to be produced by multiple-axis machining. The typical example is compressor impeller.

LIST OF REFERENCES

[1] P. Fallböhmer, T. Altan, H. K. Tönshoff, and T. Nakagawa, "Survey of die and moldmanufacturing industry". Journal of Materials Processing Technology, 1996(59), p. 158-168.

[2] G. Glaeser, J. Wallner, and H. Pottmann, "Collision-free 3-axis milling and selection of cutting tools". Computer-Aided Design, 1999(31), p. 225-232.

[3] H. Pottmann, J. Wallner, G. Glaeser, and B. Ravani, "Geometric criteria for gouge-free three-axis milling of sculptured surfaces". Transactions of ASME, Journal of Mechanical Design, 1999(121), p. 241-248.

[4] J. H. Yoon, H. Pottmann, and Y. S. Lee, "Locally optimal cutting positions for 5-axis sculptured surface machining". Computer-Aided Design, 2003(35).

[5] A. Rao and R. Sarma, "On local gouging in five-axis sculptured surface machining using flat-end tools". Computer-Aided Design, 2000(32), p. 409-420.

[6] S. H. Khan, *Local Gouging Detection and Tool Size Determination for 3-Axis Finish Machining of Sculptured Surface Parts*, in *the Department of Mechanical and Industrial Engineering*. 2006, Concordia University.

[7] J. H. Oliver, D. A. Wysocki, and E. D. Goodman, "Gouge detection algorithm for sculptured surface NC generation". Transactions of ASME, Journal of Engineering for Industry, 1993(115), p. 139-144.

[8] D. C. H. Yang and Z. Han, "Interference detection and optimal tool selection in 3-axis NC machining of free-form surfaces". Computer-Aided Design, 1999. **5**(31), p. 371-377.

[9] Y. S. Lee and T. C. Chang, "Automatic cutter selection for 5-axis sculptured surface machining". International Journal of Production Research, 1996. **34**, p. 997-998.

[10] K. K. George and N. R. Babu, "On the effective tool path planning algorithms for sculptured surface manufacture". Computers and Industrial Engineering, 1995. **28**(4), p. 823-838.

[11] A. Hatna and R. J. Grieve, "Pre-processing approach for cutter interference removal". International Journal of Production Research, 2001. **3**(39), p. 435-460.

[12] G. Liu, *Automated Cutter Size and Orientation Determinations for Multi-Axis Sculptured Part Milling*, in *Mechanical & Industrial Engineering* 2007, Concordia University: Montreal, Canada. p. 1-88.

[13] D. Rypl and Z. Bittnar, "Triangulation of 3D Surfaces Described by Stereolithography Files". (Currently 2008, http://mech.fsv.cvut.cz/~dr/papers/Lisbon04/node10.html).

[14] T. Wu and E. H. M. Cheung, "Enhanced STL". International Journal of Advanced Manufacturing Technology 2006. **29**, p. 1143-1150.

[15] B. Lauwers, P. P. Lefebvre, and K. U. Leuven, *Part Analysis Algorithms for Efficient 5-Axis Milling Strategy Planning of Sculptured Surfaces*, Department of Mechanical Engineering, Division PMA, Celestijnenlaan 300B, B-3001 Leuven, Belgium.

[16] *CATIA Online Help Documentation, CATIA V5 R17*, Dassult Systems.

[17] S. Cui, Y. Zhang, S. Liang, and D. Li, "An Algorithm for Fast Filtering Redundancy Vertices in STL files". Journal of China Mechanical Engineering, 2001(2).

[18] X. Cheng, D. Li, H. Zhou, and S. Cui, "Algorithm for fast filtering redundancy vertex in STL solid based on Hashtable". Journal of Huazhong University of Science & Technology (Nature Science Edition), 2004. **32**(6).

[19] J. B. Rosenberg, "Geographical data structures compared: a study of data structures supporting region queries [VLSI CAD]". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1985. **CAD-4**(1), p. 53-67.

[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching". Communications of the ACM, 1975. **18**(9), p. 509-17.

[21] J. L. Bentley, *K-d trees for semidynamic point sets*. 1990. Berkeley, CA, USA, Published by ACM, New York, NY, USA.

[22] H. T. Yau, C. M. Chuang, and Y. S. Lee, "Numerical control machining of triangulated sculptured surfaces in a stereo lithography format with a generalized cutter". International Journal of Production Research, 2004. **42**(13), p. 2573-2598.

[23] S. Akella, "CSCI-2300: Data Structures and Algorithms Project 2 — Kd-Trees". (Currently 2007, http://www.cs.rpi.edu/~sakella/dsa/projects/project2/project2.pdf).