

1993

# Automatic tool path generation for numerically controlled machining of sculptured surfaces

Xiaoxia Li

*University of New Hampshire, Durham*

Follow this and additional works at: <https://scholars.unh.edu/dissertation>

---

## Recommended Citation

Li, Xiaoxia, "Automatic tool path generation for numerically controlled machining of sculptured surfaces" (1993). *Doctoral Dissertations*. 1734.

<https://scholars.unh.edu/dissertation/1734>

This Dissertation is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact [nicole.hentz@unh.edu](mailto:nicole.hentz@unh.edu).

## DEDICATION

To Mom & Dad

## ACKNOWLEDGEMENTS

Many people have provided me with their assistance. Here I would like to thank Dr. Barry Fussell, Dr. David Limbert, Dr. Daniel Bergeron, Dr. Robert Jerard, Dr. Scot Drysdale and Dr. Bob Phillips for devoting time to review this thesis. Special thanks to my thesis advisor Dr. Robert Jerard for his guidance and direction. Without his support, none of the work would have been possible. He provided the creative and interdisciplinary atmosphere in which to pursue this project. I greatly appreciate the friendship with him and his family, grateful for their care and help during the endeavor. I would also like to acknowledge the contributions of Professor Drysdale's students, Kamran Ozair and Christine McGavran who wrote the software used to triangulate the surface. Thanks also to the Ford Motor Company and the National Science Foundation for funding this project.

I would also like to thank Mom, Dad, Xiaoqiu, Xiaoping and Dazhi for their love and emotional support, and for their understanding and tolerating my absence. This work is dedicated to my Mom who left the world just one week before we could meet after three year separation. Her encouragement, enthusiasm toward work and devotion to career have always been the impetus of all my academic work.

Finally, I would like to thank my husband Fuzhong Yu, without whom I would never have survived.

## TABLE OF CONTENTS

<b>Dedication</b> .....	iii
<b>Acknowledgement</b> .....	iv
<b>List of Figures</b> .....	viii
<b>List of Algorithms</b> .....	xi
<b>List of Tables</b> .....	xii
<b>Acronyms</b> .....	xiii
<b>Abstract</b> .....	xiv
<b>Preface</b> .....	xvi

### CHAPTER

<b>1 Introduction</b> .....	1
<b>2 Related Work</b> .....	5
2.1 Three-Axis Tool Path Generation .....	5
2.1.1 Parametric Indexing .....	6
2.1.2 Offset Surface Based Tool Path Generation .....	8
2.1.3 Image Based Surface Discretization .....	9
2.1.4 Object Based Surface Discretization .....	10
2.1.5 Polygonal Approximation .....	12
2.2 Five-Axis Tool Path Generation .....	14
2.3 NC Program Simulation, Verification and Correction.....	16
2.3.1 Simulation Based on Solid Modeling .....	16
2.3.2 Image Based Surface Discretization .....	17
2.3.3 Object Based Surface Discretization .....	18
2.3.4 Polygon Based Surface Approximation .....	19
2.4 Surface Quality and Machining Efficiency .....	19

<b>3</b>	<b>Three-Axis Tool Path Generation Based on Surface Triangulation</b>	
3.1	Surface Triangulation .....	21
3.2	Tool Path Generation Overview .....	22
3.3	Edge Position Identification .....	24
3.3.1	Leading Edge .....	24
3.3.2	Trailing Edge .....	26
3.3.3	Avoiding Gouging at Edge Positions .....	28
3.4	Tool Positioning .....	30
3.5	Test Criteria, Procedures and Test Results .....	33
3.5.1	Parameter selection .....	33
3.5.2	Test Surfaces .....	34
3.5.3	Test Results .....	35
<b>4</b>	<b>Tool Path Generation for Finishing Sculptured Surfaces</b> .....	<b>37</b>
4.1	Undercut Area Identification.....	38
4.2	Tool Path Planning.....	40
4.2.1	Block Cutting .....	40
4.2.2	Outer-Pruning Cutting .....	41
4.2.3	Inter-Pruning Cutting .....	43
4.3	Dwell Mark Elimination .....	45
4.4	Test Results .....	47
<b>5</b>	<b>Five Axis Tool Path Generation Based on</b>	
	<b>Surface Polygonalization</b> .....	<b>50</b>
5.1	Surface Approximation .....	50
5.2	CC Data Generation .....	52
5.2.1	Edge List Generation .....	52
5.2.2	Edge Handling .....	55
5.2.3	CC Point Interpolation .....	59
5.2.4	Effective Radius of Tilting Flat-End Tool .....	61
5.2.5	Edge List Stepover .....	63

5.3	CL Data Generation .....	66
5.3.1	Interference Checking .....	67
5.3.2	Tool Position Correction .....	71
5.3.3	Angle Adjustment .....	72
5.4	Operation Procedures, Test Cases and Test Results .....	76
<b>6</b>	<b>Fixed-Axis Tool Path Generation .....</b>	<b>79</b>
6.1	Implementation Overview .....	80
6.2	Tool Position Interpolation .....	80
6.3	Tool Path Modification .....	82
6.4	Tool Path Modification .....	83
6.5	Localization .....	84
6.6	Derivation of Stepper Distance .....	85
6.7	Test Results .....	88
<b>7</b>	<b>Conclusions and Recommendations .....</b>	<b>90</b>
7.1	Three-Axis Machining .....	90
7.2	Finish Machining .....	91
7.2.1	Future Work .....	92
7.3	Five-Axis Machining .....	92
7.3.1	Future Work .....	94
7.4	Fixed-Axis Machining .....	96
<b>Appendix A</b>	<b>The Data Structure Used in Tool Path Generation .....</b>	<b>98</b>
<b>Appendix B</b>	<b>Ball-End Tool Positioning with a Triangle .....</b>	<b>99</b>
<b>Appendix C</b>	<b>Touch Point vs. an Edge and a Triangle .....</b>	<b>102</b>
<b>Appendix D</b>	<b>Tilting Angle for the Vertex Case .....</b>	<b>104</b>
<b>Appendix E</b>	<b>Tilting Angle for the Edge Case .....</b>	<b>105</b>
<b>Appendix F</b>	<b>Tilting Angle for the Triangle Case .....</b>	<b>108</b>
<b>References</b>	<b>.....</b>	<b>111</b>
<b>Color Plates</b>	<b>.....</b>	<b>116</b>

## LIST OF FIGURES

1.1	General Structure of an Modern NC System .....	2
2.1	Parametric Indexing .....	7
2.2	Self-Intersection Loop in an Offset Surface .....	8
2.3	Image Surface Discretization .....	9
2.4	Tool Positioning with Point or Tangent Plane approach .....	10
2.5	Tool Positioning with Quad Trees .....	13
2.6	Tool Paths between Two Consecutive Isoparametric Curves .....	14
2.7	Cutter Touching an Edge with Its Leading or Trailing Edge .....	15
2.8	Simulation Based on Point-Vector Method .....	18
3.1	Bucketing Triangles .....	23
3.2	Tool Edge Position .....	25
3.3	Searching Localization .....	25
3.4	Small Local Area (SLA) .....	26
3.5	Finding a Trailing Edge .....	28
3.6	First Position after an Edge Position .....	28
3.7	Last Tool Position before a Trailing Edge .....	29
3.8	Edge Selection .....	29
3.9	Local Checking .....	30
3.10	Locating a Touch Point .....	31
3.11	An Example of Cutting Angle .....	34
4.1	Excess Material .....	37
4.2	Grid Superimposed on the Undercut Domain .....	38
4.3	Possible Cases in Bucket Subdivision .....	39
4.4	Block Cutting .....	41
4.5	Outer-Pruning Cutting .....	41
4.6	Tool Paths for Outer-Pruning Cutting .....	42

4.7	Inter-Pruning Cutting .....	43
4.8	An Example of Bucket Row Segment Sorting .....	44
4.9	Tool Paths for Inter-Pruning Cutting .....	44
4.10	Radius vs. Tool Locations .....	45
4.11	Positioning with Expanded Tools .....	46
4.12	Structure of the Finishing System .....	46
5.1	Relationship between Point Spacing (d) and Approximation Error (e) .....	51
5.2	Examples of Overlapping Surfaces .....	51
5.3	Edge List Generation .....	53
5.4	Overlapping Segment Removal .....	54
5.5	CL Edge and Exact Tool Edge Position .....	56
5.6	Finding a Leading Edge .....	57
5.7	Finding a Trailing Edge .....	57
5.8	CC Point Interpolation .....	59
5.9	Default Increment .....	60
5.10	Curved Formed by the Top Center of a Tool .....	60
5.11	Effective Tool Radius .....	62
5.12	Cusp Plane .....	64
5.13	Effective Tool Center Location .....	64
5.14	Stepover vs. Cusp Height .....	65
5.15	Offsetting Cutter Center Location .....	66
5.16	CC Points vs. Tool Paths .....	67
5.17	Interference between a Tool and a Surface .....	67
5.18	Checking Localization .....	68
5.19	Transformation on the Coordinate System .....	68
5.20	Location of A Triangle Relative to the Cutter Shadow .....	69
5.21	Interference Checking .....	70
5.22	Tilting Angle vs. Tool Position .....	71
5.23	Torus Formed by Rotating a Disk .....	74



6.1	Fixed Tool Orientation .....	79
6.2	Tool Position Modification .....	81
6.3	Default Stepsize .....	81
6.4	Transformation and the Tool Shadow .....	82
6.5	Tool Path Expansion .....	83
6.6	Localization of Searching Domain .....	84
6.7	Tilting Effect .....	85
6.8	Tilting Angle .....	86
6.9	Silhouette of the Tool Bottom on the xy Plane .....	87
6.10	Tilting Angle $\alpha$ .....	88
7.1	Curved Tool Paths .....	92
7.2	Tool Path Sorting .....	95
A1	Data Structures Used in Tool Path Generation Systems .....	98
B1	Tool Contacting an Edge of a Triangle Plane .....	99
C1	Finding a Touch Point on a Triangle .....	102
D1	Tool Bottom Touching a Vertex .....	104
D2	Tool Side Touching a Vertex .....	104
E1	Tool Touching an Edge .....	105
F1	Tool Touching a Point on a Triangle Plane .....	108

## LIST OF ALGORITHMS

3.1	Tool Path Generation .....	23
3.2	Finding a Leading / Trailing Edge .....	27
3.3	Tool Positioning .....	31
4.1	Recursive Bucket Subdivision .....	40
4.2	Outer-Pruning Tool Path Generation .....	42
4.3	Inter-Pruning Tool Path Generation .....	45
4.4	Tool Radius Selection .....	47
5.1	CC Edge List Generation and Overlapping Segment Removal .....	54
5.2	Finding a Leading / Trailing Edge .....	58
5.3	Interference Checking .....	69
5.4	Edge Selection and Touch Point Setting for a Vertical Triangle .....	70
5.5	Tool Position Modification .....	72
5.6	Finding a tilting Angle When a Triangle Interferes with the Tool .....	73
5.7	Finding a Tilting Angle for the Tool to Touch a Vertex .....	74
5.8	Interference Checking between a Tool and an Edge .....	75
6.1	Tool Position Modification .....	82

## LIST OF TABLES

3.1	Triangle Edge Selection for Tool Edge Positioning .....	30
3.2	Point Location vs. a Triangle Plane .....	32
3.3	Simulation Results vs. Tool Positioning Approaches .....	35
4.1	Simulation Results vs. Cutting Styles .....	48
5.1	Simulation Results on Five-Axis Tool Paths .....	77
6.1	Simulation Results on Fixed-Axis Tool Paths for zip5 .....	89
6.2	Simulation Results on Fixed-Axis Tool Paths for z6324r .....	89
6.3	Simulation Results on Fixed-Axis Tool Paths for saddle .....	89

## ACRONYMS

<b>2D</b>	<b>Two Dimensions</b>
<b>3D</b>	<b>Three Dimensions</b>
<b>APT</b>	<b>Automatic Programming Tool</b>
<b>B-rep</b>	<b>Boundary Representation</b>
<b>CC</b>	<b>Cutter Contact point</b>
<b>CL</b>	<b>Cutter Location</b>
<b>CLDATA</b>	<b>Cutter Location Data</b>
<b>CNC</b>	<b>Computer Numerical Control</b>
<b>CSG</b>	<b>Constructive Solid Geometry</b>
<b>D+</b>	<b>Positive Cutting Direction</b>
<b>D-</b>	<b>Negative Cutting Direction</b>
<b>FINISH</b>	<b>A tool path generation system for finishing surfaces</b>
<b>FIVEX_INDEX</b>	<b>A five-axis tool path generation system</b>
<b>FIX_AXIS_INDEX</b>	<b>A tool path generation system for tools with fixed orientations</b>
<b>LSF</b>	<b>Least Square Fitting</b>
<b>NC</b>	<b>Numerical Control</b>
<b>NURB</b>	<b>Non-Uniform Rational B-spline</b>
<b>SLA</b>	<b>Small Local Area</b>
<b>SPS</b>	<b>Surface Point Set</b>
<b>STS</b>	<b>Surface Triangle Set</b>
<b>TP</b>	<b>Touch Point</b>
<b>TRI_XYINDEX</b>	<b>A three-axis tool path generation system based on polygons</b>
<b>XBEnd</b>	<b>the last bucket index along the +x axis</b>
<b>XBStart</b>	<b>the first bucket index along the +x axis</b>
<b>YBEnd</b>	<b>the last bucket index along the +y axis</b>
<b>YBStart</b>	<b>the first bucket index along the +y axis</b>

## **ABSTRACT**

# **AUTOMATIC TOOL PATH GENERATION FOR NUMERICALLY CONTROLLED MACHINING OF SCULPTURED SURFACES**

by

Xiaoxia Li

University of New Hampshire, May, 1993

This dissertation presents four new tool path generation approaches for numerically controlled machining of sculptured surfaces: TRI\_XYINDEX, FINISH, FIVEX\_INDEX, FIX\_AXIS\_INDEX. Unlike the traditional approach to incrementing a tool along the isoparametric curves on individual surface patches, all of the above systems index the tool across the object surface in the Cartesian space so that evenly distributed tool paths are accomplished.

TRI\_XYINDEX is a three-axis tool path generation system which uses a surface triangle set (STS) representation of the surface for tool position calculations. Surface edges are detected with local searching algorithms. Quick tool positioning is achieved by selecting candidate elements of polygons. A comparison was made with an earlier system which uses a surface point set (SPS) approximation. Test results show that TRI\_XYINDEX is more efficient when machining surfaces which are relatively flat while the discrete point approach is faster for highly curved surfaces. FINISH was developed for generating three-axis ball-end tool paths for local surface finishing. It was based on the SPS. Given a surface with excess material represented by a set of discrete points, FINISH automatically identifies the undercut areas (i.e. with excess material). Depending on the undercut area distribution, three kinds of tool paths are planned. Dwell marks at the boundary of undercut areas are eliminated with an expanded tool radius scheme. Results show that FINISH provides significant improvements in machining efficiency.

FIVEX\_INDEX is developed for generating five-axis flat-end tool paths. It uses an

**FIVEX\_INDEX** is developed for generating five-axis flat-end tool paths. It uses an STS approximation. Contact points on the surface are derived from edge lists obtained from the intersections of vertical cutting planes with the polygon set. The distances between adjacent end points set an initial step-forward increment between surface contact points. To verify tool movements, some intermediate tool positions are interpolated. If gouging occurs, tool positions are corrected with a resetting scheme or an angle adjustment scheme of the tool axis orientation. The key features of **FIVEX\_INDEX** are: 1. a polygon set representing an object which may be composed of multiple surfaces; 2. Surface contact point generation by cutting plane intersection; 3. simple tool incrementing and positioning algorithms; 4. minimal user interaction; 5. user controlled accuracy of resulting tool paths.

**FIX\_AXIS\_INDEX** is a subsystem of **FIVEX\_INDEX**, generating tool paths for a tool with fixed orientations. Surface contact points are generated similar to **FIVEX\_INDEX** while tool positions are corrected with the highest point technique along the tool axis direction. Linear fitting is applied to output tool positions. **FIX\_AXIS\_INDEX** is preferred for machining surfaces curved in one direction, such as ruled surfaces. Test results show that **FIX\_AXIS\_INDEX** can serve as a three-axis tool path generation system but a five-axis machine is required to do it. By adjusting the tilting angle, equivalent ball-end tool effects can be realized.

## **PREFACE**

A major goal in software development for numerically controlled machining of sculptured surfaces (such as the stamping dies for manufacturing car bodies and ship hulls) is to provide accurate tool paths directly from a computer model of an object. If the tool paths are “error free”, the workpiece will not deviate from the design object by more than the user defined tolerances. Automatic tool path generation systems can shorten the product development cycle, reduce the labor requirement and achieve higher level of integration between design and manufacturing.

With the goal of developing improved methodologies for tool path generation, this thesis presents four systems: TRI\_XYINDEX, FINISH, FIVEX\_INDEX and FIX\_AXIS\_INDEX. These systems are evaluated with test surfaces for accuracy and efficiency tradeoffs. The thesis is organized in the following way.

Chapter 1 introduces a general model of an NC system. General methods used to overcome some of the primary obstacles to making a robust NC system are explained briefly. Relevant literature is discussed in Chapter 2. The number of degrees of freedom of a milling machine greatly impacts the approach and related work is divided into three-axis and five-axis cases. The relationship between surface quality and tool shapes is also explored in Chapter 2.

Chapter 3 presents a three-axis tool path generation system based on an STS approximation of the surface: TRI\_XYINDEX. Comparisons are made with an earlier system based on an SPS approximation. Although the STS calculations are slower and more complex than the SPS calculations, this factor is traded-off against the smaller number of triangles required to approximate a surface accurately. In general, the STS approach seems to work better when the surfaces are relatively flat. Another advantage of the STS approach is that the surface discretization is independent of tool size and shape.

Chapter 4 discusses the implementation of an automatic tool path generation system (FINISH) for finishing surfaces locally. Techniques for undercut area identification, tool

path planning and dwell mark elimination are highlighted. Results for various approaches are compared with the traditional finishing approach of remachining the whole surface.

Chapter 5 investigates the methodologies for five-axis tool path generation. FIVEX\_INDEX, a basic five-axis tool path generation system is developed based on an STS approximation. Algorithms and techniques for dealing with overlapping surfaces and gaps, surface contact point generation, tool positioning and modification are discussed in detail. The algorithms are evaluated with various test surfaces for accuracy and efficiency.

Chapter 6 discusses the implementation of FIX\_AXIS\_INDEX, a tool path generation system in which the flat-end tool is held at a fixed axis orientation with respect to its moving direction. As a subsystem of FIVEX\_INDEX, FIX\_AXIS\_INDEX serves as an alternative to three-axis machining with a ball-end tool. The algorithms of FIX\_AXIS\_INDEX are much simpler than that of FIVEX\_INDEX and work well with surfaces of modest complexity.

Chapter 7 gives conclusions and recommendations for future work. Appendices A - F provides related mathematics derivations and data structures.



## CHAPTER 1

### INTRODUCTION

Numerically controlled (NC) machining is an important aspect of the design and manufacturing of sculptured surfaces such as automobile bodies, aircraft, and ship hulls <sup>[1]</sup>. The mold or stamping die used to create such a part is usually machined with a three or five axis mill controlled by an NC program<sup>[2]</sup>. With the increased use of computers in design and manufacturing, the traditional processes for creating a stamping die have been replaced by an NC system. For example, the wooden model created by skilled craftsmen is replaced by a mathematical model stored in a computer, and the tracing mill used to replicate the wooden model in steel is replaced by tool path generation software and an NC machine<sup>[3]</sup>.

The goal of this research is to create new algorithms for NC tool path generation which are more reliable and easier to use. Specific shortcomings of current systems include:

1. NC programming is too time consuming
2. Tool paths may gouge the surfaces (overcut)
3. Tool paths may leave unwanted material (undercut)
4. Tool paths are not efficient.

A new approach to the NC tool path generation process is shown in Fig. 1.1. Each block is discussed briefly in the following paragraphs.

1. **Geometric Modeling** describes the shape of an object <sup>[4]</sup>. Commonly used methods are surface modeling and solid modeling. In surface modeling, an object is represented with a set of surface patches. The most popular surface patches are planes, ruled surfaces, paraboloid, bicubic surfaces, B-spline surfaces<sup>[5]</sup>, Non-Uniform Rational B-spline (NURB) surfaces<sup>[6]</sup>, Coons surfaces <sup>[7]</sup>, etc. In solid modeling, an object surface is generally described with either a boundary representation (B-rep) or by constructive solid geometry (CSG) <sup>[8]</sup>. In B-rep, an

object is represented by a set of vertices, edges, and faces <sup>[9]</sup>. In CSG, the object is constructed from primitives of closed surfaces (such as spheres, cubes, prisms and cones) with set theory and Euler operators <sup>[10]</sup>. For a complex sculptured surface, the design may also involve processes such as smoothing<sup>[11]</sup>, composing<sup>[12]</sup>, truncating (or trimming), blending<sup>[13, 14]</sup> and lofting<sup>[7]</sup> of primitives (or surface patches). The mathematical description of the design surface can be used directly or be transformed into other representations to fulfill some tasks. Different surface representations result in different approaches to tool path generation.

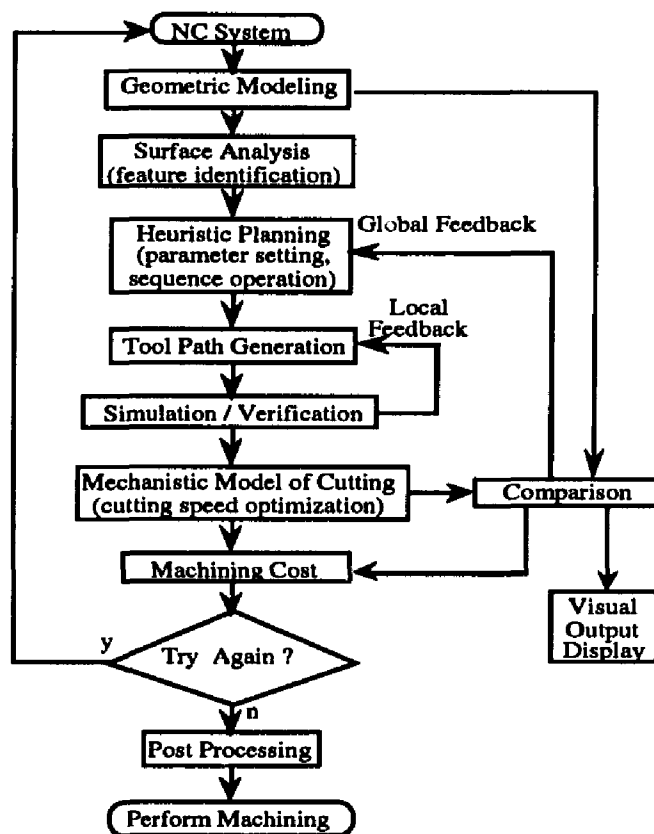


Figure 1.1 General structure of modern NC system

2. **Surface Analysis** investigates the intrinsic shape and properties of an object surface, such as surface curvature and continuity. These features are commonly visualized with an aid of a color raster device based on a color-coded map.

3. **Heuristic Planning** defines preliminary parameters and operation sequences based on surface analysis and machining requirements. For instance, it determines the shape and size of tools to be used, and the number of roughing and finishing processes.
4. **Tool Path Generation and Simulation /Verification** is the core of the new approach. The goal is to generate “error free” tool paths. Here “error free” means that the generated tool paths guarantee that the machined surface is within the predefined tolerance. Geometric simulation and verification can be used to detect errors in tool paths. Because of the risk of human error, tool paths should be simulated, verified and corrected before actual machining. The related work on these subjects is discussed in the next chapter.
5. **The Mechanistic Model of Cutting** models the interaction between the cutting tool and the workpiece. Machining efficiency and surface quality depends on factors such as the spindle speed, feedrate, tool deflection, tool wear, etc.
6. **Comparison** determines the differences between the design surface and the machined surface. An NC program is acceptable whenever the overall error of the machined surface is within the predefined tolerance.
7. **Visual Output Display** provides graphic descriptions of the designed and machined surfaces, machining errors, tool paths, etc. These features are very helpful to NC programmers.
8. **Machining Cost** is critical for a commercial NC system. The cost is directly proportional to the machining time.
9. **Post Processor** creates the NC instruction for an individual type of CNC milling machine and download to the machine to produce workpieces <sup>[15]</sup>. Currently in most CAD/CAM systems, this process is separated from tool path generation, which means that tool paths are directly generated from surface design without considering any machining restrictions. For example, one restriction in a five axis milling machine is the limitation on the angular excursion of the rotational axes. These machine restrictions should be taken into account during the tool path generation phase in order to produce an efficient NC program.

As seen from the above flowchart, an NC system is composed of many aspects which are intertwined. For instance, a suitable geometric model should be available in tool path generation and/or simulation. Also during the tool path generation phase, geometric simulation may be useful in order to determine if a candidate tool movement is acceptable. Issues related to tool path generation are investigated in this research.

## **CHAPTER 2**

### **RELATED WORK**

Tool path generation methodologies may be categorized in several different ways: three axis vs. five axis milling machines, sculptured vs. prismatic parts, and also by the underlying methods for surface description. Three axis milling machines are popular tools which have three degrees of translational freedom. Four and five axis machines have additional rotational degree(s) of freedom. Their path generation and simulation algorithms are quite different from those of the three axis case.

Tool path generation methods for sculptured surfaces may be subdivided into two general categories: isoparametric and non-isoparametric indexing. In isoparametric indexing a series of cutter contact (CC) points are determined by indexing along lines of constant parameters in the u-v space of the surface. Cutter center location points (CL) are then calculated from the CC points. In the second category, non-isoparametric indexing, constraints are placed on the CL points (e.g. x, y position is set) and the CC point is calculated. Calculation of CC points directly from CL constraints is difficult. This calculation can be simplified if the original parametric surface is replaced by an approximation. Various approximation schemes include offset surfaces, image space point sets, surface point sets (SPS) and surface triangle sets (STS). These approaches are discussed in more detail in the following sections.

#### **2.1 THREE AXIS TOOL PATH GENERATION**

The goal of tool path generation is to provide accurate and efficient tool paths. Accurate tool paths produce a machined surface within the tolerance zone of the mathematical surface. Inaccurate tool paths result in undercutting (i.e., excess material beyond the tolerance remains) or overcutting (i.e., gouging more than the allowable amount) or both. Efficiency is measured by both the time spent on tool path generation and

on actual machining. To accomplish these goals, an NC tool path generation system should possess the ability to increment the tool across the design surface correctly and efficiently. Individual tool positions must be “good”, i.e., be free of undercuts or overcuts. Spacing between tool positions along the path (step-forward distance) and between paths (step-over distance) is also critical.

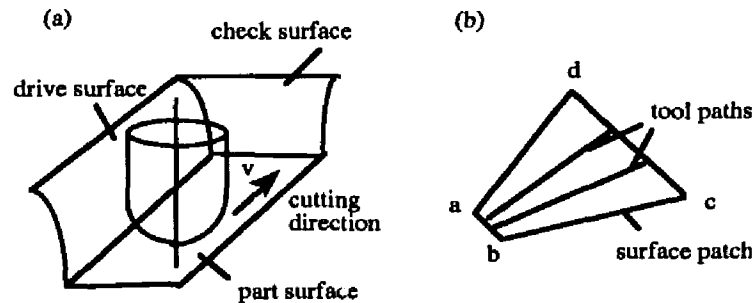
Generally, a design surface is composed of multiple surface patches connected with certain degrees of continuity such as C0, C1, C2, etc. C0 continuity refers to continuity of position; C1 refers to continuity of position and tangent vectors between surface patches; while C2 refers to continuity of position, tangent and curvature vectors. A robust system should be able to generate correct tool paths across multiple surfaces which only possess C0 continuity. Occasionally, it may be necessary to machine across surfaces that have small C0 discontinuities, i.e., small gaps between surfaces.

Generating tool paths is equivalent to finding a series of tool center locations (CLs). In traditional systems, CLs are derived from the cutter contact points (CCs) on the design surface. To generate gouge-free tool paths the algorithms should also have the ability to select the CCs correctly and to move a tool across multiple patches without gouging.

### **2.1.1 Parametric Indexing**

Parametric indexing is the most straightforward tool path generation approach. Each path is defined as an isoparametric curve on a surface patch by holding one of the two parameters constant. Since a sculptured surface is generally composed of more than one patch, interference between a tool and adjacent surfaces may take place. For a more robust system such as APT (automatically programming tool) <sup>[16]</sup>, gouging at adjacent surface patches is avoided with the aid of three temporary surfaces: check, part and drive surfaces (Fig. 2.1a). A tool moves along a drive surface while contacting a part surface. It stops when it touches a check surface. The difficulty of this approach lies in the determination of contact points of the tool with various surfaces. This is normally realized with iterative techniques like the Newton-Raphson Search Algorithm <sup>[17]</sup>. Therefore, a set of good guess positions are required in order for the algorithm to converge. To minimize the number of tool positions, variable size increments are used by some systems, such as the interactive

CAD/CAM package CISPA (Computer Interactive Surface Pre-APT) <sup>[18]</sup>. In CISPA, step-forward increments are obtained through a subdivision of each parametric curve. The curve is recursively subdivided until the deviation of the chord from the arc is within a defined bound. Stepover increments between toolpaths are defined based on the allowable cusp height and tool size to ensure that the cusps of the material left between adjacent paths are within the allowable tolerance. One inherent disadvantage of parametric indexing is that unevenly distributed tool paths can occur. For example in Fig. 2.1b, abcd is a non-rectangular surface patch with tool paths distributed densely over one end (ab) and coarsely over the other (cd).



**Figure 2.1** Parametric indexing

Huang<sup>[19]</sup> proposes a tool path planning scheme to solve the problem of unevenly distributed tool paths. Tool paths are offset from contact curves which are obtained from the intersection of cutting planes with the parametric surface. Cutting planes are planes containing the locus of CL positions. Generally, they can be arranged in any direction across the surface patch. Increments between contact points are dynamically defined by an algorithm based on “true machine error”. True machine error is derived from the orthogonal projection of tool movements onto the part surface. Therefore, errors caused by step-forward increments are bounded. However, testing adjacent surface contact points for possible interference is not sufficient to detect the interference of the tool with adjacent surface patches when the object is composed of multiple patches.

### 2.1.2 Offset Surface based Tool Path Generation

In NC tool path generation systems an **Offset Surface** consists of the locus of tool CLs when the tool moves across an object surface. The offset surface is usually obtained by offsetting from the object surface at a distance equal to the cutter radius along certain directions (such as surface normals for ball-end tools). If the offset surface is available, tool paths can be easily arranged on it <sup>[20]</sup>. The main problem in offsetting surfaces is the possibility of self-intersection (Fig. 2.2). Self-intersection areas must be removed to avoid gouging. Methods used in self-intersection removal can be categorized as pre-path-generation removal and post-path-generation removal.

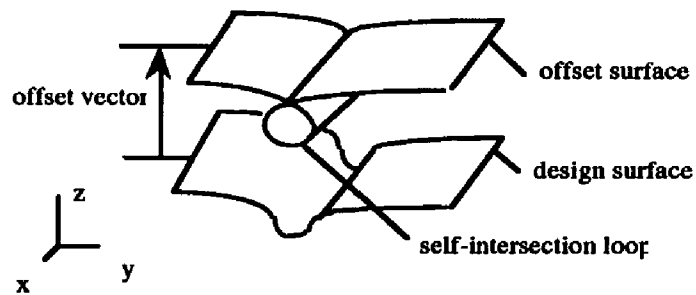


Figure 2.2 Self-intersection loop in an offset surface

In pre-path-generation removal, a self-intersection region is removed from the parametric surface patch first, then tool paths are generated. An example is Chen's method<sup>[21]</sup>, in which least squares fitting (LSF) and the Newton iteration method are used to remove self-intersection regions. This approach is, however, confined to a single bicubic surface patch. For an object composed of multiple patches, the LSF algorithm can not be directly applied. Even with some modification this approach would require extensive computation. In post-path-generation removal, the reverse order is adopted (i.e., tool paths are generated first and then overlapping tool path segments are eliminated)<sup>[22]</sup>. For example, to simplify the offsetting process, complex surface patches are approximated



by a set of bilinear surface patches <sup>[23]</sup>. Since the offset surface is directly calculated from the object approximation, it is also composed of a set of bilinear surface patches. Tool paths are planned on the offset surface by finding the intersections of these patches with parallel cutting planes. The self-intersection loops in the tool paths are removed by choosing the upper paths when more than one path is obtained. The simplicity in obtaining the offset surface is the notable advantage of this approach.

