Automatic 5-axis NC toolpath generation

BY

MAHADEVAN BALASUBRAMANIAM

B. TECH., INDIAN INSTITUTE OF TECHNOLOGY, MADRAS (1997)

S. M., MASSACHUSETTS INSTITUTE OF TECHNOLOGY (1999)

SUBMITTED TO THE DEPARTMENT OF
MECHANICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2001

Author .....
Department of Mechanical Engineering
June 11, 2001

Certified by .....
Sanjay E Sarma
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Prof. Ain A. Sonin
Chairman, Department Committee on Graduate Students

Automatic 5-axis NC toolpath generation

by

Mahadevan Balasubramaniam

Submitted to the Department of Mechanical Engineering
on June 11, 2001, in partial fulfillment of the
Requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

Abstract

Despite over a decade of research, automatic toolpath generation has remained an elusive goal for 5-axis NC machining. This thesis describes the theoretical and practical issues associated with generating collision free five-axis roughing and finishing toolpaths automatically from the CAD model of the part. Instead of decomposing the shape into manufacturing primitives we generate the collision free tool paths directly from the shape of the workpiece by using accessibility arguments. Specific points of discussion in this thesis include:

- A scheme for generating five-axis tool paths using "accessibility" as the driving constraint for tool path generation.
- A scheme for approximating the "accessibility" of a class of tools with visibility because direct evaluation of accessibility information of a tool in a part has been shown to be computationally expensive by several researchers. Visibility of a point along an orientation is a necessary condition for accessibility. We have developed algorithms to compute the visibility information rapidly using graphics hardware.
- A scheme for extracting the most "promising" orientation from the visibility information for