Choi's method<sup>[24]</sup> combines aspects of both offset surfaces and surface discretization. Local interference is avoided by testing the possible interference points related to a set of CC points. If interference is detected, related tool positions are modified. Self-intersecting tool paths are also removed. However, global interference may not be avoided since tool positions of possible interference can not be completely identified by merely testing CC points on adjacent pairs of CC paths.

### 2.1.3 Image Based Surface Discretization

In image based surface discretization, an object is represented with an image of evenly sampled points (see Fig. 2.3). If the tool axis is parallel to the z axis, the xy plane is the projection plane and the image is called a z-buffer image. Tool paths may be generated by contour tracking or x or y scanning. The former is done by tracking at a constant z value in the offset surface image, and the latter by simply scanning on the x and y axes with appropriate intervals<sup>[25,26]</sup>.

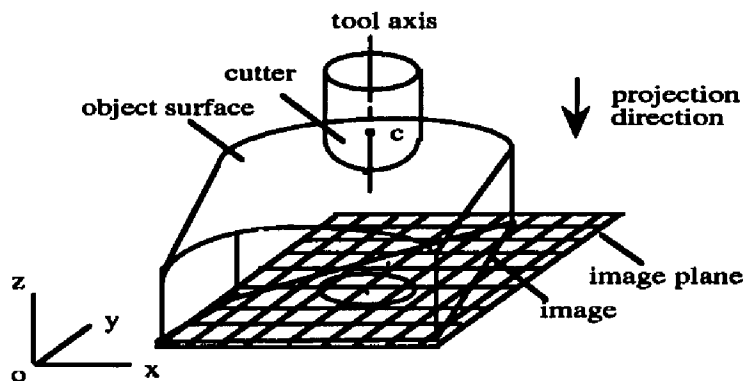


Figure 2.3 Image surface discretization

A typical example is the G-buffer (Geometry buffer) method proposed by Takafumi and Takahashi [25]. A G-buffer is very similar to the Z-buffer for hidden surface elimination in Computer Graphics. The primary difference is that a G-buffer contains more than the z depth for each pixel (i.e., surface normal, patch identifier, etc.). The surfaces are discretized into a set of points by parallel projection of rays onto the surface (Fig. 2.4). Individual CLs are found using the highest point technique, in which the cutter is set to contact the highest surface point (in the z direction) under the cutter. The method is both simple and robust. Interference between the cutter and the adjacent surfaces is eliminated. However, this approach has the problem inherent in image processing: the accuracy of all the images (i.e., z image and offset surface image) depends on the resolution or the sampling density. Since the surface is evenly sampled, a very large number of points may be required to achieve the desired accuracy. Unfortunately, accuracy and tolerance issues are not addressed by the authors.

#### 2.1.4 Object Based Surface Discretization

Tool path generation based on object surface discretization is an alternative method. Our group previously worked on the point and tangent plane methods[27, 28]. In these approaches, an object is approximated by a Surface Point Set (SPS). The density of sampled points obviously affects the accuracy of tool positioning. In order to keep the magnitude of the gouge reasonably small, e.g. 0.05 mm, a large number of closely spaced points must be generated. Densely sampled points result in more computational time but this may be traded off against the simplicity and robustness of the algorithms. Based upon the SPS surface approximation, two kinds of tool positioning approaches are available: point only and tangent plane.

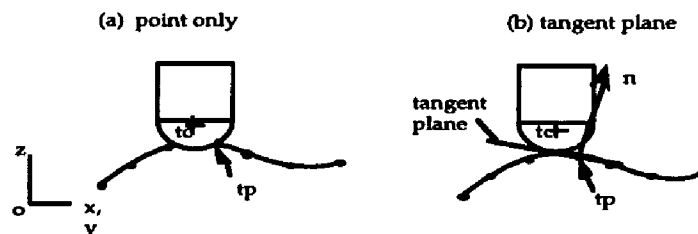


Figure 2.4 Tool positioning with point or tangent plane approach

In the point only approach (Fig. 2.4a), a continuous tool path is approximated by a set of tool locations at discrete locations with small intervals across the surface. Each individual tool location can be found using the SPS as a set of check points to position the cutter. For any tool location, the maximum amount that the tool can protrude into the surface is determined by the point separation and tool diameter. In order to bound the maximum gouging that may occur, the distance between tool locations must be limited. The undesirable effects of such a small increment are that many tool locations are produced and that there is repeated examination of the same points in the SPS. This method is considered a brute force approach. It is slow but robust for all cases, and makes use of a simple surface representation.

Tool positioning with tangent plane (Fig. 2.4b) utilizes additional surface information -- normals to find the tangent plane at the surface points. For a given x, y position, the z component of the position is calculated so the bottom of the tool is just resting on the tangent plane at the point closest to the cutter. As with the points only approach, the distance between tool locations has to be limited in order to bound the gouging that can occur when the tool moves linearly. The advantages to this approach are: first, the tool locations which are generated are more likely to follow the contour of the surface since there is less chance for the tool to drop between points in the SPS; secondly, the number of points needed to represent the surface can be decreased. In both cases, tool paths are parallel across the surface. For efficiency in tool positioning, the points (SPS) are localized with a "bucket strategy". For each tool position, only the points in the buckets that overlap with the cutter shadow are taken into account. The cutter shadow is the projection of the cutter onto the xy plane (assuming that the cutter axis is parallel to the z axis). Step-forward and stepover increments are derived based on the cusp height calculation and the maximum possible gouge amount respectively.

From the above discussion, the common features of the methods based on surface discretization are as follows:

1. These methods can be applied to very complex surfaces.
2. The algorithms for tool positioning are relatively simple.

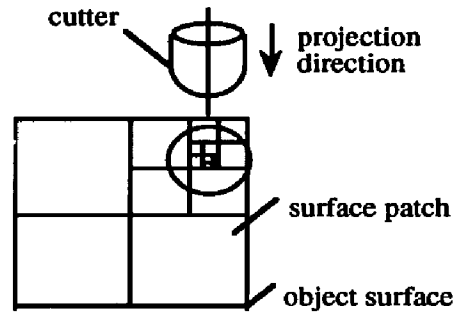
3. Computational time is proportional to the number of sampled points.
4. Interference between the tool and adjacent surfaces is within the desired tolerance.
5. The density of sampled points greatly affects the accuracy of tool positioning.

### **2.1.5 Polygonal Approximation**

A polygonal approximation of the surface has some advantages compared to the point based methods. It has  $C0$  continuity and accuracy of the surface approximation does not depend on the tool size or shape. If the cutter is positioned to touch the polygons, the error for an individual tool position can not exceed the deviation from the surface approximation. Though more time is required for positioning a cutter on a polygon than on a point, the number of polygons considered for each tool position is less than the number of points. This trade off works in favor of the polygon method when the object surface is relatively flat. A variety of methods have been investigated which differ both in the approach to obtaining the polygons and in tool positioning. In the following section, some typical approaches are discussed.

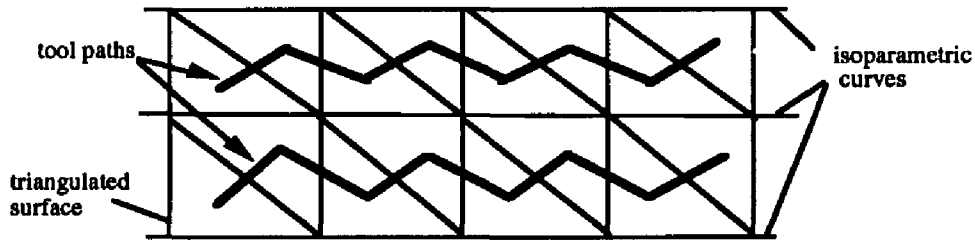
Arbab<sup>[29]</sup> represents an object surface with a quad tree of rectangular surface patches. The object surface is at the root, and the patches and subpatches are at nodes or leaves. For each tool position, the whole surface is taken into account and the whole tree is searched down one level for the (sub)patch that is closest to the cutter in the cutter projection direction (Fig. 2.5). The cutter is assumed to be very far away and will move towards the surface in some projection direction. The closest patch is subdivided into four subpatches if the deviation of this patch is beyond the tolerance. Similarly, recursive subdivision and the search for the closest subpatch among the four are carried out. The subpatch closest to the cutter in the projection direction is thus found. If its deviation is within the tolerance, the subpatch is then divided into two triangles, and the one closer to the cutter is used to position the cutter. To save calculation time, the subdivision information between successive tool positions is saved. Since the whole surface is considered for each tool position, the interference between the cutter and adjacent surface patches is avoided. Tool paths are straight lines on the xy plane.

Although, in this scheme, the searching tree is pruned (for example, in the closest patch only the closest subpatch is subdivided), the whole process is repeated for each tool position. It would seem to make more sense to perform the subdivision once and save all the polygons for future use.



**Figure 2.5** Tool positioning with quad trees

Duncan <sup>[30, 31]</sup> describes an alternative approach for polygonalized surfaces, which he calls polyhedral machining. The surface is subdivided into a set of small patches by isoparametric curves, then each smaller patch is further divided into two triangles. The localization is realized by testing only the triangles under the cutter. The highest point technique is used to set the cutter location in the z direction. In the highest point technique, the point or triangle under the cutter that sets the cutter at the highest position is used to set the cutter center location. Tool movements are forced to be along the centroids of the triangles between consecutive parametric curves. The resulting tool paths are zigzag line segments between isoparametric curves (Fig. 2.6). If the intervals between isoparametric curves are very small, the zigzag tool paths is almost the same as the tool paths along the parametric curves. Therefore, the problems related to parametric indexing such as unevenly distributed tool paths are present here also. Moreover, using isoparametric curves to triangulate the surface results in extra triangles in some areas especially on irregular surface patches, costing extra memory space and computational time. Duncan does not address the important issues of how to triangulate a surface such that a required level of machining accuracy is maintained.



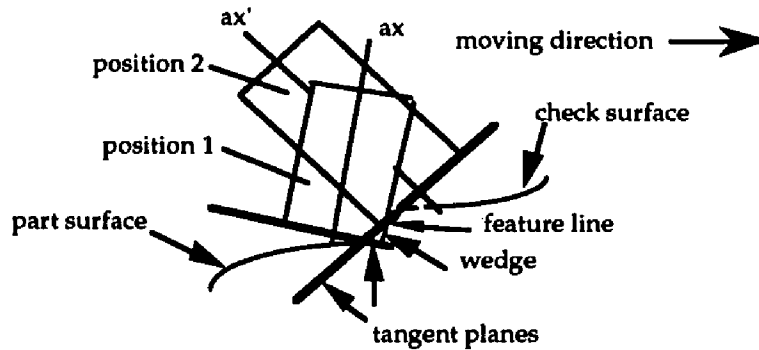
**Figure 2.6** Tool paths between two consecutive isoparametric curves

The UNH/Dartmouth group has implemented a tool path generation system based on an alternative surface triangulation scheme <sup>[32]</sup>, in which an object surface is approximated by a *surface triangle set (STS)*. The results show that for surfaces that are not very curved this method has an advantage over the point or tangent plane method in both calculation time and memory space. Details are discussed in Chapter 3.

## 2.2 FIVE AXIS TOOL PATH GENERATION

A five axis tool has two rotational degrees of freedom so that the tool can be oriented at an arbitrary angle with respect to an object surface. In five axis milling, the tool axis is usually oriented close to the normal of the surface. A flat-end tool can be tipped at an angle so that the machined surface conforms closely to the design surface. Therefore, five axis NC programs should be able to achieve acceptable surface quality with fewer tool paths. However, the two additional degrees of freedom greatly complicate the algorithms for tool path generation. The few published papers on this subject are discussed here.

The five-axis tool path generation approach proposed by Chou<sup>[33]</sup> is based on parametric indexing. By using the curvature information, a B-spline surface is divided into convex and concave regions. The concave regions are called feature lines and they must be very narrow compared with the cutter radius. The number and orientation of the tool paths are defined by the user. The interference between a flat-end cutter and adjacent surfaces is detected with the three surfaces (part, drive and check surface) similar to APT in tool positioning.



**Figure 2.7** Cutter touches the edge with its leading or trailing edge

However, this approach is limited to very specific types of surfaces because of the assumptions that only one check surface exists at a time, that both the part and check surfaces must be convex, and that the intersection angle of a part surface with a check surface must be larger than  $90^\circ$ . Each tool path is arranged along an isoparametric curve of the surface and the path must be across the feature lines. Therefore, each tool path is composed of some segments separated by the feature lines. Tool path segments at non-feature-line regions are obtained by parametric indexing. Tool path segments across a feature line are derived through an offset method. The tangent planes at the end points of a feature line along a parametric curve are used to approximate the feature line. The offset technique is used to find the tool positions where the cutter touches the intersection line of the part surface and the check surface with its leading and trailing edge, respectively. In doing so, the wedge under the cutter and that above the part and the check surface is removed accurately (Fig. 2.7). The final tool path along an isoparametric curve is the combination of the tool path segments divided by the feature lines.

As in three axis machining, using parametric indexing makes tool path planning quite simple. With the offset technique, the region around the intersection of a part surface and a check surface is machined more accurately. However, there still exists the problem of unevenly distributed tool paths.

Jensen and Anderson <sup>[34]</sup> apply concepts of differential geometry to tool positioning. Local curvature properties are used to help set an individual tool position. By matching the curvatures of the silhouette of the cutter to the curvatures of the surface at a touch point,

excess or gouging amounts of material in the vicinity of a touch point can be mathematically determined and decreased or eliminated. The silhouette is the projection of the profile formed by the cutter bottom moving along the cutting direction.

Though this positioning algorithm is plausible, global interference is not prevented. Moreover, to apply this algorithm an object surface must have at least C1 continuity at a given touch point, which is too restrictive for surface design. However, this algorithm could be used to set an initial tool position. For error free tool paths these initial positions need to be further verified and corrected.

## **2.3 NC PROGRAM SIMULATION, VERIFICATION AND CORRECTION**

To avoid the risk of failure, tool paths should be simulated before actual machining. As described by Jerard<sup>[35]</sup>, the process can be divided into three steps: simulation, verification and correction. In simulation, the swept volume of each tool movement in a CLDATA (cutter location data) file is modeled and a geometric model of the workpiece is modified by subtracting the swept volume. A swept volume is the space taken by a tool moving from one position to another. In verification, the comparison between the final workpiece model and a geometric model of the part is carried out. The workpiece is acceptable whenever it is within certain error bounds. In the correction step, tool movements are modified if they cause unacceptable errors. Here the three types of simulation approaches are briefly discussed: simulation and verification based on solid modeling, image based surface discretization, object based surface discretization and polygonal approximation.

### **2.3.1 Simulation Based on Solid Modeling**

Simulation based on solid modeling techniques is achieved by a Boolean subtraction of the swept volume of the tool movement from the workpiece model. The verification is accomplished by performing a Boolean difference between the workpiece model and the desired part model<sup>[36, 37, 38]</sup>. This scheme offers accurate simulation and verification results but is computationally expensive. The cost of simulation with the CSG approach is reported to be proportional to the fourth power of the number of tool movements<sup>[28]</sup>.



### 2.3.2 Image Based Surface Discretization

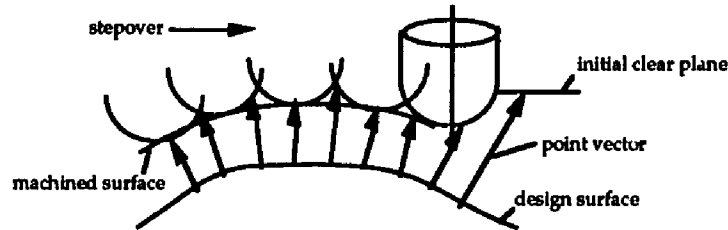
In image-based simulation systems, an object surface and/or the tool is approximated by points sampled by projecting the pixels of an image plane back to the object along a view-direction. An extended z-buffer is used to store entry and exit values of the line from each pixel through the object. NC program simulation is performed by subtracting the swept volume from the z-buffers. Verification is accomplished by performing the Boolean difference at each pixel of the design part and the workpiece.

Van Hook <sup>[39]</sup> approximates the swept volume with the tool image pixels at interpolated steps between tool positions. To reduce computation time, a precalculated tool image is used. At each step the tool pixels are subtracted from the part image pixels. The precalculated tool pixels restricted this approach to three axis cases. Atherton<sup>[40]</sup> extends Hook's method to accomplish five axis tool path simulation by integrating the tool pixel sampling with the subtraction operation.

Wang <sup>[41, 42, 43]</sup> has derived a mathematical equation representing the surface of a swept volume formed by a tool movement. To simplify the verification process, the 3D analytic swept volume surface model is approximated by a set of polyhedra. Then the scan spans for each scan line intersecting the polyhedra set are found based on a scan-line algorithm, and the intensity of the extended z-buffer is calculated based on the depth at each pixel. Simulation is accomplished with a method similar to Hook's. Position verification is achieved by comparing the pixels of images of the part and the workpiece.

Oliver <sup>[44, 45, 46]</sup> has applied the pixel sampling scheme to obtain surface points. Instead of converting the swept volume into 3D image pixels, he decomposes the surface of the three axis swept volume into a set of basic surfaces such as plane, cylinder and sphere. He then applies the "point-vector" technique first proposed by Chappel<sup>[47]</sup>, in which the part surface is represented by a set of vectors normal to the surface with the tips of the vectors touching the inside of the initial workpiece. The lengths of vectors are reduced when they intersect the tool swept volumes. The final workpiece is represented by a set of vectors (Fig. 2.8). The verification is trivially accomplished by comparing the final vector lengths

to the allowable tolerance. Narvekar and Oliver<sup>[48]</sup> have extended Oliver's system with five axis simulation and verification capability. They represent the surface of a five axis swept volume with an analytical mathematical model. The intersections of the normals with the surface is accomplished with an iteration algorithm.



**Figure 2.8** Simulation based on point-vector method

Anderson<sup>[49]</sup> divides the base of the workpiece into squares and a depth or height value is stored in each square. Then simulation is accomplished by keeping track of the swept volume subtracted from the heights of the squares. If the base of the object is regarded as a view plane and each square as a pixel, this method is quite similar to the view-based methods described above.

### 2.3.3 Object Based Surface Discretization

The object-based point method<sup>[1, 35, 50, 51]</sup> differs from the above approaches by using surface point sampling. Instead of using an image-plane, sample points are selected on an object surface with a spacing which depends on surface curvature, the required accuracy, and the tool size and shape.

In all these systems based on surface discretization, simulation and verification are achieved through the intersections of swept volumes with straight lines (i.e., point normal vectors). Direct pixel comparison of the workpiece with the design model (Hook's, Atherton's and Wang's) can produce a good volumetric approximation of the removed material. Generally, however, image-based verification may not be adequate since it is view dependent so that errors invisible in the viewing direction can not be detected. Object-based point-vector approaches do not share this limitation.

### **2.3.4 Polygon Based Simulation**

Compared with a polygonal approximation, surface discretization has some drawbacks. The required density of points is affected by the tool shape and size. Lots of sample points are necessary even for a very flat surface. However, simulation and verification based on a polygonal surface approximation may require more CPU time. Ongoing research at Dartmouth College <sup>[52]</sup> shows that the advantages of three axis simulation and verification based on a polygonal approximation compared to a point discretization depend on the flatness of the surface. For relatively flat surfaces, simulation based on a polygonal approximation is much faster than the point method, especially when flat-end tools are used.

## **2.4 SURFACE QUALITY AND MACHINING EFFICIENCY**

From the previous discussion, it is evident that the cutter selection plays an important role in an NC system. For example, it affects the machining speed, surface quality, and the efficiency of tool path generation and simulation. Flat-end and ball-end cutters are commonly used tools in sculptured surface machining. For rapid and efficient machining of an object, the cutter shape should match the surface shape as closely as possible. In general, a flat-end cutter provides fast machining speed, and good quality for convex surfaces, but poor quality for concave surfaces. For a ball-end cutter the opposite is true.

The effect of a ball-end cutter can be realized by tilting a flat-end cutter in a five axis NC machine. Vickers<sup>[53]</sup> describes in detail about the ratio of the effective cutter radius to the actual cutter radius as a function of tool tilting angles. Schmid<sup>[54]</sup> describes the relationship among the tilting angle, feedrate, and cusp height. He shows that when a flat-end cutter of 40 mm in diameter is tilted by 25 degrees with a stepover of 4 mm, the resulting cusp height is 0.012 mm; in contrast, the cusp height is 0.1 mm for a ball-end cutter with the same diameter and stepsize. This implies that a larger stepsize can be used for a flat-end tool than a ball-end one if the same cusp height is required. The number of tool paths for a flat-end tool is decreased. It is also true that surface finish can be improved

by tilting a ball-end cutter to avoid using the zero-velocity point at the tool tip.

Besides the shape and the number of degrees of freedom of the cutter, the parameters of milling machines (such as the spindle speed, feedrate, cutter deflection, cutter wear, etc.) also play key roles in productivity and surface quality. Parameters such as feedrate and speed are predefined through handbooks<sup>[55]</sup> or based on experience or rules of thumb. Parameters selected in this manner are often far from optimal. To achieve high efficiency (i.e., fully utilize the capability of milling machines), fundamentals of the manufacturing process must be understood. Good NC programs should take into account the following aspects: cycle time, machine capacity, tool life, ease of set up, operation feasibility and flexibility<sup>[56]</sup>.

An optimized NC program is one in which the overall tool path distance is minimized and feedrates are maximize<sup>[57]</sup>. Minimizing the tool path distance is mostly handled in the tool path generation stage. Currently, the optimum feedrate setting is still at the research stage. The goal of most research is centered on the application of adaptive feedrates and spindle speed. One approach is proposed by Fussell and Jerard<sup>[58]</sup>, in which feedrate optimization is accomplished by setting the material removal rate based on volumetric calculation in NC program simulation. Forces acting on the tool are directly related to the material removal rate and tool deflection. Wang<sup>[59, 60]</sup> has also introduced a similar approach based on off-line feedrate preprocessing.

## **CHAPTER 3**

### **THREE-AXIS TOOL PATH GENERATION BASED ON SURFACE TRIANGULATION**

Angleton<sup>[27]</sup> has proven the feasibility of the point-vector and tangent plane approaches to three axis tool path generation and correction. In these approaches, an object surface is approximated by a surface point set (SPS). Point spacing is controlled by the tool shape and size, surface curvature, and user defined tolerances. Tool positions are calculated using a highest point algorithm. Efficiency is achieved with a bucketing scheme and tool positions are incremented in Cartesian space. Tool paths are parallel across surfaces so that the object is machined evenly. Tool path generation based on the SPS is applicable to very complex surfaces (refer to section 2.1 for comments on these approaches).

As described in Chapter 2, a polygonal approximation to an object surface has the advantage that it does not depend on the tool size and shape as the SPS does. Tool positioning error should not exceed the deviation of the surface polygonalization. This chapter presents an alternative tool path generation system based on triangulated surfaces: TRI\_XYINDEX. To shorten the explanation, only the concepts differing from the SPS approach are described in detail.

#### **3.1 SURFACE TRIANGULATION**

The surface triangulation method employed in this work was proposed by Drysdale and Ozair<sup>[32, 52]</sup>. An object surface composed of a set of parametric surface patches is subdivided with a recursive algorithm. The recursion starts with each patch, and ends when the deviation of an individual triangle from the surface patch is within an error bound. The deviations are checked with a heuristic which finds the maximum distance of a set of selected points from the surface. If the deviation exceeds the bound, the surface

patch is further subdivided. To avoid skinny triangles (i.e., with high aspect ratios), edge lengths are confined to a certain range.

### **3.2 TOOL PATH GENERATION OVERVIEW**

In TRI\_XYINDEX, the cutting tool is initially set at a home position which is above the object surface. It is then advanced to move onto the object from an object edge. Four critical issues must be considered in tool path generation: boundary handling, tool positioning, step-forward incrementation and step-over incrementation. The aim of boundary handling is to set tool positions correctly and efficiently at surface boundary areas so that the tool can precisely start a path across the surface without gouging or leaving excess material. Tool positioning attempts to find an individual tool position at which the tool accurately contacts the surface without any interference. Step-forward and step-over incrementations choose the step size between individual tool positions and between paths, respectively. These algorithms determine in large part the accuracy and efficiency of an NC program. Small increments insure accuracy but take longer to calculate and may result in longer than necessary machining time. Good approaches achieve sufficient accuracy with a minimum of CPU and machining time.

Tool paths are currently confined to parallel paths on the xy plane. Step-forward and stepover increments are derived based on the cusp height calculation, similar to the SPS based approach. Tool positions are obtained by incrementing the tool in the x direction. It is possible that some unnecessary tool positions may be calculated. For example, two end positions are enough to represent a tool path across a flat surface. The number of tool positions are minimized with a least squared fitting (LSF) algorithm.

To provide calculation efficiency, a bucketing strategy is used in the tool positioning algorithm. Buckets are small boxes formed by evenly spaced grids superimposed onto the xy plane(Fig. 3.1). The number of buckets is defined to be one half of the total number of triangles with a maximum number of ten thousand. Triangles are sorted into corresponding buckets based on their xy domains as proposed by Ozair <sup>[32]</sup>. It is possible that a triangle covers more than one bucket. To avoid unnecessary calculation, each triangle has an associated flag. At each tool position (i.e., given the x and y components of the tool

center), related triangles are tested only once. The general procedure for tool path generation is listed in Algorithm 3.1. In the following sections boundary handling and tool positioning are discussed.

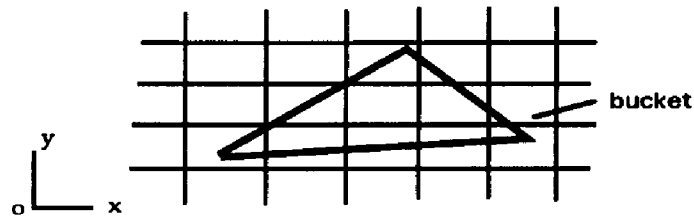


Figure 3.1 Bucketing triangles

**Algorithm 3.1** Tool Path Generation

```

initial: cutting domain (xmin, xmax, ymin, ymax)
  x = xmin
  y = ymin
for y ≤ ymax
  for x ≤ xmax
    if the tool is off the surface
      find the leading edge
      if the lead edge is found
        set tool edge positions
        x = x + Δx
        call Generation
    else call Generation
  calculate new stepover Δy for the next tool path
  y = y + Δy

```

***Generation***

```

for x ≤ xmax
  if the tool is on surface
    set tool position
    x = x + Δx
  else find the trailing edge
    set tool trailing edge positions
end of Algorithm 3.1.

```

### 3.3 EDGE POSITION IDENTIFICATION

In surface machining, a tool often has to move onto or away from surface edges, especially for a surface with holes in it. Proper detection of these edges is very important since rapid tool feedrates can be used when off the surface. Tool edge positions are the tool center locations from which the tool either begins to climb onto or leave the surface. In the point or tangent method<sup>[32]</sup>, a tool edge position is detected by constantly incrementing the tool along a path from a start position to find a position where the status of the tool is changed (i.e., over or off the surface). Then the “exact” tool edge position is derived via a “ping-pong” scheme or recursive subdivision on the increment which causes the tool to change its status. For each subdivision, valid tool positions (i.e., moving the tool over the surface) are calculated and saved in a flow line. The subdivision is ended when the increment between adjacent tool positions is less than a defined value. The last tool position thus obtained is the approximate starting point from which the tool is defined to be over or apart from the surface. Obviously, the accuracy depends on the tolerance used in the recursion. Dense tool positions near edges are used to avoid gouging at edge regions. For a surface approximated by a set of triangles, this method is computationally expensive. Positioning a tool to touch a triangle takes even more time than positioning the tool to touch a point. An alternative edge searching approach is presented which is based on local searching of a set of candidate triangle edges.

#### 3.3.1 Leading Edge

A leading edge position is defined as the position where the tool touches a surface edge (i.e., outer or inner boundary) with its cylindrical side (Fig. 3.2). Assume that a tool is currently off an object surface and it moves along the +x axis. Note that the tool is not over the surface when it is at an edge position. If the z component of the tool at an edge position is set to a default value, only the x component of the tool is unknown before the edge is found. Since the tool touches the edge of a triangle with its cylindrical side, the algorithm for finding an edge position can be simplified by projecting the tool and all the triangles onto the xy plane.



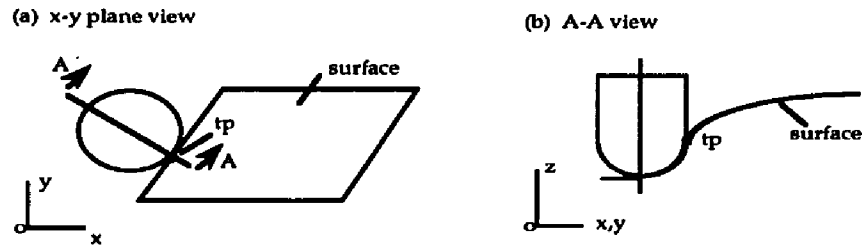


Figure 3.2 Tool edge position

To localize the searching, first a strip is formed with a minimum  $x$  value of the current cutter location ( $X_{ccl}$ ), a maximum  $x$  value of the cutting domain ( $X_{max}$ ), a minimum  $y$  value equal to  $y-r$ , and a maximum  $y$  equal to  $y+r$  (Fig. 3.3). As previously noted the triangles are sorted into buckets. Dense grids provide improved localization but this must be traded off against the need to check more buckets. To select the candidate triangles quickly, only the buckets overlapping the strip are checked. The aim is to find the first column of the buckets in which there are triangles overlapping the strip. Hence, the buckets overlapping the strip are first searched in the  $y$  direction then in the  $x$  direction (i.e., the tool moving direction). If the first column is not found, it means that no surface is ahead of the tool. The next tool position is set at  $X_{max}$  and tool positioning for current path is completed. Otherwise, the bucket index along the  $x$  direction of the first column is recorded in  $XB_{start}$  and searching moves to the next stage.

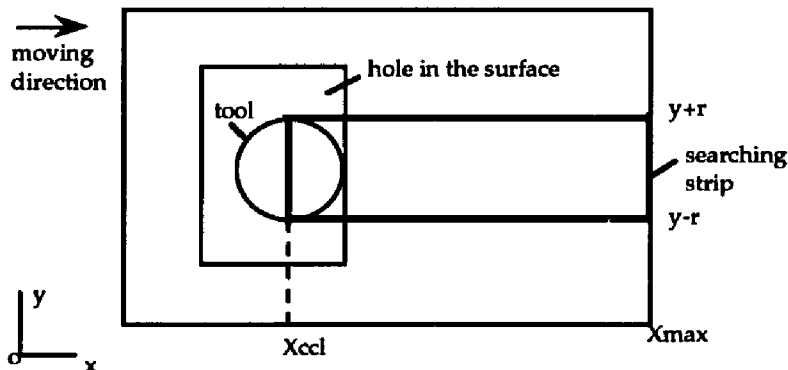
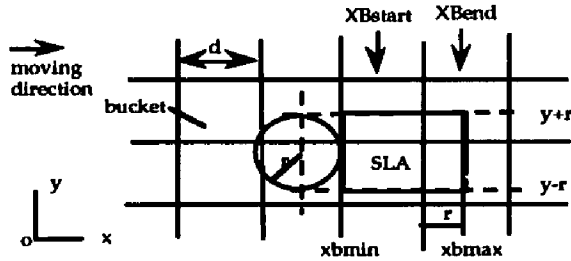


Figure 3.3 Searching Localization

After  $XB_{start}$  is found, an even smaller local area (SLA) is defined by  $xb_{min}$ ,  $xb_{max}$ ,  $y-r$  and  $y+r$  Fig. (3.4), where  $xb_{max} = xb_{min} + d + r$ ,  $d$  is the bucket size along the  $x$  direction and  $r$  is the tool radius.  $XB_{end}$  is the bucket index along the  $x$  direction of the last bucket column covering the SLA. The SLA is the area in which at least one edge of a triangle is first contacted by the tool as the tool climbs onto the surface. For each bucket that overlaps the SLA, if a triangle in the bucket also overlaps the SLA, it is used to calculate a candidate  $x$  position at which the tool first touches the triangle. Note that more than one triangle may overlap the SLA; the final  $x$  value is set to be the one that pushes the tool farthest away from the surface. For example, assuming the tool is moving in the  $+x$  direction, the  $x$  component of the tool at a leading edge position is set to the smallest of all the candidate  $x$ 's which are not smaller than the  $x$  component of the current tool center location.



**Figure 3.4** Small Local Area (SLA)

### 3.3.2 Trailing Edge

The tool eventually falls off at a trailing edge. Care must be taken not to gouge the edge when taking the last tool movement. The size of the gouge depends on the tool path between the last position before falling off the edge and the first position after. To insure accuracy some intermediate positions must be inserted.

Suppose that the tool moving toward the  $+x$  direction is just off the surface. The  $x$  component of the tool center is recorded as  $XL_{last}$  at which the tool is still over the surface

and will be off the surface if an increment is added (i.e.,  $X_{out}$ ). The trailing edge is found in a similar way to the leading edge except for the following differences (Fig. 3.5):

1.  $X_{min} = X_{last}$ ,  $X_{max} = X_{out}$ .
2. The searching direction is opposite to the tool moving direction.
3.  $XB_{start}$  is derived from the larger  $X$  (upper bound) of the first bucket column.
4. An SLA is formed by  $XB_{start}$ ,  $XB_{end}$  (equal to  $XB_{start} - r - x$  bucket size), and the  $y$  domain covered by the tool moving along the tool path.
5. The  $x$  component of the tool at a trailing edge is set to be the largest of all  $x$ 's which are not larger than the  $x$  of the current tool center location ( $X_{out}$ ).

If the tool moves in the opposite direction (i.e.,  $-x$  direction), all the searching orders (both in strip and SLA) are reversed. Details on the searching procedure is listed in Algorithm 3.2.

**Algorithm 3.2** Find a Leading /Trailing Edge

Initial : strip size and location

Find bucket indices for the strip ( $XB_{start}$ ,  $XB_{end}$ ,  $YB_{start}$ ,  $YB_{end}$ )

from  $XB_{start}$  to  $XB_{end}$

if any non-empty bucket is found in the column from  $YB_{start}$  to  $YB_{end}$

form SLA

call *SLA Search*

else increment  $XB_{start}$  to next column

if no column is found

edge is not found, set next tool positioning at  $XB_{end}$

current tool path is finished

*SLA Search*

initial:  $x_{tool}$  and SLA bucket ranges

for each bucket overlapping SLA

for each triangle overlapping SLA

select the side(s) to be tested

set temporary tool edge position  $x$

based on the tool moving direction and edge type (leading/trailing)

$x_{tool} = \min(x_{tool}, x)$  or  $x_{tool} = \max(x_{tool}, x)$

end of Algorithm 3.2

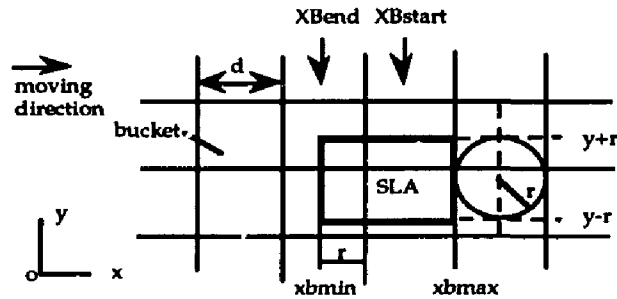


Figure 3.5 Finding a trailing edge position

### 3.3.3 Avoiding Gouge at Edge Positions

A tool moves from a leading edge position to the next tool position at a defined minimum step-forward. This increment guarantees that the material either left or removed by this movement is within a defined bound. Referring to Fig. 3.6, the first two tool positions are derived as follows: the first tool position right after a leading edge position is set equal to the height of the first tool position over the surface.

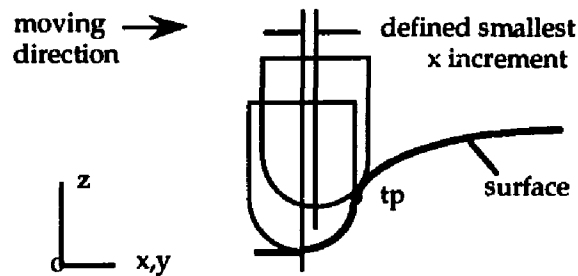


Figure 3.6 First position after an edge position

Tool positions at a trailing edge area are set similarly. Assuming a trailing edge tool position is recorded as  $X_{off}$ , a new tool position is inserted before  $X_{off}$  (Fig. 3.7). The inserted tool position is still on the surface at the minimum step-forward distance from the trailing edge position ( $X_{off}$ ).

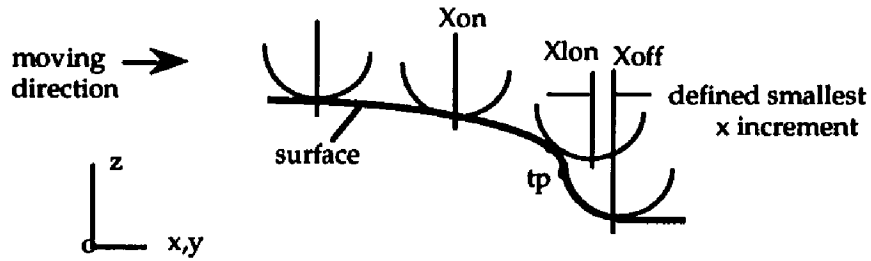


Figure 3.7 Last tool position before a trailing edge

When a tool moves forward at a constant y value, there are at most two possible sides of a triangle that are first touched by the tool. To decrease the number of sides to be considered for each triangle in the SLA, a classification scheme is used if a triangle is not vertical (i.e., parallel to the Z axis). First renumber the triangle with L, M, H by comparing its y component (Fig. 3.8). H is the highest vertex in the y direction, M is the median and L is the lowest. Table 3.1 gives the selection results based on the locations of L, M and H. For example, if M is behind edge LH in the tool moving direction, only edge LH is considered in edge positioning.

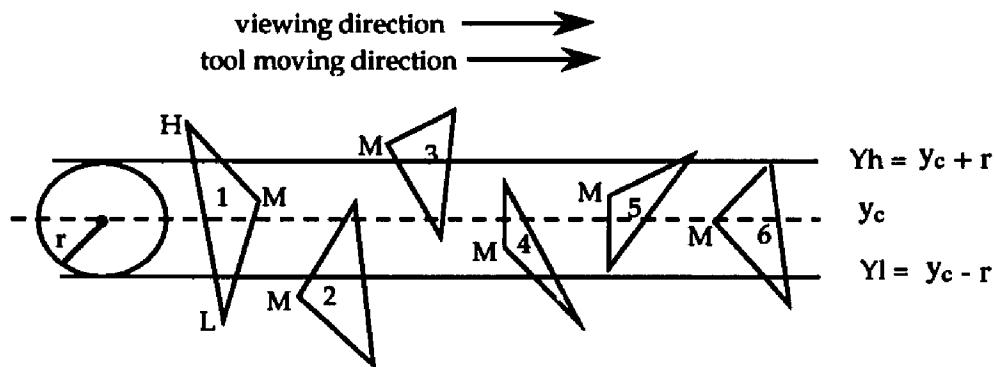


Figure 3.8 Edge Selection

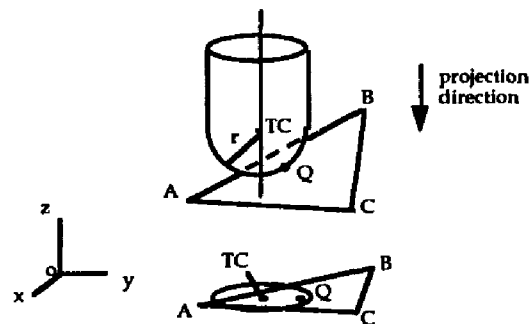
**Table 3.1** Triangle edge selection for tool edge positioning (refer to Fig. 3.8)

case	condition(s)	edge(s) selected
1	M behind HL	HL
2	M under Yl	HM
3	M above Yh	ML
4	M in front of HL & HM $\perp$ line Yl & M under Yc	HM
5	M in front of HL & ML $\perp$ line Yl & M above Yc	ML
6	M in front of HL & not case 4 or 5	HM, ML

Given an edge of a triangle to be touched by the tool side, generally two tangent points are available. The touch point is chosen based on the tool moving direction and edge type (i.e., leading or trailing edge). If the touch point on the unbounded line coincident with the edge lies outside the edge, the vertex of the edge closer to the touch point is selected.

### 3.4 TOOL POSITIONING

Given x and y components of a tool center location, the z component is set with a method similar to the highest point technique. Assume that the tool is above the object surface, then it moves down to touch the triangles beneath it. The tool center location is the position where the tool first touches a triangle. To localize tool positioning, only the triangles in the buckets overlapping the tool shadow are checked. If the tool axis is parallel to the z axis, its shadow (i.e., its projection onto the xy plane) is a disk (Fig. 3.9). The algorithm for local checking is listed below.

**Figure 3.9** Local checking

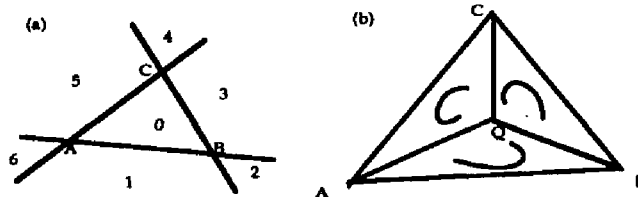
**Algorithm 3.3** Tool Positioning

```

initial : given x and y of the tool center
construct tool shadow
find buckets overlapping the shadow
for any bucket overlapping the shadow
    for any triangle overlapping the shadow
        find tool position z, so the tool touches the triangle
        ztool = max (ztool, z)
end of Algorithm 3.3.

```

Remember that a triangle may be pointed to by more than one bucket, but a flag is set so that candidate triangles are checked only once for each individual tool location. For a tool to touch a triangle overlapping the tool shadow, there exist three possible touching scenarios: touching at a vertex, on an edge, or inside the triangle. To quickly determine the candidate element(s) (i.e., vertex, edge(s), or inside), a searching algorithm is developed which is an extension of Sedgewick's algorithm <sup>[61]</sup> for testing whether a 2D point is inside a 2D polygon. The general idea of the extended algorithm is that the triangle plane is divided into seven sections: numbering from zero to six (Fig. 3.10a). If the candidate TP (touch point) is inside the triangle (including a vertex or an edge), the current tool position is acceptable. Otherwise, the candidate element is a vertex or on an edge.



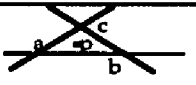
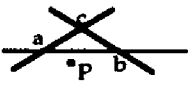
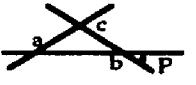
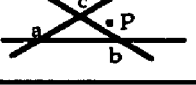
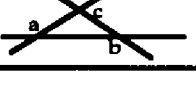
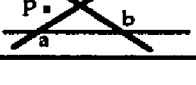
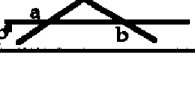
**Figure 3.10** Locating a touch point

The testing and edge selection is achieved as follows. The location of the candidate TP related to the sections in the plane is derived based on the permutation orders. The permutation orders are referred to as counter clockwise (CCW) or clockwise (CW). Based

on the location of TP relative to the triangle, the candidate element(s) can be selected. If P is not a vertex or on an edge, possible permutation sets are listed in Table 3.2.

For example, Q is a candidate TP of the tool on the triangle plane (Fig. 3.10b). Assuming that  $ABC = CCW$ , if  $ABQ = BCQ = CAQ = CCW$ , then Q is inside the triangle. Therefore, the tool position derived from touch point Q is acceptable. In the case that Q is on an edge line but outside the edge segment, the closer edge of the two whose lines do not pass through Q is selected. The mathematical equations for setting a ball-end tool on a triangle are derived in Appendix B.

**Table 3.2** Point location vs. a triangle plane

touch point location	xy view	permutation direction	side(s) to be tested
section 0		abc (CCW) abp (CCW) bcp (CCW) cap (CCW)	
section 1		abc (CCW) abp (CW) bcp (CCW) cap (CCW)	ab
section 2		abc (CCW) abp (CW) bcp (CW) cap (CCW)	ab, bc
section 3		abc (CCW) abp (CCW) bcp (CW) cap (CCW)	bc
section 4		abc (CCW) abp (CCW) bcp (CW) cap (CW)	bc, ca
section 5		abc (CCW) abp (CCW) bcp (CCW) cap (CW)	ca
section 6		abc (CCW) abp (CW) bcp (CCW) cap (CW)	ca, ab

p a (touch) point

CW clockwise

CCW counter clockwise

Note that if the permutation direction of  $\Delta abc$  is CW, the criteria are the same but all the directions are reversed respectively.



### **3.5 TEST CRITERIA, PROCEDURES AND RESULTS**

The above algorithms have been implemented into system TRI\_XYINDEX for three axis tool path generation. The implemented software has been tested with some typical sculptured surfaces and results are compared to those based on surface discretization. The criteria for testing and comparison are categorized below.

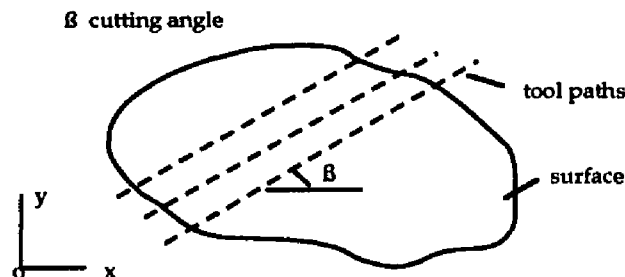
1. **Accuracy** -- The accuracy of the generated tool paths are measured by simulating the machining of the surface. Accurate tool paths produce a surface within the design tolerance zone while inaccurate tool paths result in undercutting or overcutting or both. Generally, overcutting is considered much worse than undercutting since, unlike undercutting, later processing can not compensate.
2. **Speed** -- The speed of each algorithm is measured by the total CPU time taken from startup until the last tool position is output to the CLDATA file. This is the sum of the time taken to create the SPS or STS (surface triangle set) and the time taken to generate tool paths.
3. **Robustness** -- This is measured by running the systems with a variety of surfaces to test if the results are consistent. The test cases may consist of surfaces composed of hundreds of patches and may contain holes or vertical surface patches.

#### **3.5.1 Parameter Selection**

All the software systems developed are designed to be user friendly, with ease of operation and minimum user interaction. System parameters are initialized with a set of default values. These parameters can be adjusted based on specific generation/simulation conditions and levels of accuracy. The following parameters are commonly used.

1. **Cutting Tool Shape** -- Currently limited to ball-end tools.
2. **Cutting Tool Size** -- The largest tool that can accurately machine the surface should be used to provide fast machining, small tool deflection and low wear rate.
3. **Maximum Allowed Gouging** -- This is the amount that a tool can protrude into the surface.

4. **Maximum Cusp Height** -- This is the maximum allowable height of excess material left between adjacent tool paths.
5. **Stock Allowance** -- It is a layer of excess material left on top of the surface after machining. It is usually used in three axis tool path generation for rough cutting.
6. **Cutting Style** -- It is defined to be either box cutting or zigzag cutting. Zigzag cutting removes material in both the -x and +x directions. Box cutting only cuts in one direction and lifts the tool to the clearance plane for the return direction. Surface quality can be improved with box cutting at the cost of increased machining time.
7. **Cutting Direction** -- This is the angle between the x direction and the tool paths (Fig. 3.11).
8. **Y Stepper** -- This parameter can be turned on or off. If it is on, stepsize between tool paths are dynamically derived from cusp heights between a set of sample pairs of tool positions on the same adjacent tool paths.



**Figure 3.11** An example of cutting angle

### 3.5.2 Test Surfaces

Most of the test surfaces were provide by the Body Design Group of Ford Motor Company. The properties of each test surface is listed as follows (please refer to color Plates 1 and 3 at the end of the Dissertation):

1. **convex** -- This surface is composed of three patches forming a convex surface.

- One of the patches is nearly vertical, one is flat and the other is nearly vertical with a small amount of curvature.
2. **concave** -- This surface consists of three patches forming a concave trough. Two of the patches are nearly vertical.
  3. **zip1** -- This is a small surface consisting of three convex patches two of which share an edge with C1 discontinuity. Gouges may occur along the shared edge.
  4. **trunk** -- This surface is the entire trunk lid of a car, consisting of forty-two patches. It is relatively flat at the upper part and gradually curved toward the front.
  5. **bumper** -- This surface is part of a car bumper. It consists of two hundred and sixty-three patches, including patches which are nearly vertical, flat, overlapping, concave, and convex. It also has a hole in the middle of the surface so the algorithms must be able to move on and off the surface multiple times for one pass across the surface.

### 3.5.3 Test Results

Tests were carried on an Silicon Graphics 4D/25 Workstation and the results are listed in table 3.3.

**Table 3.3** Simulation results vs. tool positioning approaches

file name (tool size)	method	triangle number	point number	read time (min)	generation time (min)	total movements	tolerance (mm)
c_vex (D12.7mm)	tangent	11184	10336	0.60	1.01	1850	0.05
	triangle	166	107	0.05	1.02	1557	0.05
c_cave (D12.7mm)	tangent	10992	10199	0.35	1.30	1711	0.05
	triangle	6	12	0.02	0.50	1531	0.05
zip1 (D12.7mm)	tangent	6480	6232	0.32	0.32	2905	0.05
	triangle	486	297	0.01	0.50	2964	0.03
trunk (D38.0mm)	tangent	62438	53271	1.75	8.40	16277	0.05
	triangle	6413	3685	0.58	21.06	12910	0.05
bumper (D12.7mm)	tangent	92807	74182	3.50	8.82	15751	0.04
	triangle	20186	12940	1.20	29.35	15266	0.04

Compared with the tangent plane approach (explained in Section 2.1.4), less memory space and time is required for relatively flat surfaces such as *c\_cave* and *c\_vex*. For slightly curved surfaces such as *zip1*, generation time is close to that based on surface point discretization. For more highly curved surfaces such as *bumper* more computation time is needed. The ratio of points to triangles for the five cases are *c\_cave* (850 : 1), *c\_vex* (97:1), *zip1* (21:1), *trunk* (15:1) and *bumper* (6:1). The corresponding ratios of the tool path generation time (i.e., triangle method time/point method time) are 0.38, 1.0, 1.56, 2.5 and 3.33. The data indicates that the speed of the triangle method relative to the point-tangent plane method depends on the ratio of the number of points to the number of triangles required to accurately approximate the surface.

## CHAPTER 4

### TOOL PATH GENERATION FOR FINISHING SCULPTURED SURFACES

Sculptured surfaces are usually machined to their final shapes by a series of roughing and finishing passes with tools of different sizes. For example, a large cutter is used first for rapid removal of most of the material. Since the large cutter may not fit into the concave areas of the surface it is often necessary to remachine these areas with a smaller cutter. A skilled NC programmer relies on experience to select a sequence of cutter diameters for both rough and finish machining. The programmer tries to select the largest possible cutter since a large cutter removes material more rapidly. However, problems can occur if a tool selected has a radius larger than the concave radius of curvature of the surface. Gouging may take place (Fig. 4.1a) or if the tool path generation algorithm is sophisticated enough to prevent gouging, excess material may be left (Fig. 4.1b).

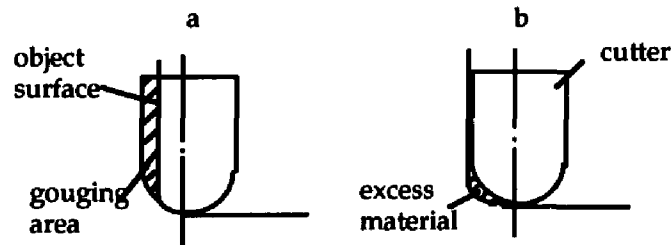


Figure 4.1 Excess material

The STS based method described in Chapter 3, as well as the previously developed SPS methods <sup>[27]</sup> avoid gouging but require remachining of the whole surface (patch) with a smaller cutter to eliminate excess material. A great deal of time is wasted since the cutter also remachines the areas which have already been correctly cut. It would be very desirable

to identify those areas of the surface with excess material automatically and only generate tool paths over those areas.

This chapter presents an automatic tool path generation system for locally finishing sculptured surfaces: FINISH. Within FINISH, undercut areas are automatically identified. Over these undercut areas three kinds of tool paths are planned and generated. Smooth transitions at the edges of uncut areas are accomplished by using an expanded radius in tool positioning. Consequently, the dwell marks at the edges of undercut areas are eliminated. For finishing modestly complex sculptured surfaces FINISH is much more efficient than the traditional approach of remachining the whole surfaces.

#### 4.1 IDENTIFICATION OF UNDERCUT AREAS

We represent an object surface with a Surface Point Set (SPS)<sup>[33]</sup>. The areas with excess material are stored in a file. Note that undercut points are seldom distributed over the whole object surface. These discrete undercut points need to be sorted into a series of continuous areas. Each area will be called an undercut pocket.

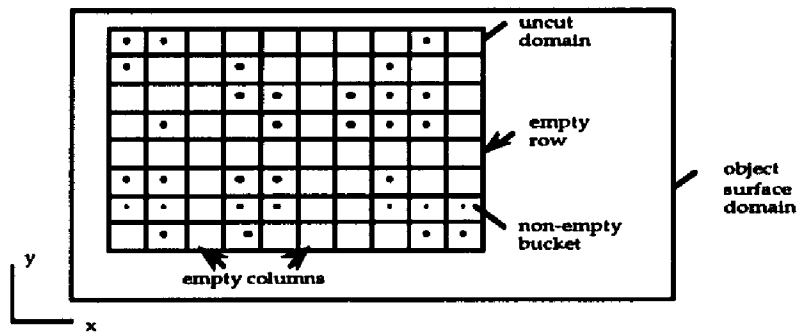


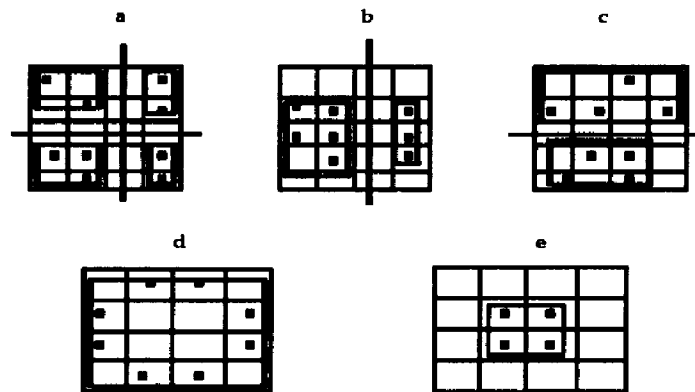
Figure 4.2 Grids superimposed on the undercut domain

Here also, the “bucketing strategy” described in the previous chapter is used to improve efficiency by localizing calculations. Here a similar bucketing method is used to identify undercut pockets. Undercut points are sorted into the buckets according to their x and y components (Fig. 4.2). To save memory and simplify the sorting process, the data structure of each bucket also contains x and y ranges of the points for that particular bucket.

It is possible that some buckets may only contain a single point. In this case, the rectangular domain of that bucket reduces to a point.

Recursive bucket subdivision is carried out after the bucketing. It starts with the overall rectangular domain which contains all the buckets. The rectangular domain is subdivided by the empty rows and columns. As shown in Fig. 4.3, the domain, after subdivision, can be:

1. divided into 4 parts by an empty row and an empty column (Fig. 4.3a),
2. divided into 2 parts by an empty row or column (Fig. 4.3b and 4.3c),
3. unchanged (Fig. 4.3d),
4. reduced to a smaller one (Fig. 4.3e).



**Figure 4.3** Possible cases in bucket subdivision

These subdivided areas are called undercut subdomains. The row and column indices are stored for each subdomain. The subdivision is continued until no empty bucket rows or columns are in the subdomains. These subdomains are stored in a linked list. Each undercut pocket contains the row and column indices used for path planning. The basic recursive subdivision algorithm is listed in Algorithm 4.1. The accuracy of approximating undercut areas with buckets depends on the bucket sizes. Smaller bucket sizes result in a more accurate approximation but more CPU time is required. Currently, the bucket size is defined to be 1.5 times the cutter diameter.

**Algorithm 4.1 Recursive Bucket Subdivision**

Start with the whole bucket domain

***BucketDivision***

row\_break = *FindEmptyRow*

column\_break = *FindEmptyColumn*

if row\_break and column\_break

    divide the bucket domain into four parts and call *BucketDivision* on each part

else if row\_break or column\_break

    divide the bucket domain into two and call *BucketDivision* on each part

else save this domain as a pocket

***FindEmptyRow / FindEmptyColumn***

Start with first row/column

while empty row/column at the boundary of search domain

    reduce the search bucket set by one row/column

while not empty

    increment

return breakpoint (i.e. dividing point)

end of algorithm 4.1

**4.2 TOOL PATH PLANNING**

Once the pockets have been identified, an efficient plan for machining them must be formulated. The goal is to remove the excess material as completely as possible in the shortest time. In the following sections three different approaches are compared.

**4.2.1 Block Cutting**

In block cutting, tool paths are generated over the whole rectangular domain of an undercut pocket, as shown in Fig. 4.4. The domain is obtained by searching all the rectangles stored in the non-empty buckets. In our simulation system, a point represents a small local area. It is possible that points may be located at the edges of the rectangle domain. To machine these undercut areas completely, the rectangle domain is enlarged in -x and +x directions by 75% of the cutter diameter, and in -y and +y directions by the



default stepover distance (assuming that tool paths are parallel to the x axis). The enlargement in the y direction ensures that no excess material will be left at the sides of the pocket parallel to the x axis. The enlargement in the x direction is designed to properly handle the entry and exit of the cutter, which is discussed below.

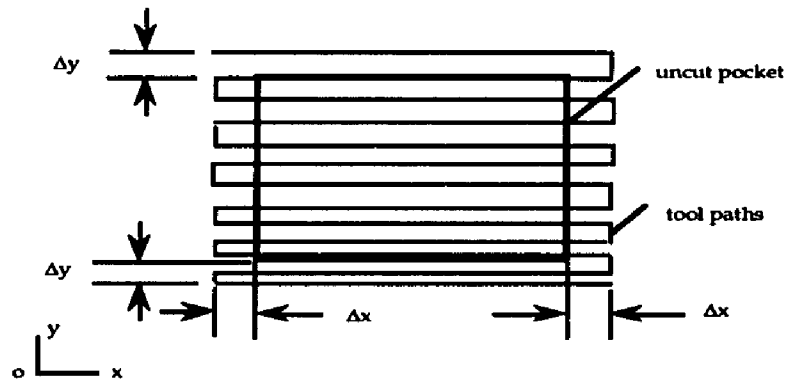


Figure 4.4 Block cutting

#### 4.2.2 Outer-Pruning Cutting

The block cutting style works well for pockets that have a high ratio of undercut area to finished area. However, if the undercut area is distributed as shown in Fig. 4.5, efficiency can be gained by deleting the unproductive portion of the tool path at the end of each row. We call this approach “outer-pruning”.

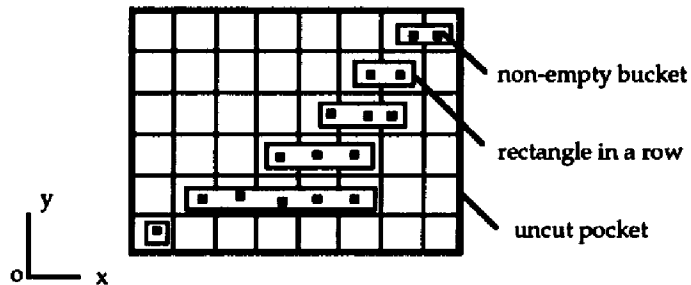
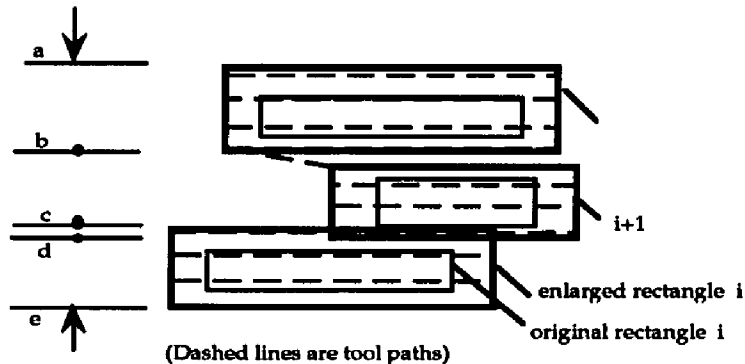


Figure 4.5 Outer-pruning cutting

This algorithm differs from the block cutting in that the outer empty buckets are pruned from the undercut domain. The buckets in a pocket with the same row index are considered as a unit. The rectangular domain that covers the points in the bucket row is calculated individually, as shown in Fig. 4.5. The rectangular domain that covers the undercut points in a bucket row is stored in a linked list. As in block cutting, each rectangle in the linked list of a pocket is enlarged in  $\pm x$  directions by 75% of the cutter diameter and in  $\pm y$  direction by the default steppover. To completely machine the undercut areas the tool paths in the vicinity of the interior edges are defined to be the union of the paths over the consecutive rectangles (as shown in Fig. 4.6). The procedure is listed in Algorithm 4.2.



**Figure 4.6** Tool paths for outer-pruning cutting

**Algorithm 4.2** Outer-Pruning Tool Path Generation

(assuming the cutting direction is along the  $+x$  axis)

for each pocket in the undercut pocket list

for each bucket row in the pocket

generate tool paths

if current bucket row overlaps next

find the length union of the two bucket rows

generate tool paths between them

end of Algorithm 4.2

### 4.2.3 Interior-Pruning Cutting

The outer-pruning method provides a minimal improvement in efficiency when the undercut areas form closed loops (Fig. 4.7). Perhaps the easiest way to increase efficiency in such cases is to lift the cutter when it passes over the interior empty bucket areas. The cutter has to be lifted several times along each tool path and even though rapid feedrates can be used, such paths are not desirable. An alternative method called interior-pruning cutting is proposed for this case.

In this approach, tool paths which have been outer-pruned are further subdivided into several segments. These segments are then connected based on the distances between adjacent path segments in both the x and y directions. The rectangular domains of the non-empty bucket row segments are stored in a linked list. These segments are linked in the order of increasing or decreasing x value. Two consecutive bucket row segments are merged if the space between them is less than twice the cutter diameter. This avoids excessive lifting of the cutter.

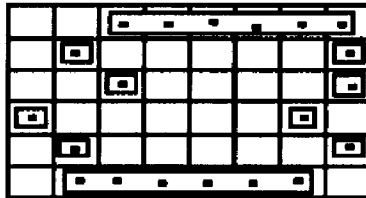
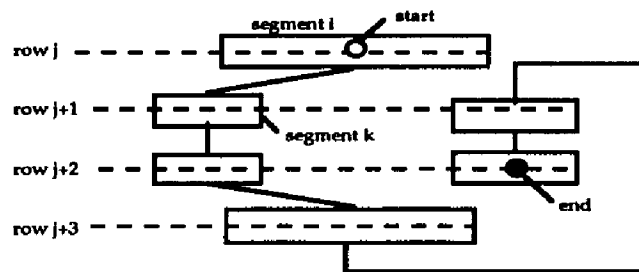


Figure 4.7 interior-pruning cutting

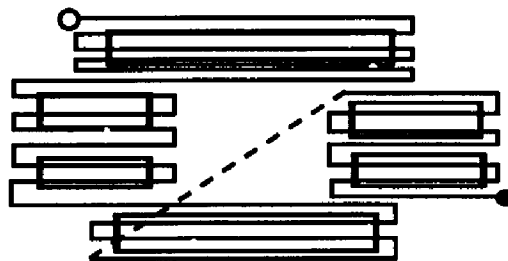
Next, the bucket row segments are sorted into a list based upon their x and y ranges. This is called the *guidance list*. A sorting example is shown in Fig. 4.8. The active element is the  $i$ th segment in row  $j$ . This segment is copied to the guidance list. The active element is compared with the segments in row  $j+1$  on the x range and distance in the y direction. If the x range of the active element overlaps with that of element  $k$  in row  $j+1$  and the distance in the y direction is less than a pre-defined distance, segment  $k$  becomes the active element and is moved into the guidance list. The pre-defined distance is set to twice the distance between tool paths in the y direction. When an element becomes the

active element, this element is moved from the bucket row to the guidance list; therefore, the segment list in a bucket row is updated each time. The process is continued to the next row ( $j+2$ ) and so on. If no segments in the next row are eligible to be added, a flag is set in the last segment in the guidance list. This flag is used during tool path generation to indicate that the cutter must be moved to the clearance plane before moving to the next set of segments. The sorting is then restarted with segment  $i+1$  in row  $j$ . The procedure is repeated until no segments are left.



**Figure 4.8** An example of bucket row segment sorting

Tool path planning now proceeds in the same manner as in outer-pruning cutting. Segments in the guidance list are enlarged and tool paths are arranged among them. Each segment is treated like a bucket row in outer-pruning cutting. However, some tool positions must be inserted after finishing a tool path on a segment with a lifting flag. These tool positions guarantee that the cutter does not gouge the part surface when it moves to the next tool path. An example of such tool paths is shown in Fig. 4.9, where the dashed line indicates that the cutter is lifted. Algorithm 4.3 lists the basic procedure.



**Figure 4.9** Tool paths for interior-pruning cutting

**Algorithm 4.3** interior-Pruning Tool Path Generation

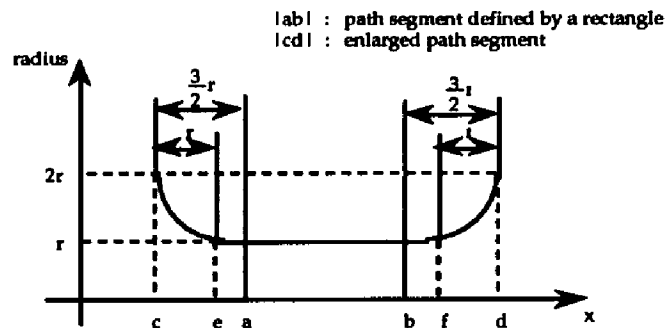
```

for each pocket
  for each segment list
    for each segment on list
      generate tool paths
      if current segment overlaps any segment in next list (along x)
        find union of the two segments
        generate tool paths between
    end of Algorithm 4.3

```

**4.3 DWELL MARK ELIMINATION**

Generally, an undercut pocket is only a portion of an object surface. If a cutter comes straight down onto the surface to machine the pocket, dwell marks are left on the surface at the start and end positions. As described in the above section, rectangles used to define the lengths of tool paths are enlarged by 75% of the cutter diameter in -x and +x directions. For smooth entry and exit, an expanded cutter radius is used when the cutter is in the vicinity of the start and end positions along each tool path. The cutter radius is forced to follow the pattern as shown in Fig. 4.10, where the expanded cutter radius is defined by the parabolic curves at the start and end positions along each tool path. The actual cutter size is used when the cutter is between point e and f (Fig. 4.10). The contact points are calculated with an expanded cutter radius. During machining, the actual distance from the center of the cutter to the surface contact point also follows the curve in Fig. 4.10.



**Figure 4.10** Radius vs. tool locations

For example, in Fig. 4.11,  $r$  is the actual tool radius and  $r_i$  ( $i = 1,2,3$ ) is the expanded tool radius defined by the curve in Fig. 4.10. Using  $r_i$  in the tool positioning algorithm while using  $r$  in the actual machining, the minimum distance between the surface and the tool center is also a parabola which is marked as the boundary of tool movements in Fig. 4.11. Algorithm 4.3 lists the algorithm for tool radius selection. An overview of the finish system is illustrated in Fig. 4.12.

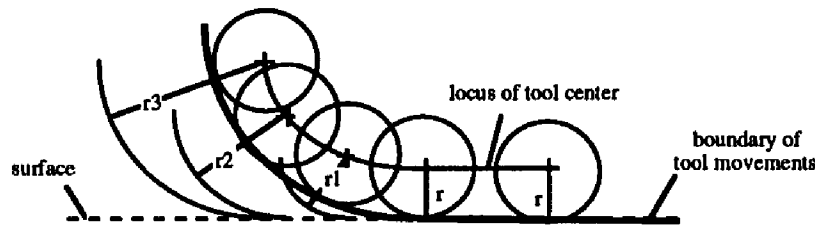


Figure 4.11 Positioning with expanded tools

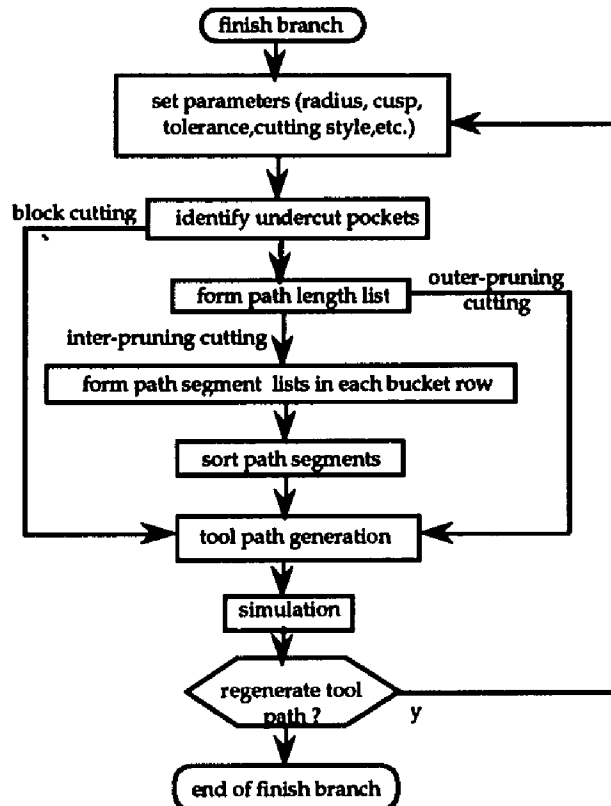


Figure 4.12 Structure of the finish machining system

**Algorithm 4.4 Tool Radius Slection**

Assuming  $r$  is the actual tool radius  
 for each tool path  $cd$  (Fig. 4.10)  
   divide the path into 3 segments ( $ce$ ,  $ef$ ,  $fd$ )  
   if tool location in ( $c,e$ )  
     expanded radius =  $r + (x - e)(x-e)/r$   
   else if tool location in ( $e, f$ )  
     expanded radius =  $r$   
   else if tool location in ( $f, d$ )  
     expanded radius =  $r + (x -f)(x -f)/r$   
 end of algorithm 4.4

**4.4 TEST RESULTS**

The above algorithms have been implemented on a Silicon Graphics 4D/25 Workstation and evaluated with some typical sculptured surfaces. Testing criteria for the finishing system are the accuracy, speed, robustness, storage requirements (refer to section 3.6 for details) and machining efficiency which is reflected by the machining time measured with the total tool travel distance divided by the feedrate. The test surfaces were provided by the Body Design Group of Ford Motor Company. The features of the test surfaces are listed below:

1. **Zip1** is a small test surface consisting of two convex patches which share an edge. Excess material is left along the shared edge (refer to Color Plates 3 and 4).
2. **bumper** is part of a car bumper. It consists of two hundred and sixty three patches, including patches that are nearly vertical, flat, overlapping, concave and convex. This surface also has a hole in the middle (refer to Color Plates 7 and 8).
3. **z6324r** is an automotive quarter panel. It consists of thirty-seven patches. Excess material resides at the concave transition area from the side to rear windows (refer to Plates 5 and 6).
4. **corner** is the inner panel used for the interior of a car hood. It is a very complex surface, consisting of one hundred and sixty-seven patches. It also contains a variety of patches with many shared and unshared edges (refer to Color Plate 2).

Results for the test cases are listed in Table 4.1. The implemented methods generate fewer tool movements and require less machining time than the traditional method (whole surface machining). Among the three algorithms (block, outer-pruning, and interior-pruning), the interior-pruning cutting style generates the fewest tool movements. Block cutting generates the largest number of tool movements.

**Table 4.1** Simulation results vs. cutting styles

case	file	diameter (mm)	cutting style	generation time (sec)	total tool movements	machining time (min)
1	zip1	12.7	block	3	533	1.28
			outer-pruning	3	533	1.28
			inter-pruning	4	533	1.28
			traditional	20	3029	8.16
2	bumper	12.7	block	390	4804	16.48
			outer-pruning	330	4279	14.34
			inter-pruning	303	4101	13.15
			traditional	520	15365	52.70
3	z6324r	12.7	block	40	1163	4.71
			outer-pruning	40	1103	4.09
			inter-pruning	40	980	3.61
			traditional	200	6310	48.67
4	corner	12.7	block	852	10748	37.19
			outer-pruning	684	2068	33.56
			inter-pruning	622	9133	32.43
			traditional	962	13597	40.81

Generation and Cutting conditions include:

1. Cutter diameter for (first) rough cutting (mm):  
25.4 for corner, 38.0 for bumper, 76.0 for zip1 and z6324r
2. Cusp height used in stepsize calculation for rough cutting: 0.2 (mm)
3. Undercut points are the points where excess material is not less than 0.26 (mm)
4. Feedrate for machining time estimation: 40 (inches / min.)

In our post-processing algorithm, the machining time is estimated by dividing the total length of the tool movements by the feedrate. In outer-pruning and interior-pruning, if the distance between two path segments is greater than the default stepsize, the cutter is lifted to a safe clearance plane at the end of each tool path before it moves to the next one. This movement could be carried out at a fairly fast speed (for example, four times the normal machining speed). Therefore, the actual machining time for interior-pruning cutting and outer-pruning could be less than that listed in the table.



**Corner is a very complex surface with many concave areas. Large portions of the object surfaces need to be remachined. Therefore, the saving in machining time compared with the traditional method is not as evident.**

## **CHAPTER 5**

### **FIVE-AXIS TOOL PATH GENERATION BASED ON SURFACE POLYGONALIZATION**

Five axis machining has the potential to provide good surface quality and fast machining speed. A five axis program consists of a list of discrete tool positions and angular orientations. Tool paths between the positions are usually interpolated linearly. To provide acceptable NC programs, good algorithms for tool positioning and incrementing are critical.

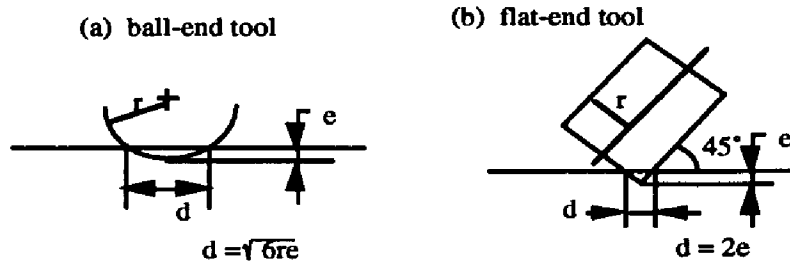
This chapter describes methods for five axis tool path generation. Tool paths are generated from a set of cutter contact (CC) points on the surface. The CC points are obtained with the aid of a set of edge lists (also called segment lists) derived from the intersection of cutting planes with a triangulated surface approximation. Increments between cutter center locations (CLs) are controlled by the increments between CC points as well as the normals to these CC points. A notable advantage of generating CL data from CC data is that the properties of the polygonal approximation can be directly used to set the step-forward increments. Finally, the tool angular orientation is adjusted to avoid gouging the surface. These algorithms have been implemented in the FIVEX\_INDEX system. The basic procedures are: 1. surface polygonalization; 2. CC data generation, including overlapping surface handling, edge detection and step-forward determination; 3. CL data generation; 4. tool position correction, including heel clearance check for gouge avoidance in concave areas or adjacent walls, etc.

#### **5.1 SURFACE APPROXIMATION**

An object surface is usually defined by a set of parametric surface patches. In FIVEX\_INDEX, an approximate representation of these surface patches with a set of

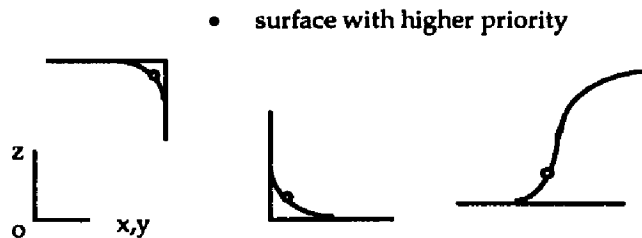
triangles is used. The triangle set is generated by the recursive subdivision method discussed in section 3.1. The triangular approximation makes it possible to perform global interference checking.

A polygonal approximation of a surface requires fewer triangles than the points of an SPS for equivalent accuracy, especially if a flat-end tool is used. For example, if the allowable approximation error is 0.001 inch and the tool diameter is a quarter inch, a one inch by one inch flat rectangular surface patch requires about 333 points for a ball-end tool and 250,000 points for a flat-end tool<sup>[50]</sup> (Fig. 5.1a and 5.1b); it only requires two triangles for a triangular approximation.



**Figure 5.1** Relationship between point spacing ( $d$ ) and approximation error ( $e$ )

In sculptured surface design, it is often necessary to add fillets to create a smooth transition from one surface patch to another<sup>[62]</sup> (Fig. 5.2). This creates a non-unique representation in overlapping areas which can cause problems for NC tool path generation. It would be desirable to remove these inconsistencies but this can be very time consuming. We have developed algorithms which can generate tool paths as automatically as possible and methods for dealing with overlapping surfaces are explained in section 5.2.1.



**Figure 5.2** Examples of overlapping surfaces

## 5.2 CC DATA GENERATION

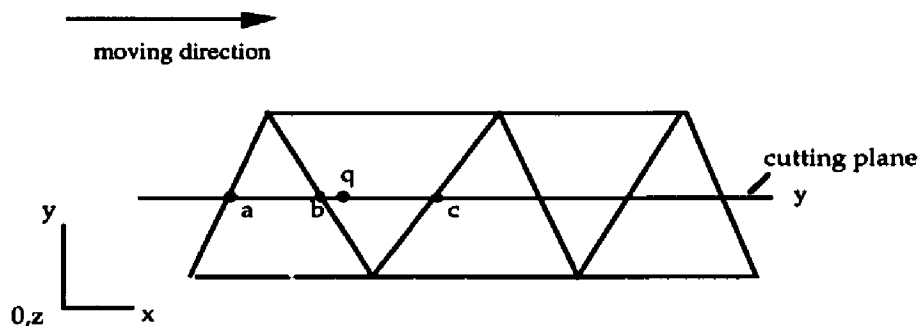
Cutter Location (CL) data for five axis machining may be generated with two different approaches: direct CL generation, and generating CLs from CC points. In the direct CL generation, a tool is dropped onto the surface with constraints on the CLs (e.g., they must be located in a vertical plane) and CC points are then calculated. Three axis SPS and STS methods described in the earlier chapters use this method. For five axis machining the approach is complicated by difficulties in finding the contact points between the tool and the surface, and also by finding the best orientation of the tool. An alternative approach is to generate CLs from CCs. This approach can be used to easily set up the tool position and orientation, but involves complicated interference checking between the tool and the surface. This is due to the fact that CLs generated directly from CCs can not guarantee gouge free tool positions.

The parametric indexing method proposed by Chou, *et al.* (see section 2.2) generates CLs from CCs. The algorithm for generating CC data is relatively simple; CC curves are confined to the isoparametric curves on each surface patch. The increment from one position to another along an isoparametric path is defined either with a constant or varied value. The varied increment is generally calculated with a CPU intensive “trial and error” approach. Other limitations of the method are mentioned in Chapter 2. In the following sections we present a new approach to five-axis tool path generation system which uses a non-isoparametric indexing method.

### 5.2.1 Edge List Generation

In the system developed for this work, CC points are derived from a set of edge lists. Each edge list is obtained from the intersection of a cutting plane with the STS approximating the object surface. To simplify the implementation, cutting planes are confined to be parallel to the xz plane, although this is not an intrinsic limitation of the method. The increments between adjacent CC points are initially defined by the points where the cutting plane intersects the edges of the triangle (e.g., point a, b and c in Fig.

5.3). The length of the intersection segment between the edges determines the increment between nominal CC points. The fact that the triangles are guaranteed to be close to the actual surface means that the length of the step-forward increment is accurately determined. This provides a simple solution to one of the difficult problems of NC tool path generation, namely, the correct determination of the step-forward distance. However, this must be used with care since gouging can easily occur at points between the CC points if there are nearby concave surfaces. Intermediate CC points can be easily added if the nominal CC points are deemed to be too far apart.



**Figure 5.3** Edge list generation

There are several approaches to deriving the edge list set. One is to use a segment sorting technique similar to the scanline algorithm in polygon hidden surface removal [70]. A large complex surface may require thousands of triangles for a small approximation error. Consequently, the scanline algorithm is computationally expensive.

To provide fast algorithms (potentially), an alternative approach based on edge linkage is developed. Assuming that the tool moves in the  $+x$  axis (Fig. 5.2), and  $ab$  is the intersection segment of the left most triangle with the cutting plane. Given position  $q$  which is a small increment from  $b$  along the tool moving direction, the triangle intersecting the line through  $q$  and parallel to the  $z$  axis is selected and the  $bc$  segment is derived, and so on. This procedure finds the triangle closest to the previous one along the tool moving direction. The triangle either shares an edge with the previous one or is of higher priority in the case of surface overlapping. If the triangle set provides information on triangle edge sharing, the edge list can be directly derived by tracking the triangle sharing an edge with

the previous one. However, the edge lists obtained with the shared edges must be checked for overlapping segments.

For overlapping surface patches or triangles (Fig. 5.4), a subdivision algorithm is used to generate unique CC curves. Note that the distance between the two surface shown in Fig. 5.4 is greatly exaggerated. Actual gaps may be as small as 0.001 inches. This algorithm is based on the priority assigned to each overlapping triangle. The segments with higher priority are chosen. In the case where part of a segment is right below other segments, the end point of the lower priority segment must be found. The position at which the cutter moves to the next triangle of higher priority is the switch point. For instance, in Fig. 5.4, segment  $bc$  is partially under segments  $de$  and  $ef$ , and the effective segment of  $bc$  is  $bs$ . The switch point  $s$  is derived with the subdivision algorithm 5.1.

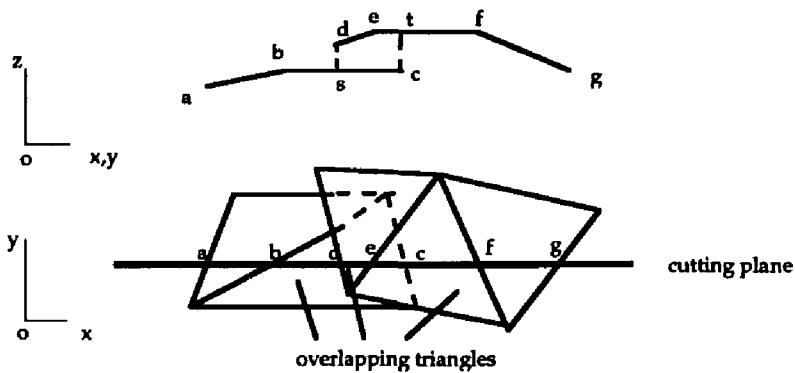


Figure 5.4 Overlapping segment removal

**Algorithm 5.1** CC Edge List Generation and Overlapping Segment Removal

parameters:

$\delta inc$	increment from the end of previous intersection segment
cur_tri	triangle to be touched
tp	touch point
cand_tp	candidate touch point
init_tp	initial touch point
next_cand_tp	next candidate touch point
$\delta dist$	distance between two points

initial parameters: (refer to Fig. 5.4)

y	current cutting plane
$\delta inc$	increment from b

procedure:

- (1) given  $\delta_{inc}$
- (2) find *cur\_tri*
- (3) find intersection of *cur\_tri* with the cutting plane  
and choose the end point as *cand\_tp* (point *c* in Fig. 5.4)
- (4) if *cand\_tp* has higher priority than the point on the overlapping surface with the  
same *x, y* values (point *t* on Fig. 5.4)
  - tp* = *cand\_tp*
  - else (overlapping)
    - tp* = *switchpoint*

*switchpoint*

- while the switch point is not found
  - find the *cand\_tp* at the mid of *init\_tp* and *next\_cand\_tp*
  - if the *cand\_tp* has the same priority as *init\_tp*
    - init\_tp* = *cand\_tp*
  - else
    - next\_cand\_tp* = *cand\_tp*
  - if the distance between *init\_tp* and *next\_cand\_tp* <  $\delta_{dist}$ 
    - the switch point is found
    - if *init\_tp* has the higher priority
      - return *init\_tp*
    - else return *next\_cand\_tp*

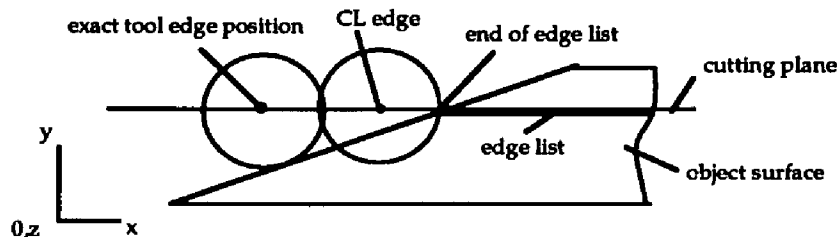
end of Algorithm 5.1

For the efficiency of intersection calculation, triangles are localized with a "bucketing strategy" which is similar to the bucketing scheme used in TRI\_XYINDEX discussed in Chapter 3. For each cutting plane, only the triangles in the buckets intersecting the cutting plane are taken into account.

### 5.2.2 Edge Handling

One of the somewhat subtle difficulties is ensuring gouge-free tool paths as the cutter moves across or along surface edges. An edge can be an unshared edge along the outer perimeter of an object or a shared edge at the junction of multiple patches in the interior of the object. At first glance boundary identification might appear to be easily handled, but the

presence of interior holes and of topologically inconsistent features such as overlapping surfaces and gaps, significantly complicate the issue. A method for boundary handling is proposed in the following paragraphs.



**Figure 5.5** CL edge and exact tool edge position

In triangulation, the surface boundary is approximated by a set of segments which are also the edges of triangles. Intersections of the cutting planes with these segments are in the CC data. To find the surface boundary is equivalent to finding ends of CC edge lists. CLs corresponding to the ends of edge lists are referred to as CL edges. CL edges are the tool positions from which the front edge of the tool begins to contact or leave the surface. Note that CL edges may not be the exact tool edge positions where a tool begins to touch or leave the object surface. Fig. 5.5 shows the difference between an exact tool edge position and a CL edge derived from an end of edge lists. There is the potential for the tool to leave material if the CL edge position is used instead of the exact tool edge position (as shown in Fig. 5.5). However, we use the CL edge position as our starting point since the unremoved material is likely to be removed by the adjacent tool paths. The CCs corresponding to the CL edges are marked with an edge flag. These flags can be used for rapid positioning of the tool when moving from the end of one tool path to the start of the next.

To localize the searching for CC edge points, buckets intersecting the cutting plane are selected. The number of the buckets to be checked depends on the searching conditions such as the searching direction and the edge type. For instance (Fig. 5.6), in searching a leading edge (current position cp is not on the surface) and the tool moving towards +x axis, the candidate buckets are those that intersect the cutting plane starting from cp to the



end of the cutting domain (ep). Furthermore, the leading edge position must be the left most end point of the intersection segments of the cutting plane with the triangles. Triangles are sorted into the buckets based upon their x and y domains; to further decrease the number of buckets, only the triangles in the first bucket column are checked. The first bucket column is the one with triangles intersecting the cutting plane and is the first column passed by the tool. The procedure for finding an edge is listed in Algorithm 5.2.

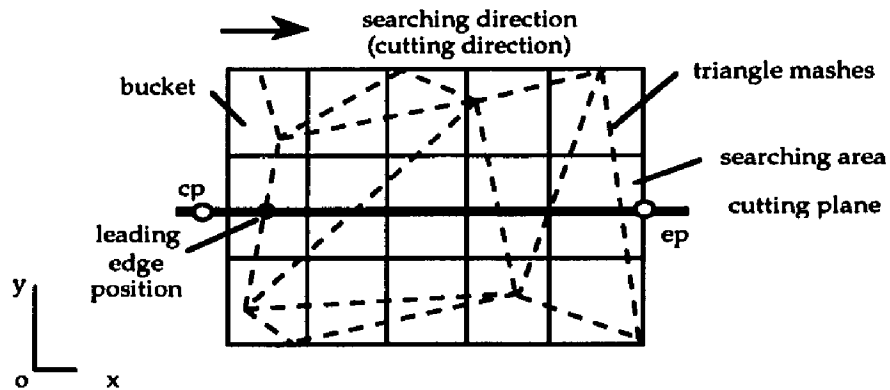


Figure 5.6 Finding a leading edge

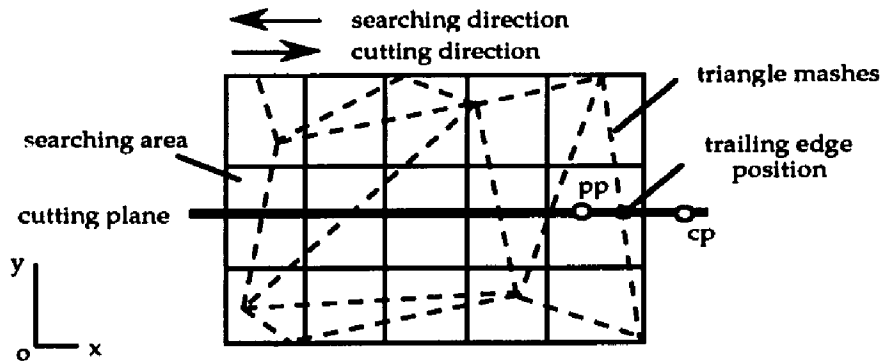


Figure 5.7 Finding a trailing edge

**Algorithm 5.2. Finding a Leading/Trailing Edge** (referring to Fig. 5.6 and 5.7)

Symbols:

find\_le    find leading edge  
 find\_te    find trailing edge  
 D+        positive cutting direction (+X direction)  
 D\_        negative cutting direction (-X direction)

initial parameters:

curr\_out\_pt      current out-of-surface position  
 prev\_on\_pt      previous position which is on the surface (if find\_te)  
 start\_pt        start position  
 end\_pt         end position

(1) setup up search domain:

start\_pt = curr\_out\_pt;  
 if find\_le and D+  
     end\_pt = end of cutting domain  
 else if find\_le and D\_  
     end\_pt = start of cutting domain  
 else (find\_te)  
     end\_pt = prev\_on\_pt

(2) setup bucket ranges:

start-bucket-column = buckets column at curr\_out\_pt.  
 if find\_le  
     end-bucket-column = bucket column at end of cutting domain  
 else end-bucket-column = bucket column at point prev\_on\_pt

(3) searching edge:

for each bucket column in the range of start-bucket-column and end-bucket-column  
 if the column with triangles intersecting the cutting plane within start\_pt and  
 end\_pt  
     callSearch

*Search*

for each triangles with higher priority in the bucket column  
 if find\_le and D+    or    find\_te and D\_  
     find the left most point from left ends of the intersection segments  
 else (find\_le and D\_    or    find\_te and D+)  
     find the right most point from right ends of the intersection segments  
 note that the edge point must be between point curr\_out\_pt and end\_pt

end of Algorithm 5.2

### 5.2.3 CC Point Interpolation

Theoretically, increments between CC points can be very large. Provided that the front edge of the cutter moves along the segment, the deviation between the machined surface and the math surface does not exceed the triangulation error. However, gouging caused by the back or side edges of the tool is not limited. Current tool path correction is based on testing individual tool positions. To increase the probability of gouge free tool movements, some CC points have to be inserted between ends of edges (Fig. 5.8).

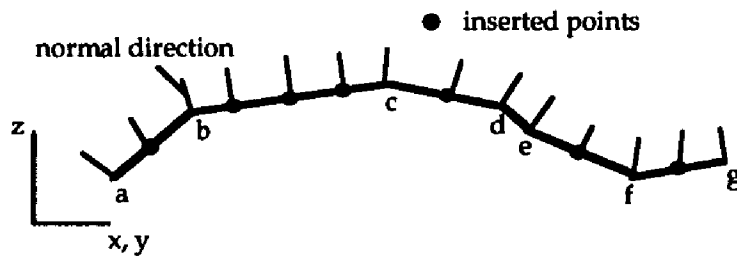


Figure 5.8 CC point interpolation

The interpolation is achieved in two steps: interpolation based on the CC point spacing followed by interpolation based on the change of normals at adjacent CC points. In the first step assume that the tool axis is not changed; if the distance between adjacent CC points is larger than the default increment ( $\delta$ ), some intermediate CC points are interpolated. The default increment ( $\delta$ ) is calculated with equation (5.1), where  $C$  is the maximum allowable gouge (Fig. 5.9). Equation (5.1) is easily derived by applying the Pythagoras's Theorem to the triangle  $abc$  in Fig. 5.9. In the second step if the normals at adjacent CC points (including the interpolated ones) are not constant, some new CC points are interpolated. This is to avoid the possible gouge caused by the top of the tool. Even though the increment between the CC points is not larger than  $\delta$ , the distance between the top centers of the tool at adjacent CC points may exceed the limit, which means that the surface may be gouged by the top edges of the tool. The mathematical equations for calculating the length of the swept curve formed by the top center of the tool can be derived as follows.

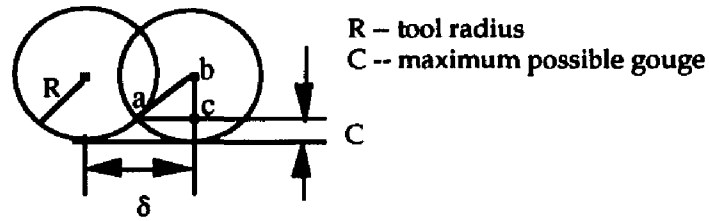


Figure 5.9 Default increment

$$\delta = 2.0 * \sqrt{2 \cdot R \cdot C - C^2} \quad (5.1)$$

In Fig. 5.10, T is the top center location of the tool with length L and axial direction  $\bar{n}$ . In equation (5.2a), T is given as a function of the parameter s which varies from 0 to 1 as the tool center moves from  $P_1$  to  $P_2$ . The length (S) of the curve formed by the top center of the tool (T) is given by equation (5.2b).

$P_1, P_2$  vertices of an edge segment  
 $\bar{n}_1, \bar{n}_2$  normals at  $P_1$  and  $P_2$   
 $\bar{n}$  normal at T(s)

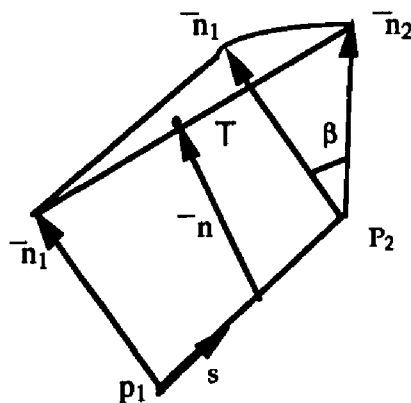


Figure 5.10 Curve formed by the top center of a tool

$$\bar{T}(s) = \bar{P}_1 + s \cdot \overline{P_2 P_1} + L \cdot \bar{n}, \quad s \in [0, 1] \quad (5.2a)$$

$$S = \int \left| \overline{P_2 P_1} + L \cdot \frac{d\bar{n}}{ds} \right| \cdot ds, \quad s \in [0, 1] \quad (5.2b)$$

where  $\bar{n}$  is calculated from equations (5.2c) - (5.2f).

Assume that the angle between  $\bar{n}_1$  and  $\bar{n}_2$  is  $\beta$ , and the angle between  $\bar{n}_1$  and  $\bar{n}$  is  $s\beta$ . Then  $\bar{n}$  at  $T(s)$  should satisfy the equations (5.2c) - (5.2f).

$$\bar{n}_1 \times \bar{n} = \bar{n}_1 \times \bar{n}_2 \quad (\text{for choosing the correct direction of } \bar{n}) \quad (5.2c)$$

$$|\bar{n}| = 1 \quad (\text{normalized vector}) \quad (5.2d)$$

$$\bar{n} \cdot (\bar{n}_1 \times \bar{n}_2) = 0 \quad (\text{on the plane formed by } \bar{n}_1 \text{ and } \bar{n}_2) \quad (5.2e)$$

$$\bar{n}_1 \cdot \bar{n} = \cos(s\beta) \quad (\text{the angle between them is known}) \quad (5.2f)$$

$\bar{n}$  is a non-linear equation of  $s$  and it may take more CPU time to calculate the exact length of the curve. Note that the distance between  $P_1$  and  $P_2$  is not larger than the default increment ( $\delta$ ) which is a very small value. To simplify the calculation of the curve length ( $S$ ), an approximation to  $S$  is given by eqn. (5.2g) under the assumption that the angle between  $n_1$  and  $n_2$  is very small (e.g.,  $5^\circ$ ). If  $S$  is larger than the default stepover, new CC points are interpolated. The number ( $n$ ) of the CC points to be interpolated is defined by equation (5.2h).

$$S = |\overline{P_2 P_1}| + L \cdot \beta \quad (5.2g)$$

$$n = S/\delta \quad (5.2h)$$

#### 5.2.4 Effective Radius of Tilted Flat-End Tool

##### List of Symbols

$r$	flat-end cutter radius
$r_e$	effective ball-end cutter radius
$k$	ratio of $r_e$ to $r$

$o$	bottom center of a flat-end cutter
$o_i$	center of effective cutter $i$
$\beta$	tilting angle of a flat-end cutter
$c$	cuspl height
$p_1, p_2$	adjacent CC points on a $x = \text{constant}$ plane
$\bar{n}_1, \bar{n}_2$	normals at $p_1, p_2$ projected onto the comparison plane
$\bar{n}_c$	average normal of $\bar{n}_1$ and $\bar{n}_2$
$\theta$	angle between $\bar{n}_1$ and $\bar{n}_2$
$y_{\text{default}}$	default stepsize in $y$ direction
$y_{\text{slope}}$	adjusted stepsize when a slope exists

The profile of a tool bottom varies with the moving direction. The main concern here is the silhouette of the tool bottom on the comparison plane which is perpendicular to the moving direction and passes through the CC point. Assuming that the tool with tilting angle  $\beta$  is moving toward the  $+x$  axis (Fig. 5.11a), the plane parallel to the  $yz$  plane and through the CC point is the comparison plane at the current CC point. The silhouette is the orthogonal projection of the cutter onto the comparison plane. If a circle is used to approximate the semi ellipse under the tool center  $O$  on the comparison plane (Fig. 5.11b), the radius of this circle is referred to as the effective ball-end tool radius  $r_e$ . Note that this circle must pass through point CC and the two end points (a, b) of the long axis.

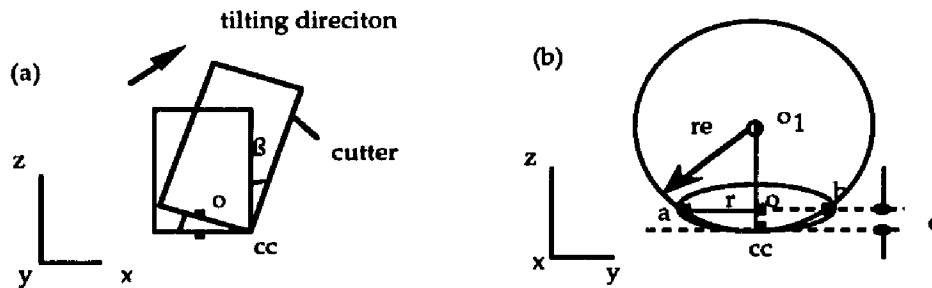


Figure 5.11 Effective tool radius

$$r_e^2 = r^2 + (r_e - c)^2 \quad (5.3)$$

$$\text{where } c = r \sin\beta$$

$$r_e = \frac{(1 + \sin^2\beta)}{2\sin\beta} r = k \cdot r \quad (5.4)$$

$$k = \frac{(1 + \sin^2\beta)}{2\sin\beta} \quad (5.5)$$

From eqn. (5.5),  $k$  varies dramatically with a small angle change at tilting angles of less than  $15^\circ$ . Initially, the tool axis is oriented with  $3^\circ - 5^\circ$  tilting angle from the normal of a CC point towards the moving direction. When a tool moves forward with the tilting angle, the profile of the tool bottom is similar to a ball-end tool. Cusp heights caused by the tilting of a flat-end cutter must be taken into account in the step-over determination. CC data generation is carried out before tool positioning, at which time the tilting angle at individual CC points is unknown. Therefore, a pre-defined tilting angle is assumed at all CC points. Based upon experimentation, the pre-defined tilting angle  $\beta$  is assigned to be  $15^\circ$  for the purpose of a conservative stepover distance calculation. The actual tilt angle is  $3^\circ$  whenever possible. From equation (5.5),  $k$  for  $15^\circ$  is equal to 2.060, meaning that the effective radius is about two times the actual radius.

### 5.2.5 Edge List Stepper

The silhouette of a flat-end tool with a tilting angle is similar to a ball-end tool. As a ball-end cutter moves along tool paths, excess material may be left between adjacent tool paths. The amount of excess material depends on the stepover distance between tool paths. The stepover increment can be derived from the comparison of cusp heights between pairs of sampled tool positions. In five axis machining, a cutter can be oriented at any direction. Given two tool positions at adjacent tool paths, the cusp height in the normal direction to the surface point is of primary concern. The sampled pair of points and the associated normals may not be on the same plane. To approximate the cusp height, a cusp plane is utilized which passes through the sampled pair of points and the average normal of those at the sampled points. For instance (Fig. 5.12), given two surface points  $P_1, P_2$  on adjacent

paths and normals  $\bar{N}_1, \bar{N}_2$  at  $P_1, P_2$ , the cusp plane is formed by vector  $\overline{P_1P_2}$  and the average normal  $\bar{n}_c$  of  $\bar{N}_1$  and  $\bar{N}_2$ . Referring this cusp plane to as the comparison plane at  $P_1$  and  $P_2$  discussed in section 5.2.4, the centers of the effective tool radii corresponding to CC points are derived with eqn. (5.8) (Fig. 5.13), where  $r_i$  is the effective tool radius at  $P_i$  ( $i = 1, 2$ ).

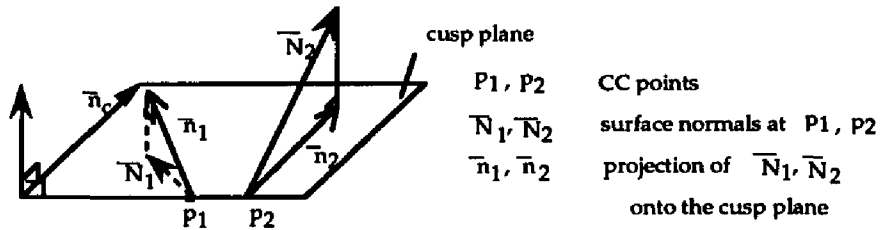


Figure 5.12 Cusp plane

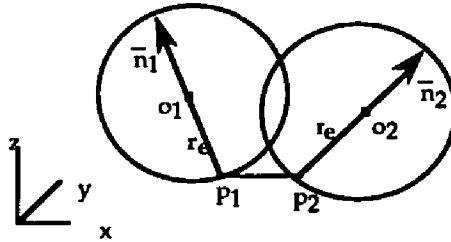


Figure 5.13 Effective tool center location

$$\bar{n}_c = \overline{P_1P_2} \times (\bar{N}_1 + \bar{N}_2) \quad (5.6)$$

$$\bar{n}_i = \bar{N}_i - (\bar{N}_i \cdot \bar{n}_c) \cdot \bar{n}_c \quad (i = 1, 2) \quad (5.7)$$

$$\bar{o}_i = \bar{p}_i + (r_e) \cdot \bar{n}_i \quad (i = 1, 2) \quad (5.8)$$

Assume that the cutter axis is parallel to the surface normal when it touches a CC point; the angles between the cusp plane and normals at the sample points define effective cutter radii on the cusp plane. The angle between the projections of the CC point normals



onto the cusp plane affects the cusp height. As described in the above section, a pre-defined tilting angle ( $15^\circ$ ) is applied to the tool at any CC points. Therefore, the effective cutter radii measured in the cusp plane at the sampled CC points are the same.

If the normals at sampled points are parallel, the stepsize can be derived from eqn. (5.8) for a given allowable cusp height and tool size (Fig. 5.14). An object surface may not be flat, and this stepsize can be further adjusted with eqn. (5.11), where  $y_{\text{slope}}$  is the adjusted stepsize.

$$y_{\text{default}} = 2\sqrt{r_e^2 - (r_e - c)^2} = 2\sqrt{2r_e c - c^2} \quad (5.9)$$

If  $c \ll 1$ ,

$$y_{\text{default}} \approx 2\sqrt{2r_e c} \quad (5.10)$$

If  $|O_1 O_2| = y_{\text{default}}$ ,

$$y_{\text{slope}} = y_{\text{default}} / \sqrt{y_{\text{default}}^2 + \Delta z^2} \quad (5.11)$$

If  $r_e = 2.061r$  ( $\beta = 15^\circ$  for tilting a flat-end cutter),  
from eqn. (5.10),

$$y_{\text{default}} \approx 2\sqrt{2 \cdot 2.061rc} \approx 4.0596\sqrt{rc} \quad (5.12)$$

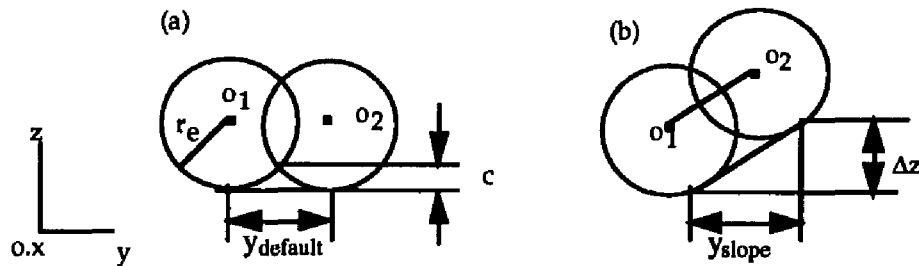


Figure 5.14 Stepover and cusp height

### 5.3 CL DATA GENERATION

For a flat-end tool, cutter locations (CLs) are the center locations of the tool bottom. Given a CC point and the associated surface normal, set the tool with its axis parallel to the normal and its front edge touching the CC points; the locus of the possible CL points forms a circle. The circle is on the plane through the CC point and perpendicular to the normal, with the center at the CC point and radius equal to that of the tool. To define a unique CL associated with a CC point, a tool center plane formed by the CC point normal and the tool moving direction is used. Tool moving direction  $\overline{mv}$  is defined as the vector from one CC point to the next (Fig. 5.15),  $\overline{ax}$  is the tool axis orientation,  $\overline{n}$  is the surface normal at  $P_i$ . Tool center  $C$  is one of the intersection points of the tool center plane with the circle and behind CC point  $P_i$  along  $\overline{mv}$ . Assuming that  $\overline{M}$  is the normal to the plane formed by  $\overline{n}$  and  $\overline{mv}$ , tool center  $C$  is derived from eqn. (5.13) - (5.15).

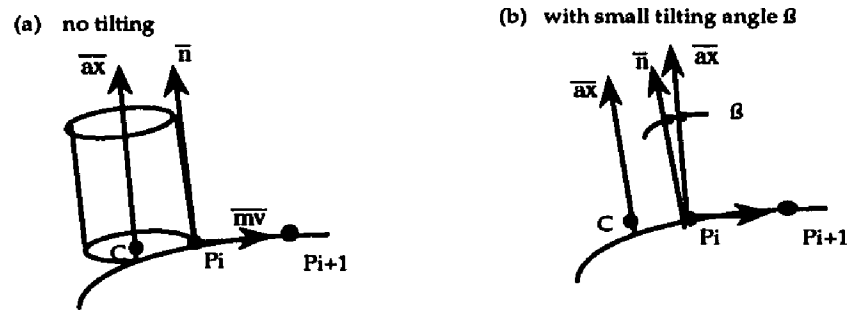


Figure 5.15 Offsetting cutter center location

$$\overline{CP_i} \cdot \overline{M} = 0 \quad (5.13)$$

$$|\overline{CP_i}| = R \quad (5.14)$$

$$\overline{CP_i} \cdot \overline{n} = 0 \quad (5.15)$$

where

$$\overline{M} = \overline{n} \times \overline{mv}$$

A small trailing angle (for example, 3 degrees) is applied to an initial tool orientation in the tool center plane. In Fig. 5.15b,  $\bar{ax}$  is the tool axis with tilting angle  $\beta$ . Tool axis  $\bar{ax} = (\sin\beta, 0, \cos\beta)$  and tool center  $C = (-r\cos\beta, 0, r\sin\beta)$ . Note that the normals to CC points change along CC curves. Though the projection of CC points onto the xy plane is a straight line, the projection of CLs may not be (Fig. 5.16).

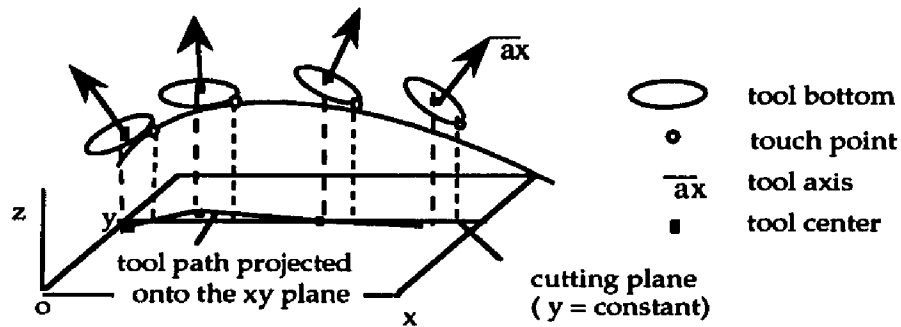


Figure 5.16 CC points vs. tool paths

### 5.3.1 Interference Checking

Interference checking is necessary after a CL is derived from a CC point. A CL is acceptable if there is no interference between the tool and the surface. It is possible that by contacting the CC point with the tool front edge, the back or side of the tool gouges into the surface (Fig. 5.17). When a CL with interference is found, it is modified with the angle adjustment described in the next section. The mathematical concept used in interference checking is discussed below.

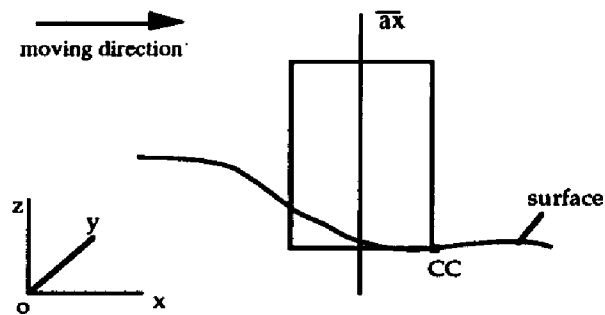


Figure 5.17 Interference between a tool and a surface

Generally, each tool movement affects a small portion of the surface. Interference checking is localized with the bucketing strategy discussed in section 3.1. For each tool position, only the triangles with the projection overlapping the tool shadow are checked. The cutter shadow is the cutter projection onto the  $xy$  plane. To simplify the calculation, the smallest rectangle containing the tool projection is used (Fig. 5.18).

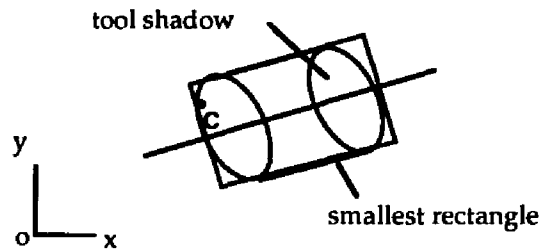


Figure 5.18 Checking Localization

To simplify the interference checking, each triangle overlapping the cutter shadow is transformed to a space in which the tool axis is parallel to the  $+z$  axis with the tool center at  $(-r, 0, 0)$  and the front edge at the origin. The transformation consists of two steps: first translating the CC point to the origin, then rotating the coordinate system so the  $z$  axis is parallel to the tool axis and with the tool center at  $(-r, 0, 0)$  where  $r$  is the tool radius. The homogeneous transformation matrix  $T$  is given in eqn. (5.16). The angle between the  $x$  axis and the projection of  $\bar{ax}$  onto the  $xy$  plane is  $\theta$  (Fig. 5.19).  $\beta$  is the angle between the  $z$  axis and  $\bar{ax}$ .  $\phi$  is the angle between  $pc$  and the plane formed by  $\bar{ax}$  and the  $z$  axis when  $\bar{ax}$  is not parallel to the  $z$  axis or  $\phi$  is the angle between  $cp$  and the  $-x$  axis if the tool axis is parallel to the  $z$  axis. The procedure for interference checking is listed in Algorithm 5.4. The mathematical concepts in identifying the relationship between a triangle and a flat-end tool is discussed next.

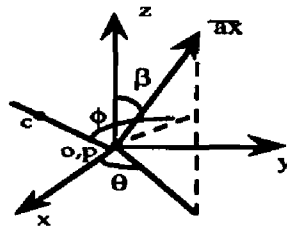


Figure 5.19 Transformation on the coordinate system

$$T = \begin{bmatrix} \sin\phi\cos\beta\sin\theta - \cos\phi\cos\theta & \sin\phi\cos\beta\cos\theta + \cos\phi\sin\theta & -\sin\phi\sin\beta & 0 \\ -\cos\phi\cos\beta\sin\theta - \sin\phi\cos\theta & -\cos\phi\cos\beta\cos\theta + \sin\phi\sin\theta & \cos\phi\sin\beta & 0 \\ \sin\beta\sin\theta & \sin\beta\cos\theta & \cos\beta & 0 \\ -tp_x & -tp_y & -tp_z & 1 \end{bmatrix} \quad (5.16)$$

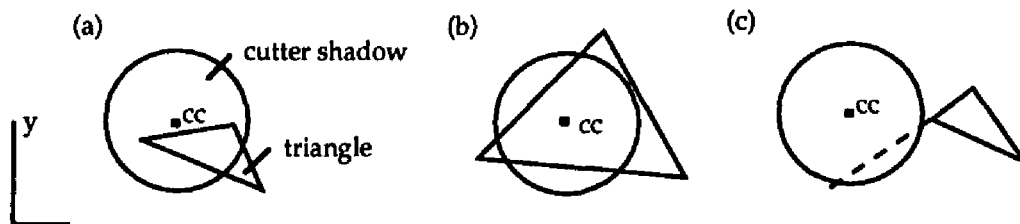
**Algorithm 5.3 Interference Checking**

```

for each CL in CL lists
  find the tool shadow
  form transformation matrix (TM)
  for each triangle overlapping the tool shadow
    transform vertices with TM
    check interference
  if interference
    exit checking to correction process
end of Algorithm 5.3

```

After the transformation, the tool axis is parallel to the z axis with the center at  $(-r, 0, 0)$ . The projection of the tool onto the xy plane is a disk. Triangles whose xy projections overlap the disk are checked for interference. Figure 5.20 shows some cases of the projections of the tool and triangles (after transformation). Assume that the tool is at the initial tool position, and is then either raised or lowered until it just touches the triangle. If the tool has to be raised, gouging is detected and the initial tool position needs modification.



**Figure 5.20** Triangle location vs. the cutter shadow

The technique for finding a touch point on a non-vertical and non-horizontal triangle plane is discussed below. Referring to Fig. 5.21, Q is a touch point on the plane, A, B and C are 3 vertices of a triangle, and  $c$  is the tool center location. Note that when the tool touches a triangle, the touch point can be on a vertex, on an edge or inside the triangle. Therefore, Q is tested for its location relative to the triangle. If it is not in the triangle, the possible edges to be touched first are selected with the permutation table (refer to section 3.4). Appendix C discusses the calculation of Q. For a triangle parallel to the tool bottom, the z value is the same as the z component of any vertex. If the triangle is vertical (i.e., parallel to the tool axis), touch point Q must be on an edge or at a vertex. The procedure for selecting the candidate edges of a vertical triangle is described in Algorithm 5.4.

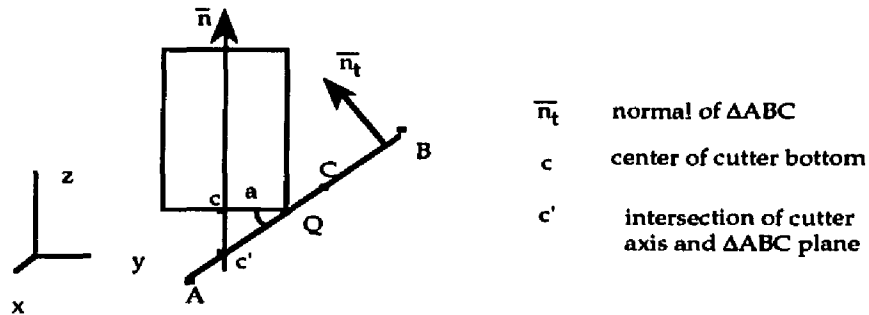


Figure 5.21 Interference Checking

**Algorithm 5.4** Edge Selection and Touch Point Setting for a Vertical Triangle

if a vertical edge exists

    the edge sharing the high vertex in z direction with the vertical edge is selected

else order vertices in the z direction

    if the edge opposite the median vertex is above the median vertex

        this edge is selected

    else the edges sharing the median vertex are selected

find touch point on each selected edge

choose upper touch point if more than one is available

end of Algorithm 5.4.

### 5.3.2 TOOL POSITION CORRECTION

Tool positions causing gouging are corrected in such a way that the tool front edge touches or gets as close as possible to the nominal CC points on the surface. To keep the tool front edge position unchanged, an angle adjustment method is proposed. Tool positions to be corrected are modified by rotating the tool around the axis through the touch point and the normal to the cutter center plane so that the tool touches the surface instead of gouging into it. In Fig. 5.22a, the tool collides with a triangle. By tilting the tool about the contact point (CC) and towards the moving direction ( $\overline{mv}$ ), the tool touches the triangle either at a vertex, an edge or in the triangle (Fig. 5.22b). The maximum allowable rotation angle is the smaller one of the pre-defined maximum tilting angle (such as  $45^\circ$ ) and the angle between the initial  $\overline{ax}$  and  $\overline{mv}$  (Fig. 5.22c). The tilting angle is the minimum angle of rotation that avoids interference. Since a tool may gouge more than one triangle, the required tilting angle is the maximum of all the tilting angles of the triangles interfering with the tool.

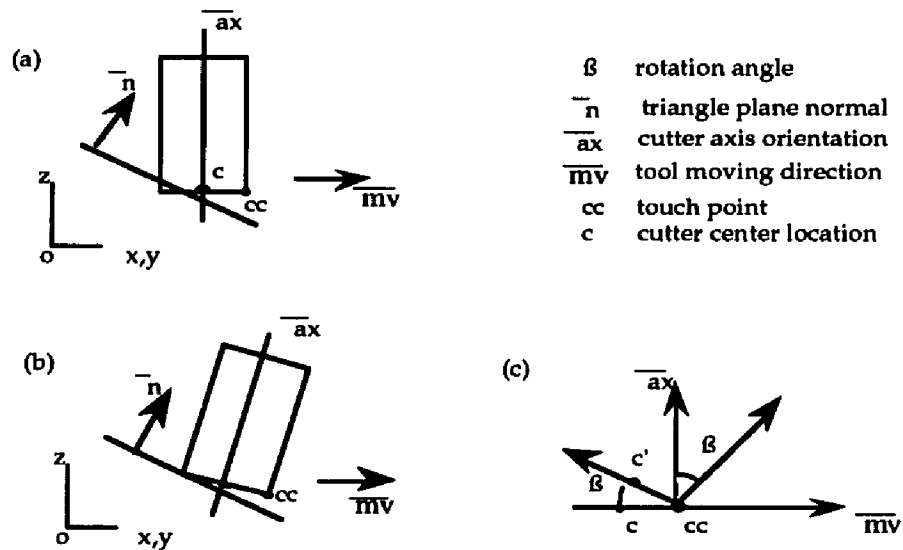


Figure 5.22 Tilting angle and tool position

Triangles to be tested are also selected with the localization scheme discussed in section 5.3.1. Note that the search domain changes with the tool axis. If the tool is tilted, it must be retested for gouging. If a gouge occurs again, the tool is lifted a minimum amount along the normal direction. The minimum lifting amount is calculated using the resetting method used in the interference checking process discussed in section 5.3.1. The tool is lifted so that it just touches the surface at some point other than the original CC point. Algorithm 5.5 lists the procedure for correcting tool positions.

**Algorithm 5.5** Tool Position Modification

```

initial tilting angle  $\beta$  = zero
new tilting angle  $\beta_{\text{new}}$  = zero
 $\beta_{\text{max}} = \min(\text{angle}(\overline{ax}, \overline{mv}), \text{pre-defined maximum allowable tilting angle})$ 
ChangeTiltAngle
if failure
    LiftTool

ChangeTiltAngle
    for each triangle interfering with the tool tilted by  $\beta$ 
        find  $\beta_{\text{new}}$ 
        if  $\beta \leq \beta_{\text{new}} \leq \beta_{\text{max}}$ 
             $\beta = \beta_{\text{new}}$ 
        else return failure
    LiftTool
    initial lifting amount = zero
    for each triangle interfering with the tool with lifting amount
        find the new minimum lifting amount
        lifting amount = max (lifting amount, new minimum value)
end of Algorithm 5.5

```

### 5.3.3 ANGLE ADJUSTMENT

The rotation required to ensure gouge avoidance can be calculated by finding the tilting angle at which the tool just barely touches the triangle. The touch point can occur at a vertex, an edge, or on the interior of the triangle. The probability of the tool touching the interior of a triangle is much less than for a vertex or an edge. Some cases require



numerical solutions to find the tilting angle. To speed up the computation, the tilting angle for a tool to touch the vertices is checked first, then the edges, and finally the interior of a triangle. The procedure for finding a tilting angle for a triangle is listed in Algorithm 5.6. Note that in the vertex or edge case, there are two possibilities: the tool bottom touches or the side touches; while only the tool bottom can touch in the inner triangle case.

**Algorithm 5.6** Finding a Tilting Angle When a Triangle Interferes with the Tool

```

initial  tilting angle  $\beta$  = current updated tilting angle
         new tilting angle  $\beta_{\text{new}}$  = current updated tilting angle
          $\beta_{\text{max}} = \min(\text{angle}(\overline{ax}, \overline{mv}), \text{pre-defined maximum allowable tilting angle})$ 
given a triangle, find the number of vertices above the bottom plane of the tool
if one vertex above tool bottom
    ToolTouchOneVertex to find a new tilting angle
else if two vertices above tool bottom
    ToolTouchTwoVertices
else ToolTouchThreeVertices

ToolTouchOneVertex
    if the vertex is inside the tool in its current tipped position
        find  $\beta_{\text{new}}$  for the vertex above tool bottom
        if  $\beta \leq \beta_{\text{new}} \leq \beta_{\text{max}}$ 
             $\beta = \beta_{\text{new}}$ 
        else set lift up flag and exit
    if edges sharing above vertex interfere with the tool
        find  $\beta_{\text{new}}$  for the tool just to touch the edges
        if  $\beta \leq \beta_{\text{new}} \leq \beta_{\text{max}}$ 
             $\beta = \beta_{\text{new}}$ 
        else set lift up flag and exit
    if the triangle interferes with the tool
        find  $\beta_{\text{new}}$  for the tool to touch the triangle
        if  $\beta \leq \beta_{\text{new}} \leq \beta_{\text{max}}$ 
             $\beta = \beta_{\text{new}}$ 
        else
            set lift up flag and exit
end of Algorithm 5.6

```

Note that the structures of *ToolTouchTwoVertices* and *ToolTouchThreeVertices* in Algorithm 5.6 are similar to *ToolTouchOneVertex* except that one or two more vertices and edges are checked for interference with the tool with tilting angle  $\beta$ .

Given a vertex (A), first it is checked for interference against the tool (i.e., whether the vertex is inside the tool). A tilting angle is calculated if interference occurs, so that the tool touches the vertex instead of enclosing it. A vertex can be touched by either the bottom or the side of a tool. In Fig. 5.23, if vertex A is within the torus formed by rotating the tool bottom disk about the y axis (i.e.,  $r_{\min} < \text{the distance from A to the y axis} < r_{\max}$ ), it is touched by the tool bottom; otherwise, it is touched by the side. The maximum tilting angle is the angle between the original tool axis and the tool moving direction. It is possible that a vertex can not be touched by either the tool bottom or side when rotating the tool within the tilting angle limit. In this case, the tool must be lifted. The procedure is listed in Algorithm 5.7. The derivation of the tilting angle for a vertex case is described in Appendix D.

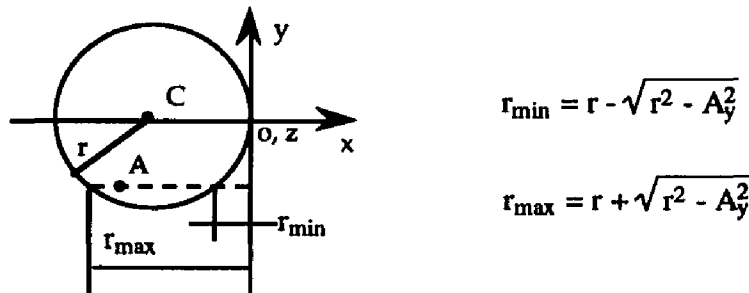


Figure 5.23 Torus formed by rotating a disk

**Algorithm 5.7** Finding a Tilting Angle for the Tool to Touch a Vertex

given: vertex A, tool center C, tool axis AX and tool radius R

if A inside the tool

if A within the torus

the tool bottom touches the vertex

else the tool side touches the vertex

find tilting angle

if tilting angle out of the angle limit

set lifting flag

end of Algorithm 5.7

Similarly, in the edge case, an edge is first checked for interference with the tool tilted to the most recently updated value of  $\beta$ . To simplify the checking process, the edge is transformed by the negative tilting angle, so that the tool axis is still parallel to the z axis with center and front edge at  $(-r, 0, 0)$  and  $(0, 0, 0)$  respectively. The procedure for interference checking is listed in Algorithm 5.8. A new tilting angle is derived only when necessary. Equations for finding a tilting angle for the edge case are given in Appendix E.

**Algorithm 5.8** Interference Checking between a Tool and an Edge

```

given: 2 vertices, tool radius, tool center location, tilting angle
rotate the edge by the negative tilting angle
if the two vertices of the edge under the tool bottom plane
    no gouge
else if at least one vertex inside the tool
    gouge
else if edge  $\parallel$  xy plane
    if distance from tool center to the projection of edge onto the xy plane  $< r$ 
        gouge
    else no gouge
else find intersection point (T) of bottom plane with the edge
    if edge segment above T overlapping the tool (projected onto xy plane)
        gouge
    else no gouge
end of Algorithm 5.8

```

The procedure for finding a tilting angle for the case of touching the triangle interior is similar to the vertex and edge cases. However, only one touching possibility exists: the triangle is contacted by the tool bottom only. Also note that in all the cases described above, generally there is more than one tilting angle satisfying the equations. The tilting angle is the minimum angle within the angle limit for which the tool no longer interferes with the triangle. The calculation of the tilting angle for the interior triangle case is given in Appendix F.

Note that the angle adjustment is applied to the cutter center plane at a CC point and the tool is rotated about the CC point. It is possible that the location of a gouge may be close to the CC point. Simply applying the angle adjustment to avoid interference requires a large tilting angle. Tool path smoothness greatly depends on the behavior of the tilting angles as the tool moves across the surface. To generate smooth tool paths (i.e., gradually changing tool axis angles), a tool orientation is defined based on the difference of the tilting angles at the previous tool position and the current position. If the difference is too large (e.g., more than 5°), a new candidate tool position is calculated with the algorithms of section 5.3.1. Then by comparing the two positions, the one with smaller displacement to the initial tool position is selected.

#### **5.4 OPERATION PROCEDURES, TEST CASES AND TEST RESULTS**

Most parameters in five axis tool path generation are similar to those in section 3.6, including the Maximum Allowable Cusp Height, Cutting Style, Cutting Direction, Cutting Tool Size, and Cutting Direction. The parameters to be specially defined in five axis tool path generation include:

1. **Cutting Tool Shape** -- It is currently confined to a flat-end tool.
2. **Maximum Tolerance Deviation** --The overall tolerance deviation is the sum of the surface triangulation tolerance and the error bound for tool path generation. Currently the error for surface triangulation is defined to be 75% of the Maximum Tolerance Deviation and 25% of the Maximum Tolerance Deviation for tool path generation error.

The algorithms discussed in this chapter have been implemented on a Silicon Graphics Indigo R4000 Workstation and evaluated with respect to the following criteria: accuracy, speed, robustness and storage requirement. The following surfaces are used to evaluate the software.

1. **zip5** is composed of two convex patches sharing an edge. Gouge may occur along the edge (refer to Color Plates 9 and 10).
2. **z6324r** is a part of a panel between the side and rear window of a car. It consists

of thirty-seven patches with some overlapping surfaces. C1 discontinuities exist at some shared edges of patches. Gouges are likely to occur at the concave area between the side and rear window (refer to Color Plates 11 and 12).

3. **saddle** is composed of nine rectangular bicubic patches. These patches consist of doubly convex, doubly concave and saddle type surface. C0 and C1 discontinuities exist between some shared edges. Gouges are likely occur at the concave areas (refer to Color Plates 13 and 14).

**Table 5.1** Simulation results on five axis tool path generation

file	tool diameter (mm)	scale factor	tolerances	read time(min)	number of triangles	generation time (min)	total movements	max. gouge (mm)	max. excess (mm)
zip5	6.35	0.50	+/- 0.050	0.15	432	0.4	6937	0.041	0.101
saddle	6.35	0.50	+/- 0.050	0.56	9118	13.8	4789	0.046	0.169
z6324r	6.35	0.25	+/- 0.075	0.25	3324	1.8	2611	0.048	0.8194

Note:

Read time is the CPU time spent on surface triangulation.

Generation time is the CPU time for tool path generation only.

Tolerances are the Maximum Allowable Gouge and possible maximum excess amount.

Simulation results are reflected by maximum gouge and maximum excess amount<sup>[35]</sup>.

Test results are listed in Table 5.1. The maximum gouge caused by the tool paths generated for the test surfaces are below the maximum tolerance deviation. Excess material is left on the concave areas where the tools selected are too large to fit. In **zip5**, even though the number of tool positions is very large, only a small portion of the tool positions require adjustment. Therefore, **zip5** takes the least amount of time among the three test cases. **z6324r** is more complex than **zip5** and **saddle**. However, only a small concave area exists, and consequently, only a few of tool positions need modification. **Saddle** is a very curved composite surface with many triangles generated by the triangulation process. As discussed in Chapter 3, it takes more time to position a tool on a polygon than on a point. Note that the stepover between tool paths can be larger when using a flat-end tool in five-axis mode compared to a three axis ball-end tool of the same

diameter. Therefore, fewer tool paths are generated than with the TRI\_XYINDEX method. The results also show that the implemented algorithms can effectively handle the overlapping surfaces, and can detect and eliminate unnecessary gaps between surface patches.

## CHAPTER 6

### FIXED-AXIS TOOL PATH GENERATION

Along with the implementation of FIVEX\_INDEX described in chapter 5, a subsystem of tool path generation for a tool with fixed orientation is also developed. In the subsystem, FIX\_AXIS\_INDEX, the cutter axis is oriented at a fixed angle with respect to the workpiece. This orientation is a user defined angle and is measured from the +x axis and +z direction. When the tool moves along the +x axis in the xy plane (Fig. 6.1a), its axis is oriented with angle  $\alpha$  and  $\beta$  to the +x and +z axis respectively. If the tool moves in the -x axis, angle  $\alpha$  is measured from -x axis and  $\beta$  is unchanged (i.e., the tool axis is symmetric with respect to the z axis).

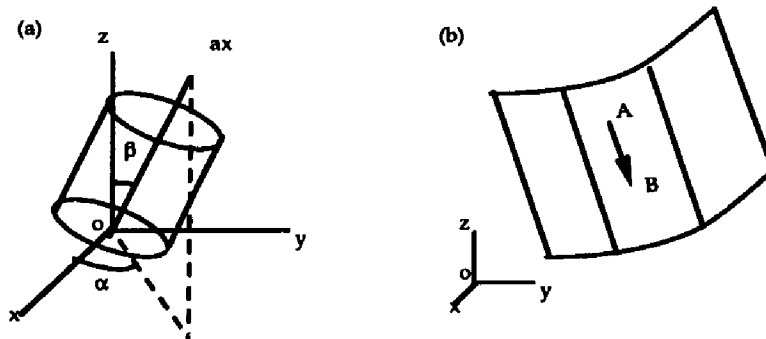


Figure 6.1 Fixed tool orientation

The fixed-axis approach is motivated by the fact that a flat-end tool is preferred for machining certain types of parts with special features such as the ruled surface patches, where the tool is confined to move along the non-curved direction (e.g., AB in Fig. 6.1b). From the discussion in Chapter 2, a tilted flat-end tool can be substituted for a ball-end tool with smaller cusp height for a given tool diameter and stepover distance. In the following paragraphs, the implementation of FIX\_AXIS\_INDEX is discussed. Since it is based on

the five-axis system discussed in Chapter 5, only the differences between the two are highlighted.

## **6.1 IMPLEMENTATION OVERVIEW**

In `FIX_AXIS_INDEX`, an object surface is also approximated by the STS. Tool paths are generated from CC points with an approach similar to the five axis method explained in Chapter 5. Since the tool axis is fixed, the stepover between adjacent tool paths can be calculated with a simpler algorithm. Tool positions are verified and corrected with an algorithm that sets the tool to the highest touch point along the tool axis direction. To simplify the calculation, triangles are transformed to a new coordinate system in which the tool axis is parallel to the z axis.

With this algorithm, excess material may be left at trailing edge positions. To remove this material efficiently, tool positions at trailing edges are modified with an extrapolation algorithm which extends the tool paths. Intermediate tool positions are interpolated between two tool positions derived from CC points. The interpolated tool positions are added to tool paths if gouging is detected during verification.

## **6.2 TOOL POSITION INTERPOLATION**

Control Polylines (the edge lists) are the locus of candidate contact points (CC points) on the object surface touched by a flat-end tool with its front edge. As discussed in Chapter 5, there are an infinite number of tool center locations that can be offset from a CC point. However, we select the tool location which is in the plane (called cutter center plane) formed by the tool axis and the tool moving direction. When the tool moves from one center position to another, its front edge always touches the control segment. If this movement does not cause a gouge, its related tool center locations are written to the CL data file. This tool movement must be modified if gouge occurs.

Tool path verification is an important topic in NC research. The envelope of a three-axis tool movement can be tested to see if the envelope collides with the object surface. However, for tool path generation we must also be able to modify the envelope to avoid



gouging. To achieve these tasks, a simple method based on tool movement discretization is implemented.

As discussed in Chapter 5, an object surface is approximated by a polygon set. Assume that a, b and c (Fig. 6.2a) are three consecutive tool center locations offset from the vertices of control segments (i.e., the intersection segments of the cutting planes with the polygon set). Based on the distances between adjacent tool center locations, some tool positions are interpolated. The maximum spacing is controlled by  $\delta$  derived from equation (6.1) so that the maximum gouge caused by the tool movement does not exceed the given gouge limit (c) (Fig. 6.3). After the insertion, each individual tool movement is verified and modified if necessary.

$$\delta = 2\sqrt{2rc - c^2} \quad (6.1)$$

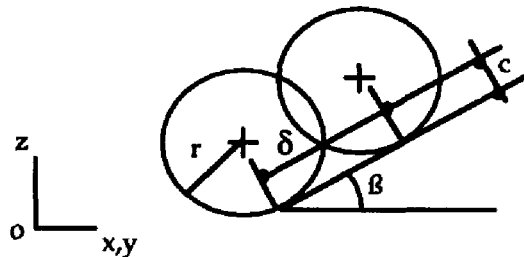
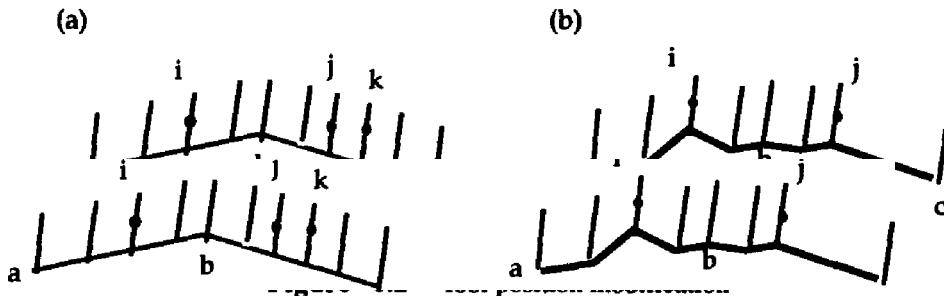


Figure 6.3 Default stepsize

### 6.3 TOOL POSITION CORRECTION

The mathematical concepts for correcting tool positions are illustrated with Fig. 6.4. Assume that the tool moves upward (i.e., from the initial position it moves along the  $+\bar{ax}$  direction). For a given triangle  $abc$ , an initial tool center location  $tc$ , and tool orientation  $\bar{ax}$ , transpose the coordinate system so that the  $z$  axis is parallel to  $\bar{ax}$  (Fig. 6.4b). This operation is just for simplifying the calculation. If the projection of  $\Delta abc$  onto the  $xy$  plane overlaps the tool projection (the tool shadow) as shown in Fig. 6.4c, the tool center location  $z$  for a given  $x$  and  $y$  position is found such that the tool contacts the triangle either at a vertex, an edge or inside triangle. If  $z$  is larger than the initial  $z$  value, the tool is lifted. The final lift amount is the maximum for all triangles tested. The algorithm is Given in Algorithm 6.1.

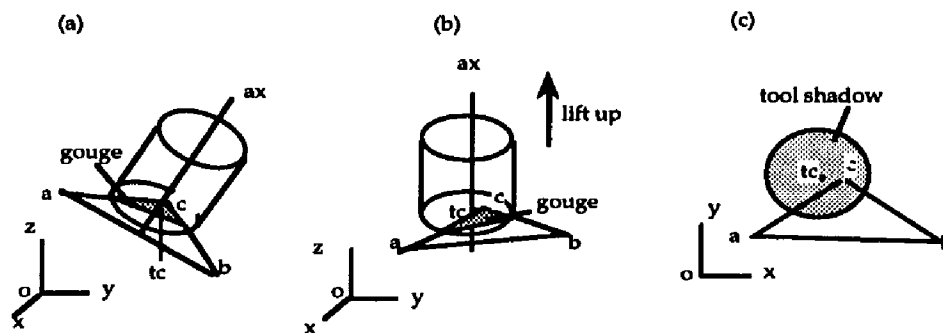


Figure 6.4 Transformation and tool shadow

#### Algorithm 6.1 Tool Position Modification

lift\_amount = zero

For each triangle in the search domain

    Transform the triangle

    if its projection overlaps with the tool shadow

        find tool center position  $z$  for given  $x$  and  $y$

        if  $z - \text{initial } z > \text{lift\_amount}$

            save the difference into lift\_amount

If lift\_amount  $\neq$  zero

    set lifting flag and lift the tool up by lift\_amount

End of Algorithm 6.1

## 6.4 TOOL PATH MODIFICATION

The insertion of interpolated tool positions described in the last section results in a large number of unnecessary tool positions. In `FIX_AXIS_INDEX` unnecessary inserted tool positions are eliminated with a linear fitting scheme. The linear fitting algorithm is illustrated with the example shown in Fig. 6.2b, where the modified tool positions are marked by dots on their axis vectors (i, j and k). The first position a is an end position which is directly stored in the CL data file. Positions after a are tested. If the deviation of any position (i) to segment ab is larger than the required fitting limit, this position along with the one just before it (position i-1) are tested in the following way. If the deviation of position i-1 to segment ai is larger than the fitting limit, write the position i-1 and position i into the CL data file; else, only position i is saved. A similar process is applied to segment bc. For example (Fig. 6.2b), the deviation of the position j-1 is out of the limit while position j is not. Save position j-1 while all the tool positions after j and before c are removed since the deviations of these positions to segment jc are all within the limit.

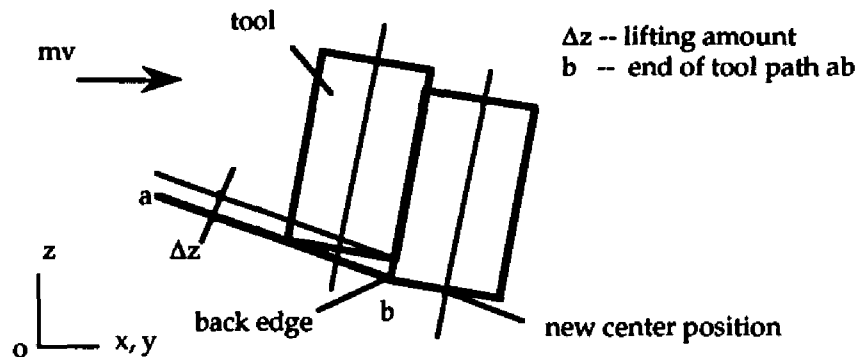


Figure 6.5 Tool path expansion

Tool paths formed with the modification described above may leave excess material at some modified tool positions. This means that after the tool position correction, the front edge of a flat-end tool no longer contacts the initial surface point. When the modified tool positions are not on the ends of individual tool paths, material left at these positions is

(partially) removed by following tool movements. However, for the modified positions at the ends of tool paths, excess material may be left (Fig. 6.5). To eliminate the excess material in FIX\_AXIS\_INDEX, tool paths are extended at the ends if the end tool positions are modified. By using the back edge of the flat-end tool to touch the contact point, a new tool center location is generated. This new position is regarded as the end of a tool path. Then the last newly added tool movement is verified with the process described above.

## 6.5 LOCALIZATION

Generally the size of a tool is quite small with respect to the design surface, and it would be inefficient to consider the whole surface when verifying an individual tool position. As in Chapter 5, triangles are sorted into buckets. For each tool position, only the polygons stored in the buckets overlapping the tool shadow are tested. One particular characteristic of a fixed-axis tool is the unchanged shape of the tool shadow. To avoid unnecessary calculations, the shape of the tool shadow is precalculated based on the tool length, radius and orientation. To simplify the algorithms for defining the tool shadow and for deriving the bucket domain for testing, the search domain is defined as the minimum rectangle that contains the tool shadow (Fig. 6.6).

It is worth mentioning that the fixed tilting angle is measured from the z axis to the tool axis. For zigzag cutting, the projection of the tool axis is symmetric about the origin of the system. This means that the tool shadow in the next path can be directly derived from the tool shadow geometry of the current tool path. For box cutting, the tool shadow is unchanged since the cutting is always goes in the same direction.

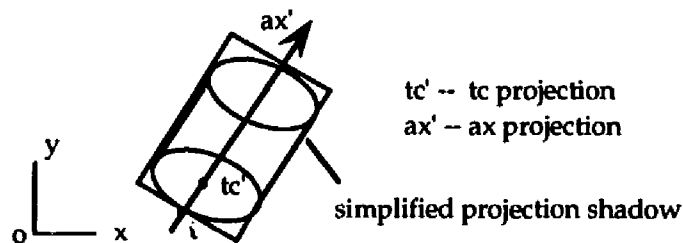


Figure 6.6 Localization of search domain

## 6.6 CALCULATION OF STEPOVER DISTANCE

As discussed in section 5.9, the cusp height is proportional to the angle between the tool movement direction and the tool bottom plane. Although the tool axis is fixed, the tool movement direction changes with the surface slope, thereby changing the effective radius and the cusp height between adjacent tool paths. In section 5.9, a method for defining stepover distance is discussed, in which only the slope across the tool moving direction is considered. This stepover can be used as an initial value, but to limit the cusp heights between adjacent tool paths, the initial stepover is modified based on the algorithm described below.

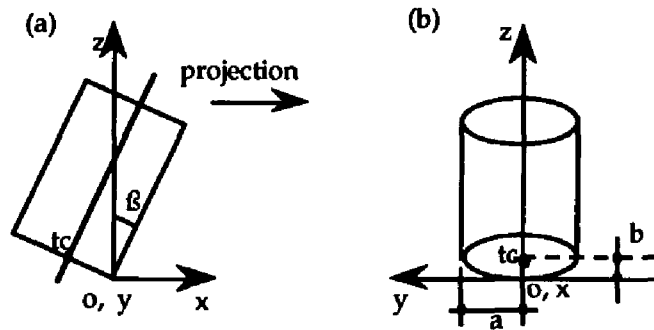


Figure 6.7 Tilting effect

Assume that the tool movement direction is parallel to the  $x$  axis. For a tilting angle  $\beta$  (Fig 6.7.a) in the  $xz$  plane, the silhouette of the tool projection onto the  $yz$  plane is composed of two half ellipses and one rectangle. The most interesting part of the silhouette is the half ellipse projected by the bottom. Assume that  $tc$  is the center of the ellipse on the  $yz$  plane with  $a$  and  $b$  as its long and short axis; the mathematical equation for the ellipse is given by eqn. (6.1). As in Chapter 5, a circle is used to approximate the ellipse (Fig. 6.7b); the resultant radius is referred as the effective radius  $r_e$ . Equation (5.5) gives the relationship between  $r$  and  $r_e$ . From this equation, it can be seen that the effective radius is always larger than the long axis unless the tilting angle equals  $90^\circ$  in which case  $r_e = r$ . Comparing the curvature of the ellipse at the bottom half with that of the circle, the arc of

the circle below the tool center is always within that of the ellipse. Therefore, when the tool positions are not too far apart (i.e., the intersection of two ellipses is within the bottom half ellipse), the actual cusp height does not exceed that of the two circles approximating the ellipses. This also indicates that generally a tilted flat-end tool provides better surface quality than a ball-end tool of the same size.

$$\frac{y^2}{a^2} + \frac{(z-b)^2}{h^2} = 1 \quad (6.1)$$

Note that in the above section, the silhouette of the tool projection is derived under the assumption that the front edge of a tool moves along the x axis. Since most sculptured surfaces are not flat, the angle between the tool axis and the tool moving direction changes with the tool position. In Fig. 6.8,  $\alpha$  is the angle between the xy plane and the tool moving direction, and  $\theta$  is the angle between the z axis and the tool axis. The silhouette of the tool bottom on the yz plane varies with the change of  $\theta$ . The overall tilting angle ( $\beta$ ) required for deriving the silhouette of the tool projection at each individual tool position is the sum of  $\alpha$  and  $\theta$  if the tool moves towards the +z axis, or the subtraction of  $\theta$  by  $\alpha$  if the tool moves towards the -z axis. In Fig. 6.9, p1 and p2 are two surface contact points at adjacent CC curves, c is the cusp height between the ellipses, o1 and o2 are tool center locations with respect to touch points p1 and p2 respectively, and  $\Delta s$  is the stepover distance.

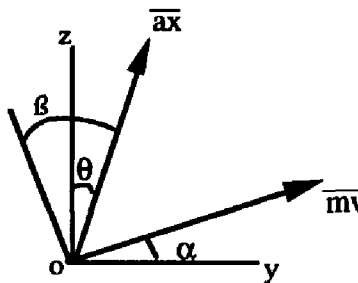
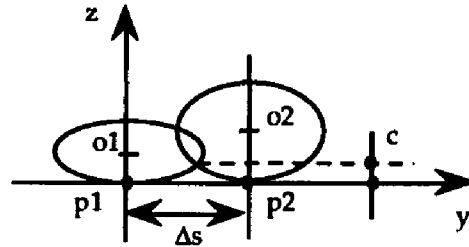


Figure 6.8 Tilting angle



**Figure 6.9** Silhouette of the tool bottom on the yz plane

The required stepsize for each tool path is derived based on the allowable cusp height. Assume the allowable cusp height is  $c$ ,  $a_1$  and  $a_2$  are the long axes, and  $b_1$  and  $b_2$  are the short axes of ellipses  $o_1$  and  $o_2$  respectively. If setting  $z = c$  in eqn. 6.1, we can solve for the stepover distance  $\Delta s$ .

$$\Delta s = \frac{a_1}{b_1} \sqrt{2b_1c - c^2} + \frac{a_2}{b_2} \sqrt{2b_2c - c^2} \quad (6.2)$$

Since there are many tool positions on each tool path, the actual stepover is set to the minimum among a set of sampled pairs at various points on the tool path. The sampled pairs are chosen at an interval equal to one tenth of the tool radius. Note that the overall tilting angle is confined to the xz plane when deriving equation (6.2). If the tool axis is not in the xz plane, assume an angle  $\alpha$  (not equal to  $90^\circ$ ) between the x axis and the projection of the tool axis onto the xy plane (Fig. 6.10a), the actual stepsize  $\Delta s'$  is defined by equation (6.3). The relationship of the long axis of each ellipse and tilting angle  $\alpha$  is defined in equation (6.4). Fig. 6.11b shows that the long axis shrinks as  $\alpha$  increases. Assume  $a'$  is the long axis for  $\alpha \neq 0$ ,

$$\Delta s' = \Delta s \cos\alpha \quad (6.3)$$

$$a' = a \cos\alpha \quad (6.4)$$

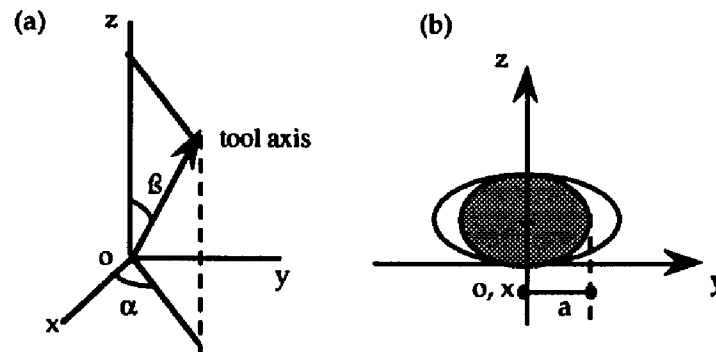


Figure 6.10 Tilting angle  $\alpha$

Note that if the tool axis is located in the  $xz$  plane,  $a$  equals the tool radius and  $b$  equals  $r\sin\beta$ . For  $\beta_i \neq 0$  and  $\alpha \neq 90^\circ$

$$\Delta s = \frac{\cos\alpha}{\sin\beta_1} \sqrt{2\sin\beta_1 rc - c^2} + \frac{\cos\alpha}{\sin\beta_2} \sqrt{2\sin\beta_2 rc - c^2} \quad (6.5)$$

## 6.7 TEST RESULTS

The algorithms discussed in this chapter are implemented in `FIX_AXIS_INDEX` on the Silicon Graphics Indigo R4000 workstation and were tested with the surfaces: `zip1`, `saddle` and `z6324r` (refer to section 5.4). The results are listed in Table 6.1 - 6.3. It shows that tool paths generated are gouge free (i.e., below the maximum gouge allowance). In `zip5`, the two convex surface patches are ruled surfaces. By adjusting the initial tool axis orientation, tool paths of accuracy equivalent to those of `FIVEX_INDEX` are generated (refer to Color Plates 15 and 16). For instance, by setting the tool at angle  $x = 0^\circ$  and  $z = 15^\circ$ , the material left on the surface is close to that of `FIVEX_INDEX` (refer to Table 5.1).

*Saddle* is a very curved composite surface. Though the tool orientation is fixed, the angle between the surface normal and the tool axis changes along tool paths. At some tool positions the angles are very large while at others they are small. The tool position with a small angle may not fit into a very curved concave area where excess material is left. Such



is the case of *z6324r*. On the other hand, *saddle* and *z6324r* are very curved across the cutting direction. To avoid high cusps left between adjacent tool paths, densely distributed tool paths are required.

The tool shadow expands as the angle between the tool axis and the z axis increases. More triangles are selected for gouge checking so more CPU time is required. Also note that except in zigzag cutting, machining with fixed-axis tools is identical to three-axis machining. In zigzag cutting, though the tool axis doesn't change along individual tool paths, it requires five axis machining to accomplish the change of tool orientation for opposite directions of movement.

**Table 6.1** Simulation Results on Fixed-Axis Tool Paths for *zip5*

Design File: <i>zip5</i>							
Tool Diameter: 6.35 (mm); Scale Factor: 0.5; Tolerances: +/- 0.05 (mm)							
tool axis orientation		point no.	triangle no.	generation time (min)	total movements	max. gouge (mm)	max. excess (mm)
x°	z°						
0	0	246	432	0.24	2027	0.024	0.306
0	5	246	432	0.50	1992	0.011	0.161
0	15	246	432	0.83	1867	0.018	0.102
0	20	246	432	0.90	1699	0.019	0.115
0	30	246	432	1.00	1538	0.017	0.147

**Table 6.2** Simulation Results on Fixed-Axis Tool Paths for *z6324r*

Design File: <i>z6324r</i>							
Tool Diameter: 6.35 (mm); Scale Factor: 0.25; Tolerances: +/- 0.075 (mm)							
tool axis orientation		point no.	triangle no.	generation time (min)	total movements	max. gouge (mm)	max. excess (mm)
x°	z°						
0	0	2292	3324	2.2	6609	0.045	3.0
0	10	2292	3324	4.9	6719	0.046	3.0
0	15	2292	3324	5.8	6660	0.048	3.0

**Table 6.3** Simulation Results on Fixed-Axis Tool Paths for *saddle*

Design File: <i>saddle</i>							
Tool Diameter: 6.35 (mm); Scale Factor: 0.5; Tolerances: +/- 0.05 (mm)							
tool axis orientation		point no.	triangle no.	generation time (min)	total movements	max. gouge (mm)	max. excess (mm)
x°	z°						
0	0	5628	9118	3.7	8965	0.050	1.765
0	3	5628	9118	7.0	9213	0.048	1.754
0	5	5628	9118	9.3	9062	0.043	1.754
0	15	5628	9118	16.5	8988	0.049	1.754

## **CHAPTER 7**

### **CONCLUSIONS AND RECOMMENDATIONS**

We have developed a three-axis tool path generation system (TRI\_XYINDEX), a three-axis tool path generation system for finish machining (FINISH), a five-axis tool path generation system for flat-end tools (FIVEX\_INDEX) and a tool path generation system for tools with fixed orientations (FIX\_AXIS\_INDEX). The common aspects of these systems are: 1. All the tool paths generated are within the accuracy of the tolerances specified by the user. 2. The systems require minimal user interaction. After the parameters have been defined for the cutting conditions, the programs run with no further interaction. 3. The systems are beneficial to the manufacturing industry since they generate tool paths which are gouge-free and the cycle of simulation and testing of tool paths is no longer necessary. Therefore, the user does not have to be an expert in NC machining. In the following sections, conclusions and future work for each system is discussed separately.

#### **7.1 THREE-AXIS MACHINING**

TRI\_XYINDEX is based on surface polygonalization. It serves as an alternative to the three-axis tool path generation system developed by Angleton<sup>[31]</sup> which was based on a point approximation of the surface. Compared to a point based approximation, polygonal approximation has the advantage such that it does not depend on the tool shape and size. If the cutter is positioned to touch the polygons, the error for an individual tool position can not exceed the deviation from the surface approximation. Though more time is required to position a cutter on a polygon than on a point, the number of polygons considered for each tool position is less than the number of points. This trade off works in favor of the polygon method when the object is relatively flat. The corner of a flat-end tool can protrude into the surface between points more than a ball-end tool. To ensure that the tool

is positioned within the required tolerance, many more points are required when using a flat-end cutter rather than a ball-end. The tool path generation process may be slowed down since many more points are checked for each tool position.

Simulation results using TRI\_XYINDEX show that for relative flat surfaces, tool path generation based on polygons is more efficient than point based methods. For highly curved surfaces, point methods are more efficient. The edge handling approaches developed in TRI\_XYINDEX can also be applied to the point or tangent plane. With some modification, the algorithms for finding a touch point on a triangle can also be used in five axis tool path generation based on surface triangulation.

## **7.2 FINISH MACHINING**

Based on the surface point set, FINISH generates more efficient tool paths for three-axis finishing of sculptured surfaces than the traditional approach. Among the three tool path planning schemes, inner-pruning is favored if the undercut areas form closed loops. Block cutting and outer-pruning cutting take less time to generate, and are better choices in machining undercut areas without holes.

Machining results show that a smooth transition can be provided by applying the expanded tool radius technique for proper landing and take-off from the surface for a ball-end cutter. A similar approach can also be applied to flat-end tools with the tool expansion in two directions, one along the tool axis and the other in the radial direction. At the tool corner the expanded tool is filled with a quarter of a torus and a fillet-end tool is formed. Dwell marks can be eliminated by forcing the size of the fillet-end tool to follow the predefined parabolic curve as in section 4.4. However, more tool positions are required to form an equivalent smooth transition because of the sharp edges of flat-end tools. Similarly, the expansion technique can be applied to fillet-end tools as well. Tool path generation approaches can be directly applied to systems based on other surface representations such as polygonalization or offset surfaces. The method of undercut area identification and formation can also be applied to other systems provided that undercut areas are presented by discrete points.

### 7.2.1 Future Work

The cutting style in FINISH is chosen by the user and is fixed for all undercut pockets. It would be possible to automate this selection process. For example, the cutting style can be chosen according to the distribution of the undercut points and ratio of empty buckets to non-empty ones in a pocket. Tool paths are currently confined to straight lines. In some cases, curved paths would be appreciably better, such as the case shown in (Fig. 7.1). Tool path generation would be similar to that of contour machining of pockets. To arrange curved tool paths on undercut areas, 2D patterns (on the xy plane) formed by the uncut points must be discerned first which will require significant data analysis and sorting.

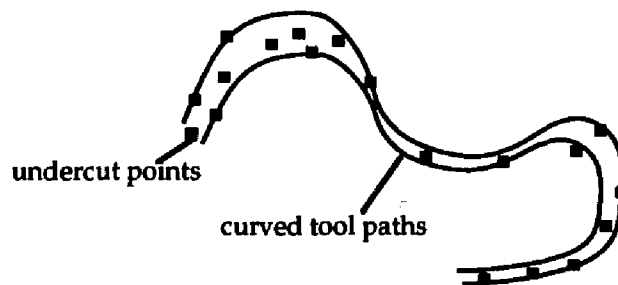


Figure 7.1 Curved tool paths

### 7.3 FIVE-AXIS MACHINING

In five-axis machining, a tool can be oriented at any angle to the workpiece. Therefore, five-axis NC programs should be able to achieve acceptable surface quality with fewer tool paths. To match the surface with the tool bottom as much as possible, a flat-end tool is often used in five-axis milling. As discussed in section 7.1, surface polygonalization does not depend on the tool shape and size. It is preferred for flat-end tools compared to the point based approach. Representing the object with polygons increases the complexity in calculation compared to points. However, it provides the possibility of fulfilling some tasks with simpler algorithms.

FIVEX\_INDEX is based on surface polygonalization. As discussed in Chapter 5, using edge lists to generate CC points is much more straightforward and efficient than the traditional approaches which spend much CPU time on deriving acceptable step-forward increments. Defining the CC points as the locations of the tool front edge, CLs are directly calculated from the CC points and CC point normals avoiding the iterative calculation for finding CLs which is necessary in the parametric indexing approach. To guarantee gouge-free tool paths, CLs are verified and modified if gouging occurs. The tool angular orientation is adjusted to keep the tool in contact with the CC point while lifting the heel of the tool to avoid gouging. If the tool position can not be adjusted by tilting, it is lifted along the tool axis direction. Calculation efficiency is achieved by the localization of calculations in interference checking and tool position modification.

Unlike the parametric indexing method, tool paths are planned in Cartesian space. Therefore, non-evenly distributed tool paths are eliminated. An object surface is represented by a single set of polygons. The difficult problem in parametric indexing of avoiding gouges with adjacent surfaces is eliminated. The algorithms used in tool positioning are almost entirely direct solutions which do not require troublesome iterative solutions.

Though the polygons currently used are triangles, most of the algorithms are directly applicable to N sided polygons. Only the algorithm used to locate a point with respect to a triangle needs some modification.

Compared with TRI\_XYINDEX, FIVEX\_INDEX requires fewer tool paths to produce equivalent accuracy. However, positioning and adjusting the angle requires more CPU time than simply setting the tool along the tool axis direction. FIVEX\_INDEX generally requires more CPU time than TRI\_XYINDEX, especially when many initial tool positions need correcting.

Simulation results of FIVEX\_INDEX also show that the edge list generation approach can effectively handle overlapping surfaces and gaps between surface patches. Most importantly, gouge-free five axis tool paths are generated.

### 7.3.1 Future Work

The number of tool positions generated by FIVEX\_INDEX could be further reduced by using fitting algorithms such as the linear fitting algorithm used in FIX\_AXIS\_INDEX. However, FIVEX\_INDEX requires linear fitting of two quantities: tool position and tool axis orientation. The number of required tool positions could be dramatically decreased if the linear fitting algorithm is implemented, especially when surfaces are smooth and convex.

Currently, the tilting angle is found in the plane formed by the initial tool axis and the tool moving direction. For cutting with the front edge of a flat-end tool, this technique is acceptable. However, there exists the possibility that gouging may be caused by the side of a tool instead of the back edge. In this case, it would be better to tilt the tool in the plane formed by the initial tool axis and the surface normal at the gouge point. The difficulty of defining a good tilting plane lies in the fact that a tool may gouge more than one place at a time.

Final tool positions are set in the tool movement verification process. The verification is currently accomplished by checking individual interpolated tool positions. This verification technique, though relatively accurate, is not very efficient. A better approach might be based on the swept volume modification, in which the swept volume formed by adjacent initial tool positions are checked for interference with the surface. Tool positions are modified and/or inserted only at the places of the swept volume where the interference is located.

Tool path generation creates a CL data file which is later post processed into G codes for a specific machine tool. This approach works well for three-axis machining but not as well for five-axis machines. Five axis machines employ a wide variety of methods for achieving the tool angular orientation, including rotary-tilt table, rotating heads, articulated arms, etc. In all cases the angular excursion of the mechanism is limited. For example, our FADAL milling machine uses an A-B axis rotary tilt table. The angular excursion of the B axis is limited to range from  $-95^{\circ}$  to  $+5^{\circ}$ . Tool positions with orientation outside of this limit are reached by rotating the A-axis about  $180^{\circ}$  and then aligning the tool to the

transformed tool orientation. Even though fast rotational speed can be applied, much time is wasted.

To avoid the unnecessary lifting and rotating, an algorithm based on tool path segment sorting is proposed. Tool paths are divided into segments by the tool orientation limits. For example, assuming that the worktable is parallel to the  $xy$  plane and the rotation domain of the tilting table is  $0^\circ \sim 90^\circ$ , tool paths are divided by the tool positions with the extreme orientations (Fig. 7.2a). These segments are sorted and joined together based on the orientation domain. Fig. 7.2b illustrates a schematic view of tool paths after sorting. Sorting tool paths using this approach creates another problem: dwell marks left on the surface at the end of path segments. Consequently, the algorithms for optimizing tool paths based on sorting must provide the dwell mark elimination function discussed in Chapter 4.

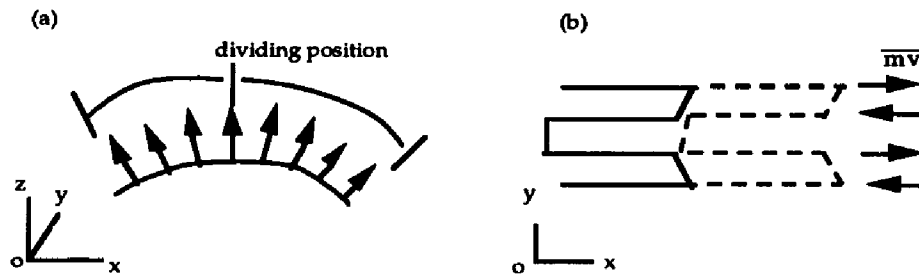


Figure 7.2 Tool path Sorting

Currently tool paths in FIVEX\_INDEX are generated with constant stepsize derived from equation (5.6). To achieve the desired level of accuracy, dense tool paths are required, especially for curved surfaces.

Also note that the angle adjustment based on triangles may require numerical solutions. To eliminate the possibility of using iteration to find a tilting angle, the hybrid of points and polygons can serve as an alternative tool positioning approach. Initial tool center locations are derived from CC points using polygons. Then the tool positions interfering with the surface are corrected with the angle adjustment based on points for which the exact solutions of the tilting angle can be directly calculated (i.e. the current vertex case).

Currently, tool path generation is based on purely geometric characteristics. It is possible that a new tool path generation approach may involve concepts from other fields such as physics. For instance, it is difficult to define the best tilting plane in the angle adjustment process. This problem may be solved with an approach based on the gravitational force of the object onto the tool. Based on the gravitational force of the design surface acted on the tool, the tilting plane can be defined. Then with this tilting plane, the “equilibrium” position of the tool can be found at which the tool will rest on the surface without gouging. To simplify this process, the object surface can be decomposed into a set of surface points (the SPS). Each point is treated as an object. The quantity of the gravitational force exerted by each object onto the tool can be defined based on its distance to the tool along the surface normal direction. A repelling force is assigned to the object interfering with the tool while an attracting force to the object away from the tool.

#### **7.4 FIXED-AXIS MACHINING**

**FIX\_AXIS\_SYSTEM** can be used as a three axis tool path generation system. By tilting the flat-end tool, equivalent ball-end tool effects can be achieved. For some surfaces such as zip5, it can produce tool paths with accuracy equivalent to **FIVEX\_INDEX** but with less CPU time. Simulation results also show that the linear fitting algorithm produces accurate and smooth tool paths.

As in **FIVEX\_INDEX**, this system simplifies the calculation of acceptable step-forward increments between surface contact points. Based on the edge lists (i.e. intersection segments), tool center locations with fixed orientations can be derived easily. These tool positions are lifted in the direction of the tool axis only if gouging occurs. Compared to **TRI\_XYINDEX**, this system can generate tool paths more rapidly, especially for convex surfaces in which tool positions directly derived from **CC** points do not require any modification.

There is only one tool orientation for box-cutting and two for zigzag cutting (one for each cutting direction). Currently, at each tool position, candidate triangles are transformed into the space in which the tool axis is parallel to the z axis. Efficiency could be improved by transforming all the triangles prior to calculating tool positions. This effectively



transforms the part so that the tool aligns with the z axis of a three-axis milling machine. In zigzag cutting, two tool axis orientations are used and, therefore, two set of transformed triangles are required.

## APPENDIX A

### DATA STRUCTURES IN TOOL PATH GENERATION

Fig. A1 shows the data structure used to store point information from the surface discretization or triangulation. The information is stored in SIMPATCH. When truncation lines were added, it was necessary to go to N sided polygons. Each surface patch defined by SIMPATCH contains a pointer to the TRIANGLELIST, a pointer to the POINTLIST and a pointer to the POLYGONLIST. Each TRIANGLELIST has three pointers to the POINTLIST. While each POLYGONLIST contains a linked list of vertices (VERTEXLIST) which consist of a set of pointers to the global array of points. The data structure is defined in struct.h.

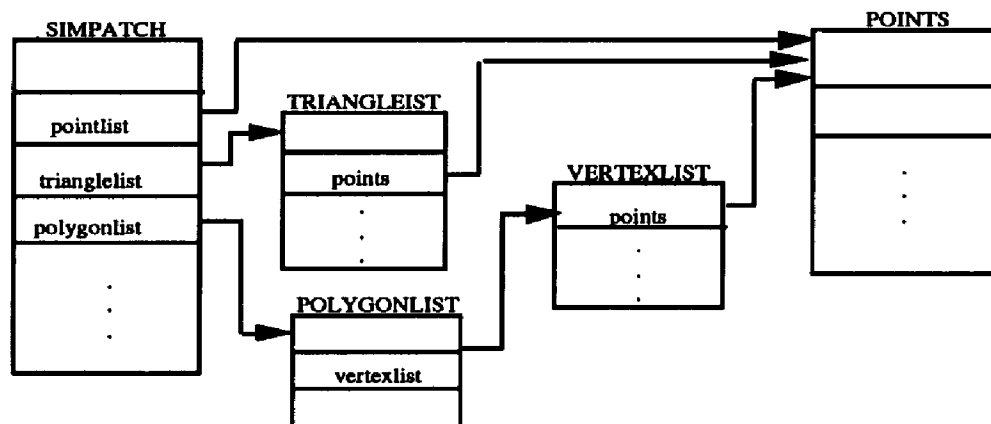


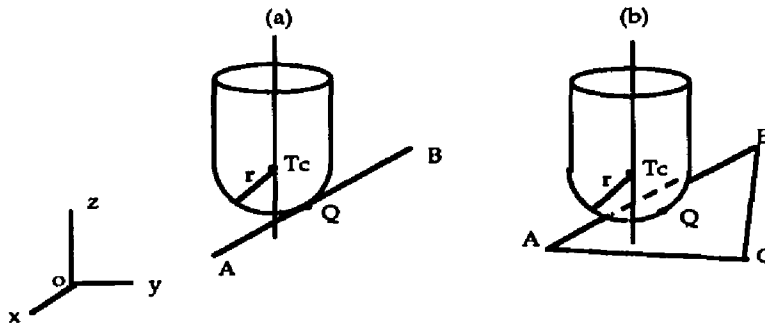
Figure A1 Data structures used in tool path generation systems

## APPENDIX B

### BALL-END TOOL POSITIONING WITH POLYGONS

#### Case 1: The Ball-End Tool Touches an Edge

Given  $x$  and  $y$  components of a tool center location and 2 vertices of an edge, assuming that the tool moves down along the  $z$  axis, find the  $z$  component of the tool center where the tool first touches the edge with its ball-end (Fig. B 1a).



**Figure B1** Tool contacting an edge or a triangle plane

Refer to Fig. B 1a,

- Q touch point of the tool at edge AB
- Tc tool center
- r tool radius
- A,B two vertices of an edge

If Tc is the position where the tool contacts the touch point Q,

then

$$|\overline{T_c Q}| = R \quad (\text{B1})$$

$$\overline{T_c Q} \cdot \overline{AB} = 0 \quad (\text{B2})$$

$$\overline{Q} = \overline{A} + t \cdot \overline{AB}/|\overline{AB}|, \quad \text{for } Q \text{ on edge } AB, t \in [0,1]. \quad (\text{B3})$$

Substitute

$$\overline{AB}/|AB| = \{l, m, n\}$$

$$Q = \{x_q, y_q, z_q\}$$

$$T_c = \{x_c, y_c, z_c\}$$

$$A = \{x_a, y_a, z_a\}$$

into (B1) - (B3) for  $n \neq 0$

then for given  $x_c$  and  $y_c$ ,

$$z_q = 0.5 (-b + \sqrt{b^2 - 4c(1-n^2)}) \quad (B4)$$

$$x_q = x_c + (z_q - z_a) * l/n \quad (B5)$$

$$y_q = y_c + (z_q - z_a) * m/n \quad (B6)$$

$$z_c = [z_q/n - (l*a + m*b)]/n \quad (B7)$$

where

$$b = -2n(l*u + m*v) u$$

$$c = n^4(u^2 + v^2 - r^2) + n^2(l*a + m*b)^2$$

$$u = x_c - x_a + z_a * l/n$$

$$v = y_c - y_a + z_a * m/n$$

If  $n = 0$ ,  $x_q = x_c$ ,  $y_q = y_c$ ,  $z_q = z_a - R$ , and  $z_c = z_a$ .

## Case 2: The Ball-End Tool Touches a Triangle Plane

Fig. B1b shows a tool touching a point on a triangle plane. Assume that the tool moves down along the z axis to touch a triangle. For given x and y components of the tool center, the z component of the tool center ( $z_c$ ) and the touch point (Q) on the triangle plane are calculated from equations (B8) - (B10).

In Fig. B1b,

A,B, C	3 vertices of a triangle
N	unit normal to the triangle plane
Q	touch point on the triangle plane
r	tool radius

If the tool with center  $T_c$  touches a point (Q) on the triangle plane,

$$\overline{T_cQ} \times \overline{N} = 0 \quad (\text{B8})$$

$$|\overline{T_cQ}| = r \quad (\text{B9})$$

$$\overline{QA} \cdot \overline{N} = 0 \quad (\text{B10})$$

Substituting

$$N = \{n_x, n_y, n_z\}$$

$$Q = \{x_q, y_q, z_q\}$$

$$T_c = \{x_c, y_c, z_c\}$$

$$A = \{x_a, y_a, z_a\} \quad \text{into (B8) - (B10),}$$

yields

$$z_q = z_a - [(x_q - x_a) \cdot n_x + (y_q - y_a) \cdot n_y] / n_z \quad (\text{B11})$$

$$x_q = x_c - (z_c - z_q) \cdot n_x / n_z \quad (\text{B12})$$

$$y_q = y_c - (z_c - z_q) \cdot n_y / n_z \quad (\text{B13})$$

$$z_c = z_q + r \cdot n_z \quad (\text{B14})$$

## APPENDIX C

### FLAT-END TOOL POSITIONING ON AN EDGE OR A TRIANGLE

#### Case 1: The Flat-End Tool Touches a Triangle Plane

Given triangle ABC, and x and y components of the tool center c (Fig. c1), assuming that the tool moves down along the z axis, the z component of the tool center and touch point Q on the triangle plane are derived as follows.

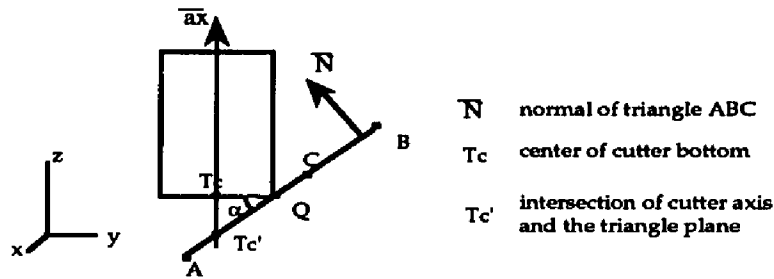


Figure C1 Finding a touch point on a triangle

For the tool with center at  $T_c$  touching a point (Q) on the triangle plane,

$$|\overline{T_c Q}| = R \quad (C1)$$

$$Q_z = T_{c_z} \quad (C2)$$

$$\overline{Q T_{c'}} \cdot \bar{N} = 0 \quad (C3)$$

$$\overline{T_{c'} A} \cdot \bar{N} = 0 \quad (C4)$$

$$T_{c_z} = T_{c'_z} + r \tan \alpha \quad (C5)$$

where

- |                          |                                    |
|--------------------------|------------------------------------|
| $A = (x_a, y_a, z_a),$   | a vertex of an edge,               |
| $N = (n_x, n_y, n_z),$   | unit vector of the line direction, |
| $Q = (x_q, y_q, z_q),$   | touch point                        |
| $T_c = (x_c, y_c, z_c),$ | tool center                        |

$$Tc' = (x_c, y_c, z')$$

$$\tan \alpha = \sqrt{(1 - n_z^2)/n_z^2}$$

R tool radius

From equation (C1) - (C5),

$$z_q = z_c = z_a + [(x_a - x_q)n_x + (y_a - y_q)n_y]/n_z + R \cdot \tan \alpha$$

$$x_q = x_c - R \cdot n_x / \sqrt{1 - n_z^2}$$

$$y_q = y_c - R \cdot n_y / \sqrt{1 - n_z^2}$$

### Case 2: The Flat-End Tool Touches an Edge

The derivation of the tool center location at which the flat-end tool touches an edge is discussed below. Assume that the tool moves down along the z axis and Q is the touch point on the edge. For given  $x_c$  and  $y_c$  of the tool center,  $z_c$  and Q are derived from equation (C4) - (C5).

$$(x_c - x_q)^2 + (y_c - y_q)^2 = R^2 \quad (C4)$$

$$\frac{x_a - x_q}{n_x} = \frac{y_a - y_q}{n_y} = \frac{z_a - z_q}{n_z} \quad (C5)$$

Assume

$$a = 1 - n^2$$

$$b = 2n_z[n_x(x_a - x_c) + n_y(y_a - y_c)]$$

$$c = 2n_z[(x_a - x_c)^2 + (y_a - y_c)^2 - R^2]$$

then

$$\Delta z = (\sqrt{b^2 - 4ac} - b)/2a$$

$$x_q = x_a - n_x \Delta z / n_z$$

$$y_q = y_a - n_y \Delta z / n_z$$

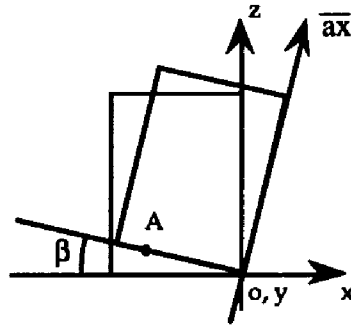
$$z_q = z_a + \Delta z$$

$$z_c = z_q$$

## APPENDIX D

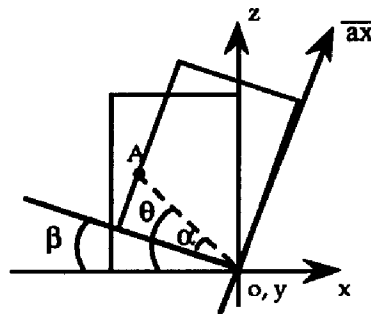
### TILTING ANGLE FOR THE VERTEX CASE

Refer to Fig. D1 and Fig. D2, A is a vertex,  $\overline{ax}$  is the tool axis, o is the origin and the location of the tool front edge. The aim is to calculate the minimum  $\beta$  that eliminates interference between the tool and vertex A. Note that a tool touches an edge with its bottom or side. The related tilting angle ( $\beta$ ) for the two possibilities are derived below. Fig. D1 and D2 represents the situations that occur when the vertex is touched by the tool bottom and side respectively.



$$\beta = \tan^{-1} \frac{A_z}{\sqrt{A_x^2 + A_y^2}}$$

**Figure D1** Tool bottom touching a vertex



$$\beta = \theta - \alpha$$

$$\theta = \tan^{-1} \frac{A_z}{|A_x|}$$

$$\alpha = \cos^{-1} \frac{r + \sqrt{r^2 - A_y^2}}{\sqrt{A_x^2 + A_z^2}}$$

**Figure D2** Tool side touching a vertex



## APPENDIX E

### TILTING ANGLE FOR THE EDGE CASE

#### Case 1: The Tool Bottom Touches an Edge

In Fig. E1a, T is the point on edge AB touched by the tool rotating about the y axis. Note that the tool front edge is located at the origin and is not changed in the rotation process while the tool center (T<sub>c</sub>) and the tool axis orientation ( $\bar{ax}$ ) do. Assume that  $\beta$  is the minimum rotation angle for the tool to avoid gouging edge AB, then T<sub>c</sub> and  $\bar{ax}$  can be expressed by equation (E4) and (E5) respectively. The mathematical equations for deriving  $\beta$  are listed in equations (E1) - (E3).

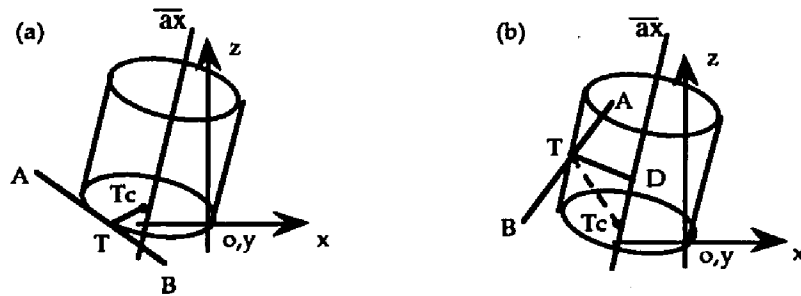


Figure E1 Tool touching an edge

$$\overline{T T_c} \cdot \bar{ax} = 0 \quad (E1)$$

$$|\overline{T T_c}| = R \quad (E2)$$

$$\bar{T} = \bar{A} + t \cdot \frac{\overline{AB}}{|\overline{AB}|}, \quad t \in [0,1] \quad (E3)$$

Substitute

$$\bar{ax} = (\sin\beta, \quad 0, \quad \cos\beta) \quad (E4)$$

$$T_c = (-R\cos\beta, \quad 0, \quad R\sin\beta) \quad (E5)$$

$$L = \overline{AB}/|\overline{AB}| = (l_x, l_y, l_z)$$

$$A = (x_a, y_a, z_a)$$

The explicit form of  $\beta$ :

$$0 = f(\beta) = \frac{1}{(l_x \sin \beta + l_z \cos \beta)^2} [g^2 + h^2 l_z^2 - 2gR l_x \sin \beta - 2gR l_z \cos \beta + 2(l_y g h + l_x l_z h^2) \sin \beta \cos \beta + (\frac{l_y^2}{l_z^2} g^2 + \frac{l_x l_y}{l_z} g h + l_x^2 h^2 - l_z^2 h^2) \sin^2 \beta]$$

where

$$g = l_x x_a - l_z z_a$$

$$h = y_a - \frac{l_y}{l_z} z_a$$

### Case 2: The Tool Side Touches an Edge

Fig. E1b shows a case where the tool touches an edge with its side when rotating about the y axis. For tilting angle  $\beta$ ,  $T_c$  and  $\bar{a}x$  are also defined with equation (E4) and (E5) respectively. If the tool with tilting angle  $\beta$  touches a point (T) on the edge (AB) with its side, then  $T_c$ ,  $\bar{a}x$ , and T must satisfy the simultaneous equations (E6) - (E9), assuming that D is a point on  $\bar{a}x$  and  $\overline{TD}$  is perpendicular to  $\bar{a}x$ .

$$|\overline{DT}| = R \quad (E6)$$

$$\overline{DT} \cdot \bar{L} = 0 \quad (E7)$$

$$\overline{DT} \cdot \bar{a}x = 0 \quad (E8)$$

$$\bar{T} = \bar{A} + t \cdot \bar{L} \quad (E9)$$

where

$$\bar{L} = \overline{AB}/|AB|$$

$$\bar{D} = \bar{T}_c + (\overline{TT}_c \cdot \bar{a}x) \cdot \bar{a}x$$

Substituting equation (E4) and (E5) into the above equations, yields

$$0 = f(\beta) = 2[-\alpha_0 + l_z z_a - l_x R \cos \beta + l_z R \sin \beta + \alpha_1 \cos \beta \sin \beta + (l_x x_a - l_z z_a) \sin^2 \beta] t + \alpha_2 + 2x_a R \cos \beta - 2z_a R \sin \beta - (x_a \sin \beta + z_a \cos \beta)^2 \quad (E10)$$

where

$$t = \frac{\alpha_0 + l_z z_a - l_x R \cos \beta + l_z R \sin \beta + \alpha_1 \cos \beta \sin \beta + (l_x x_a - l_z z_a) \sin^2 \beta}{(l_x \sin \beta + l_z \cos \beta)^2 - 1}$$

$$\alpha_0 = l_x x_a + l_y y_a + l_z z_a$$

$$\alpha_1 = l_x z_a + l_z x_a$$

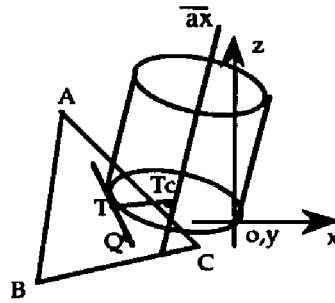
$$\alpha_2 = x_a^2 + y_a^2 + z_a^2$$

Equation (E10) is a non-linear equation, which requires numerical solutions.

## APPENDIX F

### TILTING ANGLE FOR THE TRIANGLE CASE

Assume that the tool initially interferes with triangle ABC (Fig. F1). By rotating around the y axis with the minimum angle  $\beta$ , the tool touches a point (T) on the triangle. Thus, the tool no longer gouges the triangle. The tilting angle ( $\beta$ ) is derived as follows.



**Figure F1** Tool Touching a point on a triangle plane

In the rotation process, tool center  $T_c$  and orientation  $\bar{a}x$  change with the angle  $\beta$  and are defined with equation (E4) and (E5) respectively. The intersection of the tool bottom plane with the triangle plane is represented by  $TQ$ , where  $T$  is the touch point on the triangle plane and  $Q$  is a point on the triangle plane defined by equation (F3) derived from equations (F1) and (F2).

From

$$\overline{QT_c} \cdot \bar{a}x = 0 \quad (F1)$$

$$\overline{QA} \cdot \bar{M} = 0 \quad (F2)$$

where

$$\bar{M} = (m_x, m_y, m_z), \quad \text{unit normal to the triangle plane}$$

$$A = (x_a, y_a, z_a), \quad \text{a vertex of the triangle ABC}$$

$$Q = (x_q, y_q, z_q)$$

$$\bar{a}x = N = (n_x, n_y, n_z), \quad \text{unit vector}$$

then

$$\frac{x_q}{-m_y n_z} = \frac{y_q - \frac{e}{m_y}}{m_x n_z - m_z n_x} = \frac{z_q}{m_y n_x}$$

where

$$e = m_x x_a + m_y y_a + m_z z_a$$

Therefore, the intersection line passes through point  $(0, e/m_y, 0)$  with line direction  $(-m_y n_x, m_x n_z - m_z n_x, m_y n_x)$ .

Define

$$Q = (0, e/m_y, 0) \quad (F3)$$

$$\bar{L} = (-m_y n_x, m_x n_z - m_z n_x, m_y n_x)$$

$$\bar{L}' = \bar{L}/|\bar{L}| = (l_x, l_y, l_z), \text{ unit vector}$$

For the tool with center  $T_c$  touching a point  $(T)$  on the triangle plane,  $T_c$  and  $T$  must satisfy the following equations:

$$\overline{TT_c} \cdot \bar{L}' = 0 \quad (F4)$$

$$|\overline{TT_c}| = R \quad (F5)$$

$$\overline{TT_c} \cdot \bar{a}_x = 0 \quad (F6)$$

$$\bar{T} = \bar{Q} + t \cdot \bar{L}', \quad t \in [0,1] \quad (F7)$$

Substituting (F3) and  $\bar{a}_x$  into above equations, the function of the tilting angle  $\beta$  for  $m_y \neq 0$  is as follows:

$$0 = f(\beta) = (y_q^2 - r^2)m_y + 2y_q r(m_x \cos\beta - m_z \sin\beta)$$

$$\therefore \sin(\beta - \varphi) = \frac{(y_q^2 - r^2)m_y}{2y_q r \sqrt{1 - m_y^2}}$$

$$\beta = \sin^{-1}\left(\frac{(y_q^2 - r^2)m_y}{2y_q r \sqrt{1 - m_y^2}}\right) + \varphi \quad (F8)$$

where

$$\varphi = \tan^{-1} \frac{m_x}{m_z}$$

If  $m_y = 0$ , the triangle ABC is perpendicular to the xz plane. Then

$$\overline{TA} \cdot \overline{M} = 0 \quad (\text{F9})$$

$$T_y = 0 \quad (\text{F10})$$

$$|\overline{TC}| = 2R \quad (\text{F11})$$

From (F9) :

$$(x_t - x_a) \cdot m_x + (z_t - z_a) \cdot m_z = 0 \quad (\text{F12})$$

substitute

$$t_x = -2R \cdot \cos\beta$$

$$t_z = 2R \cdot \sin\beta \quad \text{into (F12),}$$

yielding

$$\beta = \sin^{-1} \left( \frac{m_x \cdot x_a + m_z \cdot z_a}{2R} \right) + \tan^{-1} \left( \frac{m_x}{m_z} \right)$$

## REFERENCES

- 1 R B Jerard, K Hauck, R L Drysdale, *Simulation of Numerical Control Machining of Sculptured Surfaces*, International Symposium on Automotive Technology and Automation, Flims, Switzerland, Oct. 1986.
- 2 D. L. *Advanced Manufacturing Technology*, Delmar Publishers, Inc., 1992
- 3 J. F. Reintjes, *Numerical Control Making a New Technology*, Oxford University Press, 1991
- 4 A Baer, C Eastman and M Henrion, *Geometric Modeling: a Survey*, Displays Technology & Application, 1978.
- 5 M. E. Mortenson, *Geometric Modeling*, John Wiley & Sons., Inc., 1985
- 6 L. Piegl, *On NURBs: A Survey*, IEEE Computer Graphics & Applications, January 1991
- 7 P. C. Gasson, *Geometry of Spatial Forms*, Ellis Horward Series Mathematics and Its Application, 1983
- 8 C. M. Hoffmann, *Geometric and Solid Modeling*, Morgan Kaufmann Publisher, Inc., 1989
- 9 K Weiler, *Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments*, IEEE CG &A, January 1985.
- 10 Martti Mantyla, *An Introduction to Solid Modeling*, 1988, Computer Science Press, Inc., USA
- 11 P. Lancaster & K. Salkauskas, *Curve and Surface Fitting*, Harcourt Brace Jovanovich, 1990
- 12 W. R. Harris, *CAD Comes to the Surface*, Computer-Aided Engineering, February, 1989
- 13 J. C. Cavendish, S. P. Marin, *Feature-Based Surface Design and machining*, IEEE Computer Graphics and Applications, September 1992
- 14 J. C. Cavendish, S. P. Marin, *A Procedural Feature-Based Approach for Designing Functional Surfaces*, Society for Industrial And Applied Mathematics, Philadelphia, 1992, p148 - 165
- 15 H-P Wang, *The Use of Microcomputer in Numerical Control Instructions*, Software for Engineering Workstations, Vol. 4, July 1988.
- 16 I D Faux, M J Pratt, *Computational Geometry for Design and Manufacturing*, Ellis

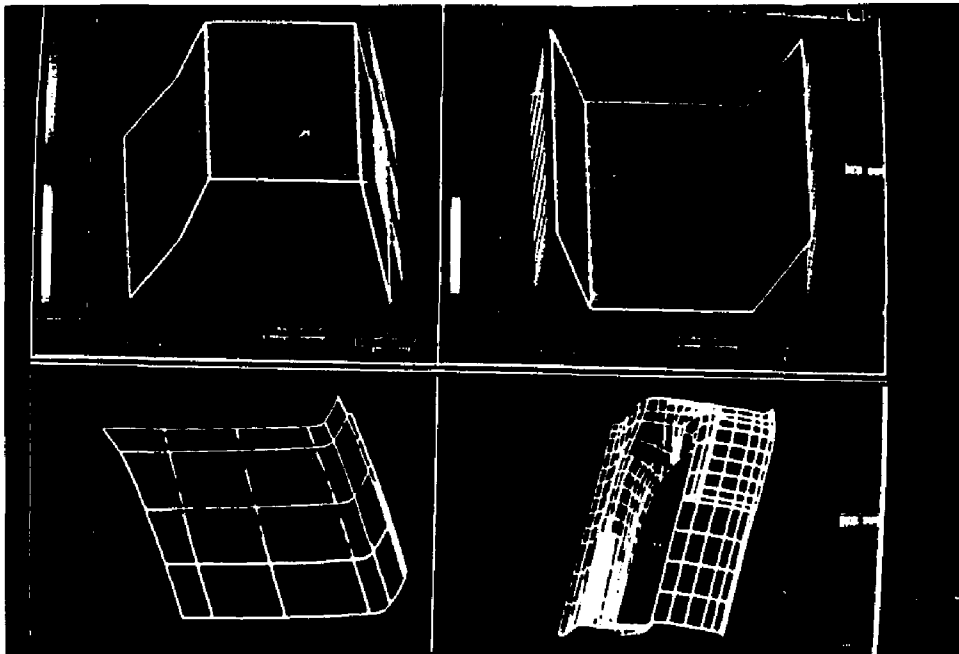
- Horwood, 1979, pp. 268 - 272.
- 17 W H Press, B P Flannery, S A Teukolsky, W T Vetterling, *Numerical Recipe in C*, Cambridge University Press, 1988
  - 18 G C Loney, T M Ozsoy, *NC machining of free form surfaces*, Computer Aided Design, Vol.19, No.2, Mach 1987, pp. 85-90.
  - 19 Y Huang, J H Oliver, *Non-constant parametric NC tool path generation on sculptured surfaces*, ASME-PED, WAM 1992
  - 20 S G Papaioannou, D Kiritsis, *An Application of Bertrand Curves and Surfaces to CAD/CAM*, Computer-Aided Design, Vol. 17, No. 8, Oct. 1985, PP 348 - 352.
  - 21 Y J Chen, B Ravani, *Offset Surface Generation and Contouring in Computer-Aided Design*, ASME Journal of Mechanisms, Transmissions & Automation in Design, Vol. 109, March 1987, pp.132-142.
  - 22 Y D Chen, J Ni, S M Wu, *Real-Time CNC Tool Path Generation for Machining IGES Surfaces*, Sensors, controls and Quality Issues in Manufacturing, ASME 1991, PP187 - 199.
  - 23 T Sakuta, M Kawai, Y Amano, *Development of an NC Machining System for Stamping Die by Offset Method*, Proceedings of Autofact'87, Society of Manufacturing Engineers, Detroit, Society of Manufacturing Engineers, Nov.1987, pp.2-13 - 2-27.
  - 24 B K Choi, C S Jun, *Ball-End Cutter Interference Avoidance in NC Machining of Sculptured Surfaces*, Computer-Aided Design, Vol. 21, No. 6, 1989, PP371 - 378.
  - 25 T Saito, T Takahashi, *NC Machining with G-buffer Method*, ACM, Computer Graphics, Vol. 25, No. 4, July 1991.
  - 26 J C Vogel, *Automated Machining of Custom Anatomical Models Using A Small-Scaled Integrated Facility*, Proceedings of Autofact'85, SME, Nov. 4- 7, 1985, PP 37 - 50.
  - 27 J M Angleton, *Automatic Generation and Correction of Tool Paths for Sculptured Surface Machining*, M.S. thesis, Computer Science Department, Univ. of New Hampshire, 1989.
  - 28 R B Jerard, R L Drysdale, et la, *Methods for Detecting Errors in Sculptured Surface Machining*, IEEE Computer Graphics and Applications, Jan.1989, pp. 26-39.
  - 29 A Hansen, F Arbab, *Fixed-Axis Tool Positioning with Built-in Global Interference Checking for NC Path Generation*, IEEE Journal of Robotics and Automation, vol.4, no.6, Dec. 1988, pp. 610-621.
  - 30 J P Duncan, S G Mair, *Sculptured Surfaces in Engineering and Medicine*, Cambridge University Press, 1983



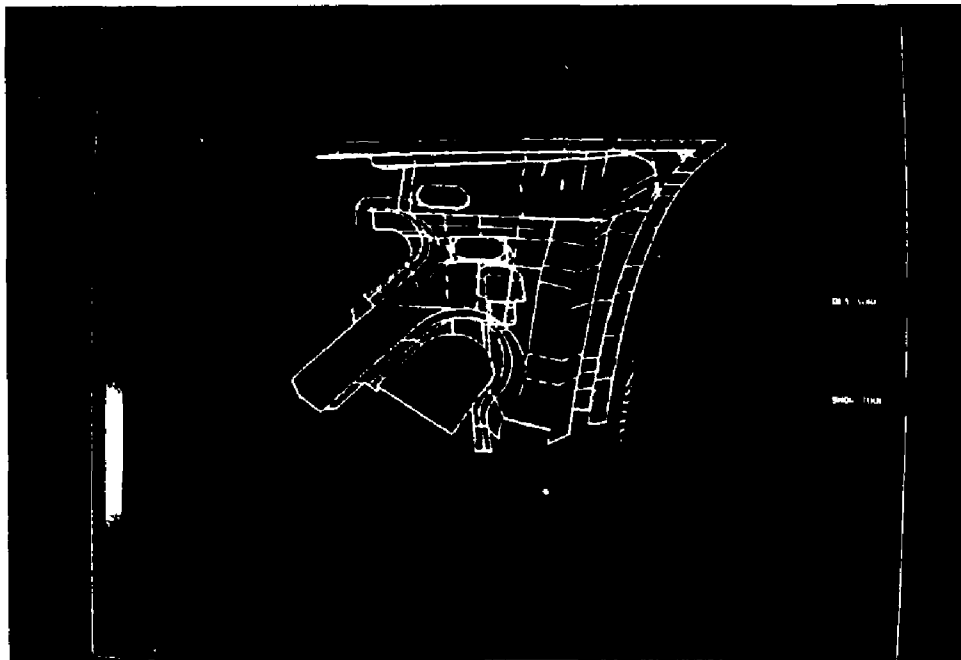
- 31 J P Duncan, K K Law, *Computer Aided Sculpture*, Combridge University Press, Combridge, UK, 1988
- 32 K. Ozair, *NC Machining Simulation and Verification Using Triangles rather than Points*, B.S. Thesis, Computer Science Department, Dartmouth College, 1990.
- 33 J J Chou, E Cohen, S H Drake, *Automatic Sculptured Five-Axis Milling with Check Surfaces*, Computer Science Department, Univ. of Utah, Tech report UUCS-89-010, April 1989.
- 34 C G Jensen, D C Anderson, *Accurate Tool Placement & Orientation for Finish Surface Machinig*, (School of Mech. Eng., Purdue University, West Lafayette, Indiana, 47907), Journal of Design and Manufacturing (1993), in press.
- 35 R B Jerard, S Z Hussian, R L drysdale, B Schaudt, *Approximate Methods for Simulation and Verification of Numerically Controlled Machining Programs*, The Visual Computer, Vol. 5, no. 6, 1989, pp. 329-348.
- 36 U A Sungurtekin, H B Voelcker, *Graphical Simulation and Automatic Verification of NC Machine Programs*, IEEE Publication, 1986, CH2282-2/86/0000/-156.
- 37 W A Hunt, H B Voelker, *An Explorary Study of Automatic Verification of Programs for Numerically Controlled Machine Tool*, Production Automation Project, Tech. Memo. No. 34, University of Rochester, NY, 1982.
- 38 R Fridshal, K P Cheng, D Duncan, N Zucker, *Numerical Control Part Program Verification System*, Proceedings of The Conference on CAD/CAM Technology in Mechanical Engineering, MIT, Cambridge, MA, USA, March 1982.
- 39 T V Hook, *Real - Time Shaded NC Milling Display*, SIGGRAPH, Vol. 20, N. 4, 1986.
- 40 P Atherton, C Earl, C Fred, *A Graphical Simulation for Dynamic Five-Axis NC Verification*, Proc. Autofact, SME, Dearborn, Mich., 1987, PP2-1 to 2-12.
- 41 W P Wang, *Solid Geometric Modeling for Mold Design and Manufacture*, Ph. D Thesis, Cornell University (1984).
- 42 W P Wang, K K Wang, *Real - Time Verification of Multiaxis NC Program with Raster Graphics*, Proceedings of The IEEE International Conference on Robotics and Automation, San Francisco, California, April 7-10, 1986.
- 43 W P Wang, K K Wang, *Geometric Modeling for Swept Volume of Moving Solids*, IEEE Computer Graphics & Applications, Dec. 1986, PP8 - 17.
- 44 J H Oliver, E D Goodman, *Graphical Verification of N/C Milling Programs For Sculptured Surface Parts*, First Symposium on Integrated Intelligent Manufacturing 1986 ASME Winter Annual Meeting, Anaheim, California.
- 45 J H Oliver, E D Goodman, *Direct Dimensional NC Verification*, Computer-Aided Design, Vol. 22, No. 1, Jan/Feb. 1990.
- 46 J H Oliver, *Efficient Intersection of Surface Normals with Milling Tool Swept*

- Volumes for Discrete Three-Axis NC Verification*, ASME Design Automation Conference, Chicago, Illinois, USA, September 16 - 19, 1990.
- 47 T Chappel, *The Use of Vectors to Simulate Material Removed by Numerically Controlled Milling*, Butterworth & Co (Publishers) Ltd., Computer-Aided Design, vol. 15, no.3, May 1983
  - 48 A P Narvekar, J H Oliver, *Intersection of Vectors with General Five-Axis NC Swept Volumes*, ASME Manufacturing International 1990, Atlanta, GA, March 25-28, 1990
  - 49 R O Anderson, *Detecting and Eliminating Collisions in NC machining*, *Computer-Aided Design*, Vol. 10, No. 4, July 1978, PP 231 - 237.
  - 50 R B Jerard, R L Drysdale, K Hauck, *Geometric Simulation of Numerically Controlled Machining*, Proceedings of the ASME International Conference on Computers in Engineering, July. 31 - Aug. 3, 1988, San Francisco, Vol. 2, pp. 126 - 135.
  - 51 R L Drysdale, R B Jerard, Barry Schaudt, Ken Hauck, *Discrete Simulation of NC Machining*, Algorithmica Special Issue on Computational Geometry, 4, (1), 1989, pp.33-60.
  - 52 R L Drysdale, K Ozair, R Jerard, *Discrete Surface Representations for Simulation, Verification, and Generation of Numerical Control Programs*, Proceedings of the 1991 NSF Design and Manufacturing Systems Conference, Austin, Texas, Jan. 9-11, 1991.
  - 53 G W Vickers, K W Guan, *Ball-Mills Versus End-Mills for Curved Surface Machining*, Transactions of the ASME, Vol. 111, February 1989.
  - 54 R Schmid, *An Integrated Turbine Blade Manufacturing System*, Rigid Milling Machine Company, Switzerland.
  - 55 A. R. Weyers, T. J. Stallery, *Basic Machining Reference Hand Book*, 1988
  - 56 K. J. Laviance, *Basic Computer Numerical Control Programming*, Merrilng Publishing Company, 1990
  - 57 T. C. Chang, R. A. Wysk, H. P. Wang, Prentice-Hall, Inc., *Computer-Aided Manufacturing*, 1991
  - 58 R B Jerard, B K Fussell, *The Use of Variational Geometry in the Automatic Generation of NC Machining Programs*, Geometric Modeling for Product Engineering, Elsevier Science Publishers, B.V. North-Holland, 1990
  - 59 W P Wang, *Integration of Solid Geometric Modeling for Computerized Process Planning*, ASME publication, PED-Vol. 19, Nov. 1985, PP177-187.
  - 60 W P Wang, *Application of Solid Modeling to Automate machining Parameters for Complex Parts*, International Seminar on Manufacturing systems, June 1-2, 1987, CIRP 87, Penn. State. USA.

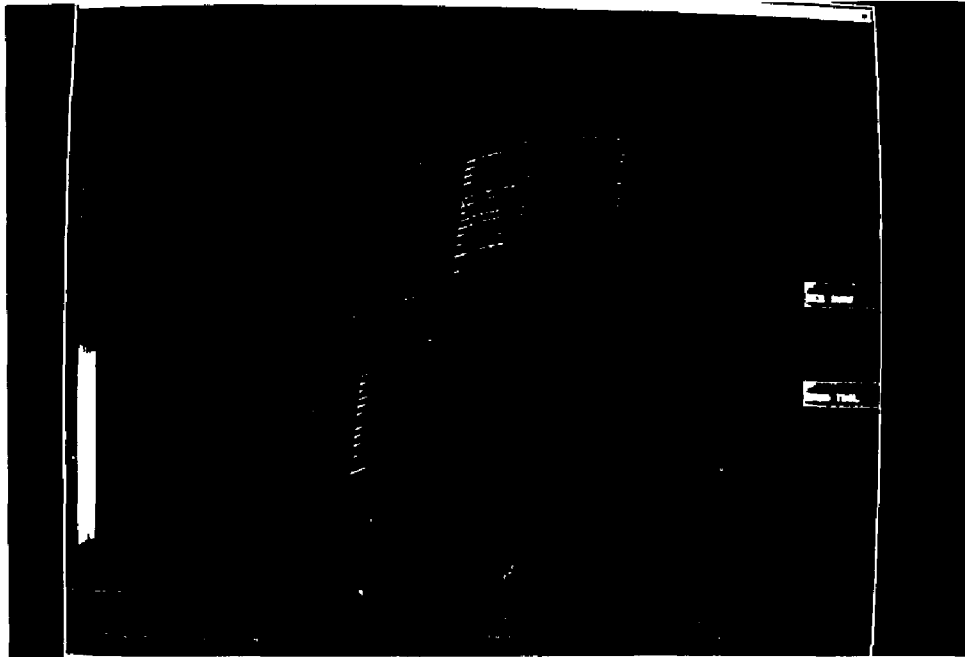
- 61 R. Sedgewick, *Algorithms*, Addison-Wesley Publishing Company, 1988
- 62 B C Kuttner, et. al. *Systematic Processing: An Approach to Fully Automatic NC Tool Path Generation*, ASME Conference, Autofact 1985.
- 63 J D Foley, A V Dam, S K Feiner, J F Hughes, *Computer Graphics*, Addison-Wesley Publishing Company, 1990



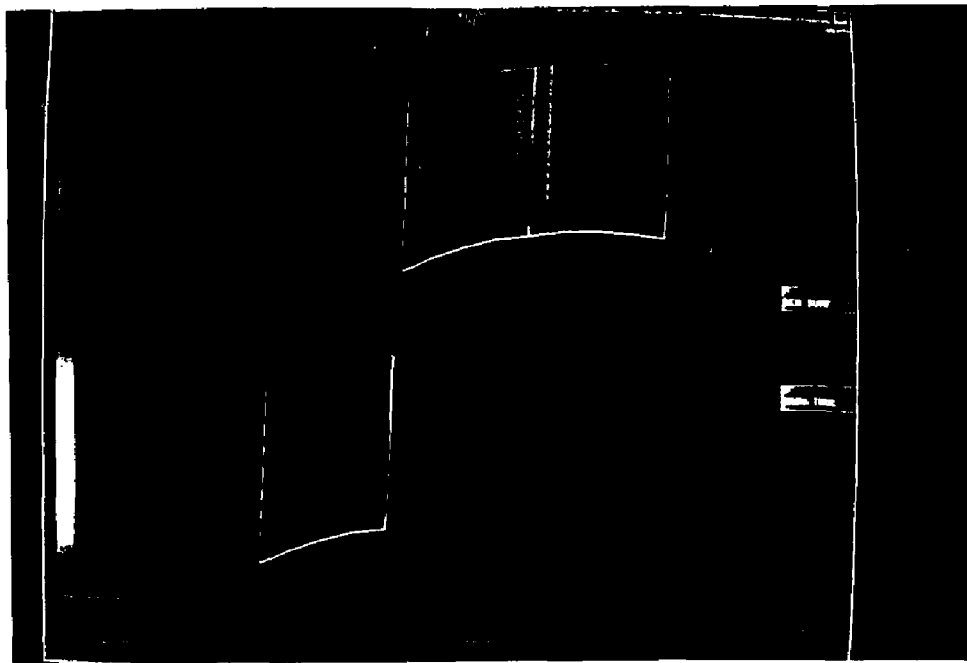
**Plate 1** *Upper left.* Simulation results of the test surface **concave**. Yellow lines are the tool paths superimposed on the image of the machined surface. Machining error is reflected by the color defined by the fringe bar at the left side. Green color indicates the machined surface is within the tolerance, blue indicates excess material (undercut) and red for gouging (overcut). Units are in millimeters. *Upper right.* Test surface **convex** with tool paths superimposed. *Lower right.* Test surface **bumper**. *Lower left.* Test surface **trunk**.



**Plate 2** Tool paths are superimposed on the test surface **zip1**. Green color indicates the machined surface is within the tolerance, blue indicates excess material (undercut) and red for gouging (overcut). Units are in millimeters.



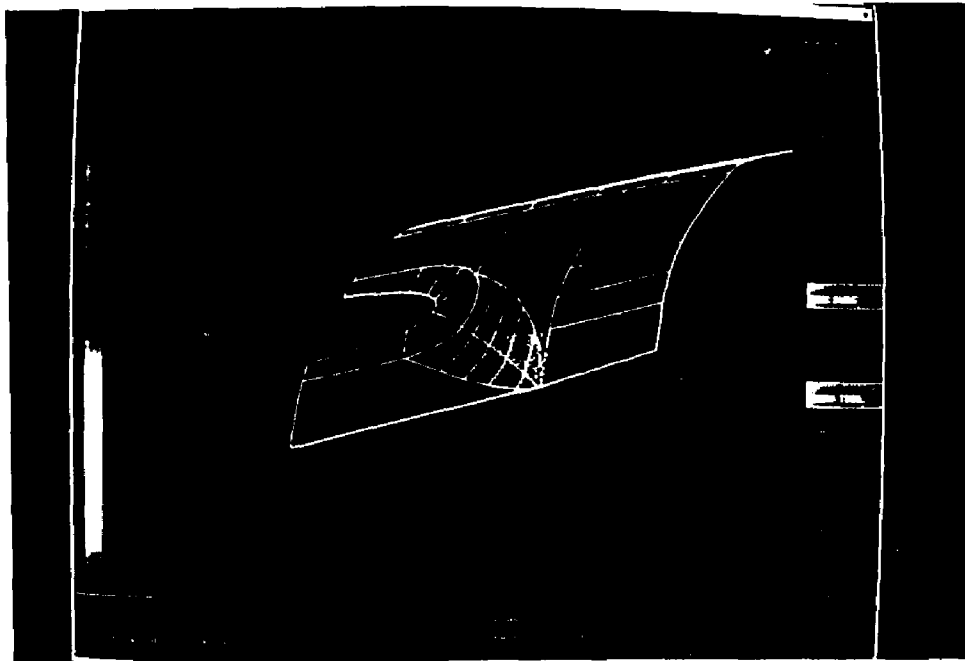
**Plate 3** Simulation of the tool paths generated with a large tool for test surface zip1. Blue indicates that excess material is left. Units are in millimeters.



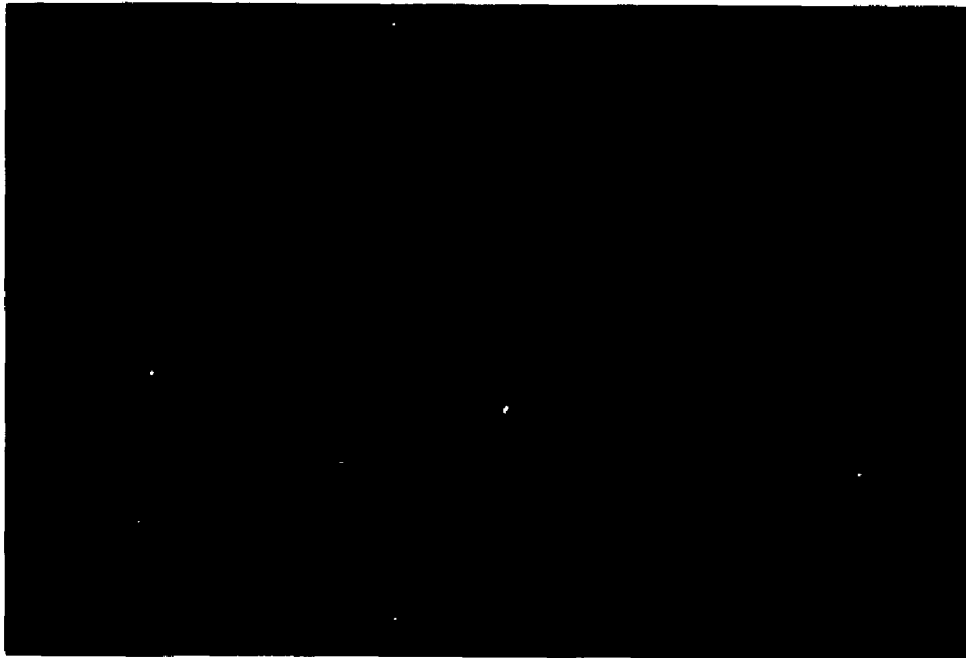
**Plate 4** Simulation of the tool paths generated by FINISH for test surface zip1. Tool paths are generated only over the area with excess material. Units are in millimeters.



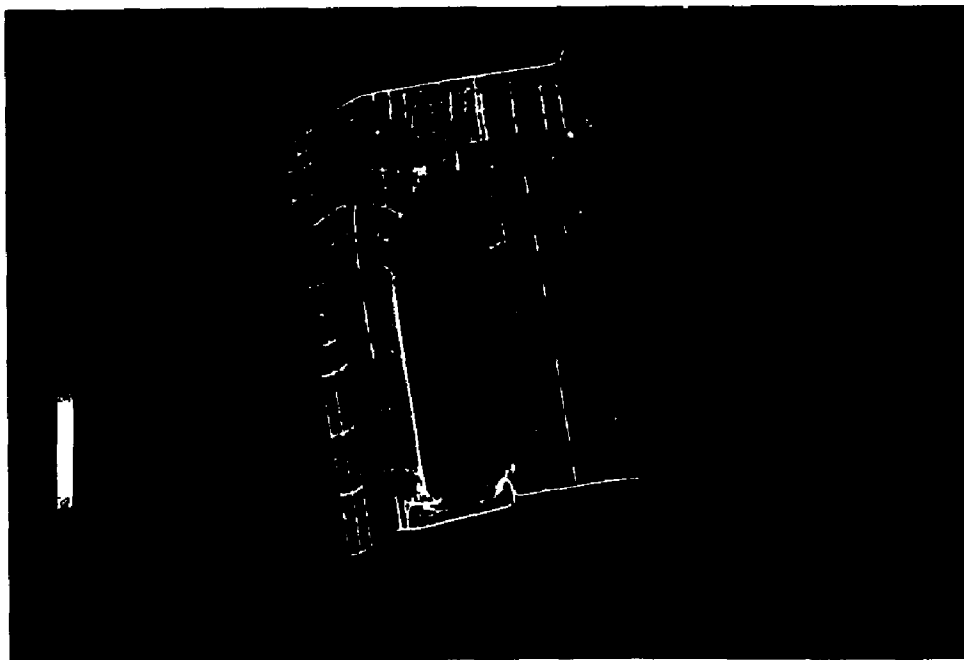
**Plate 5** Simulation of the tool paths generated with a large tool for test surface **z6324r**. Blue indicates excess material. Units are in millimeters.



**Plate 6** Simulation of the tool paths generated by FINISH for test surface **z6324r**. Tool paths are distributed only over the area with excess material.



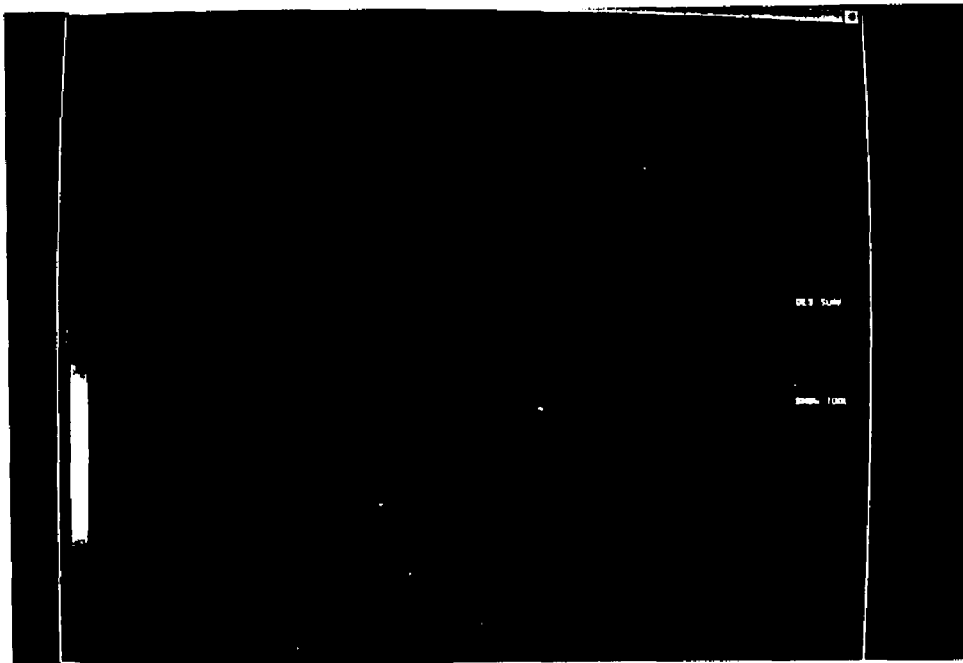
**Plate 7** Tool paths generated with a large tool for test surface **bumper**. Blue indicates excess material. Units are in millimeters.



**Plate 8** Tool paths generated by FINISH for test surface **bumper**. Tool paths are distributed only over the area with excess material.

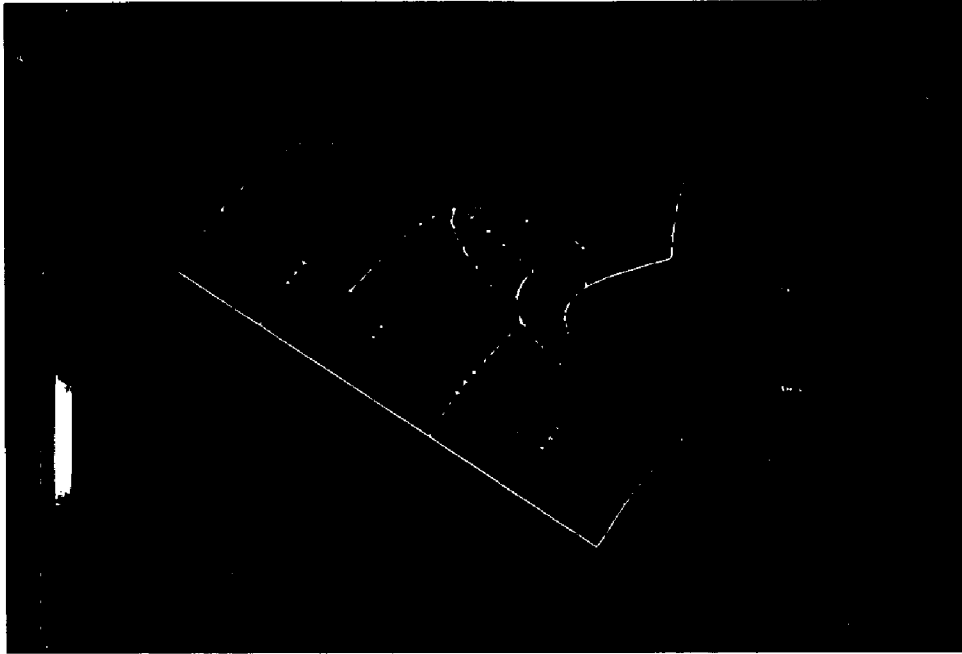


**Plate 9** Single five-axis tool path for test surface **zip5**. Tool axis orientations at individual tool positions are reflected with magenta color. Units are in millimeters.

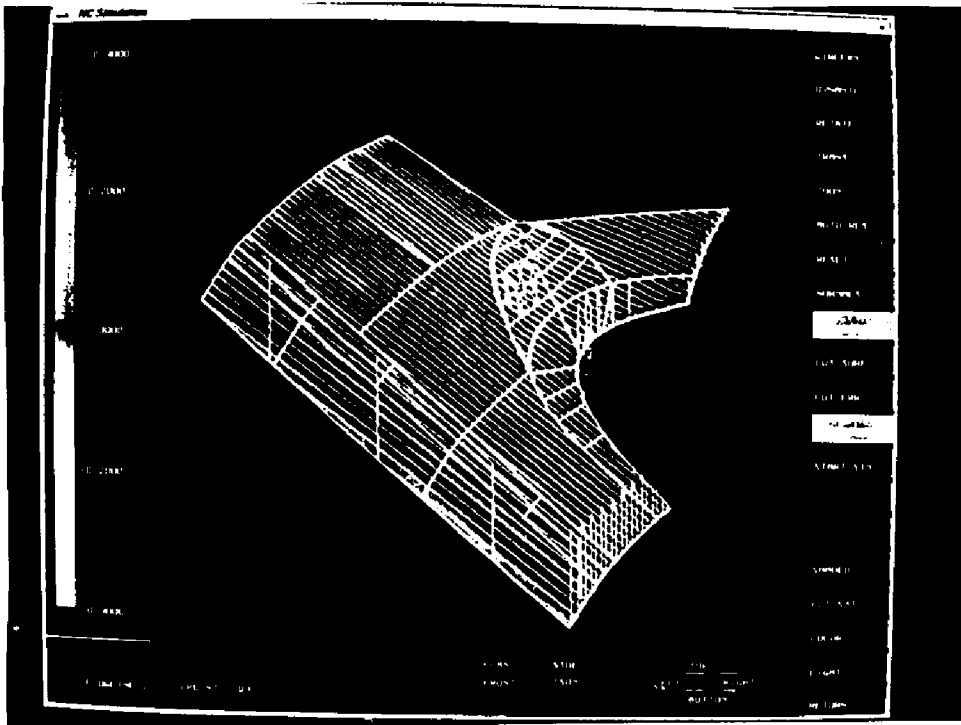


**Plate 10** Five-axis tool paths for test surface **zip5**. Green color indicates the machined surface is within the tolerance, blue indicates excess material and red for gouging.





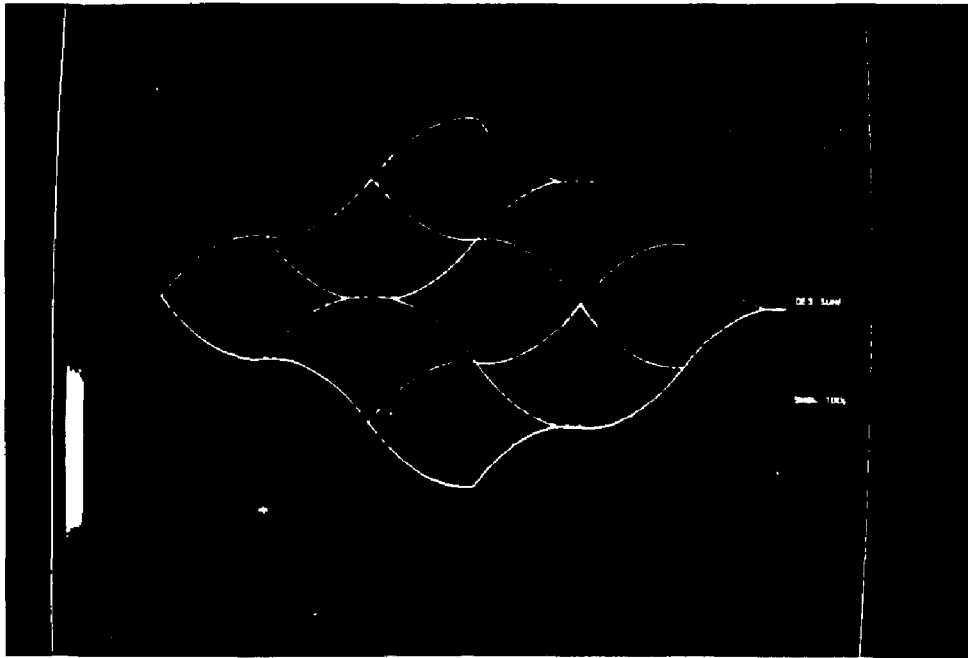
**Plate 11** Single five-axis tool path for test surface **z6324r**. Tool axis orientations at individual tool positions are reflected with magenta color. Units are in millimeters.



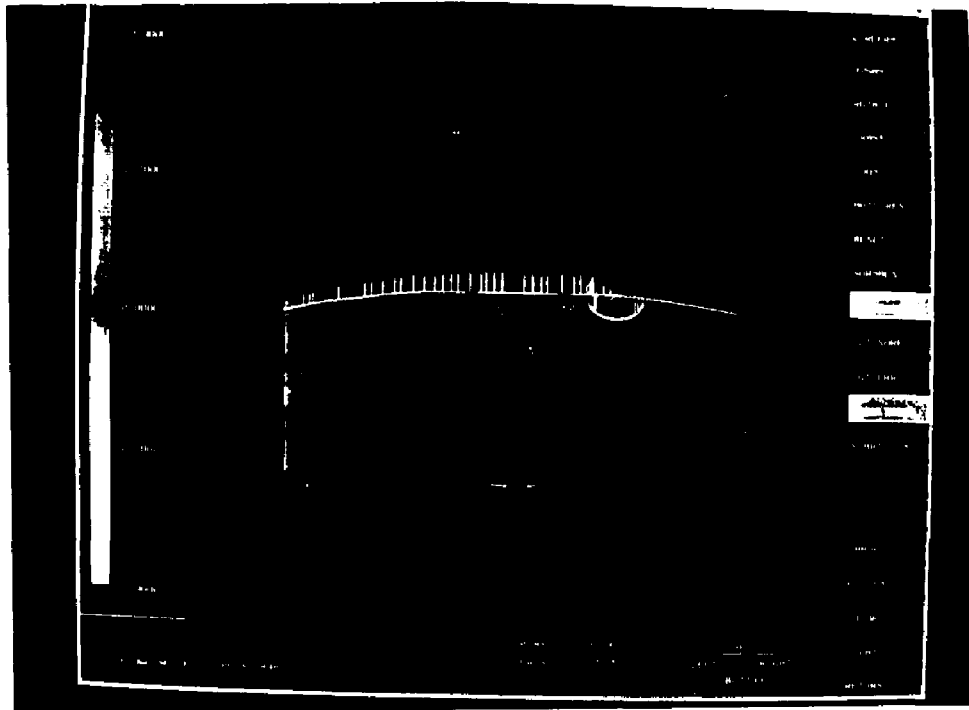
**Plate 12** Five-axis tool paths for test surface **z6324r**.



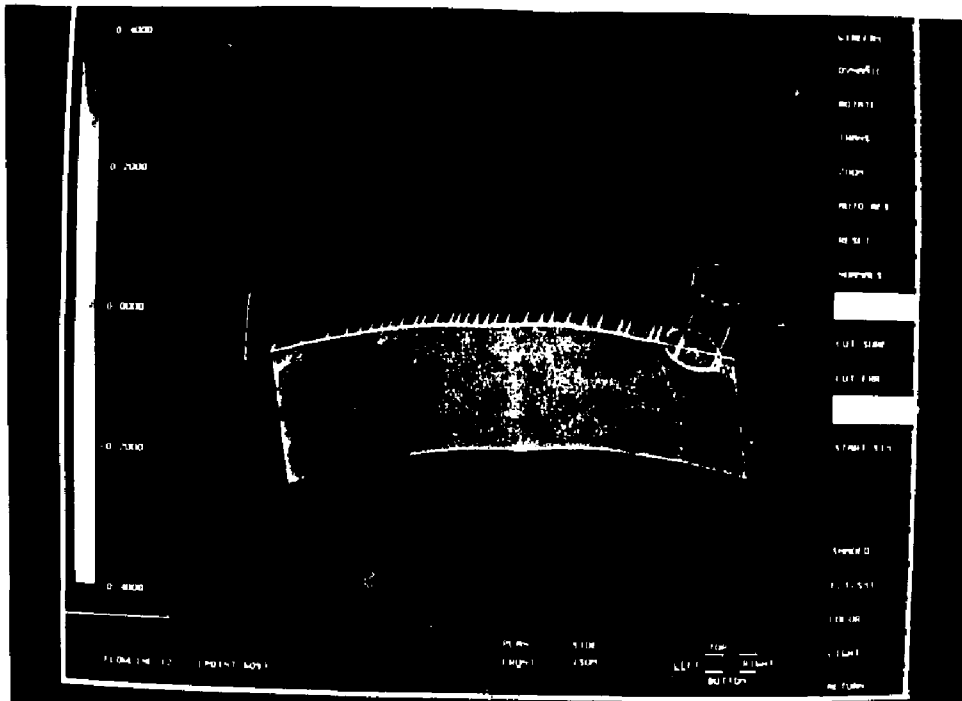
**Plate 13** Single five-axis tool path for test surface **saddle**. Tool axis orientations at individual tool positions are shown with magenta color. Units are in millimeters.



**Plate 14** Five-axis tool paths for test surface **saddle**.



**Plate 15** Partially completed simulation of the fixed-axis tool paths for test surface zip5. Tool paths are distributed over the area with excess material. The tool is parallel to the z axis without tilting. Units are in millimeters.



**Plate 16** Same as Plate 15 except that the tool is tilted  $5^\circ$  from the z axis towards its moving direction.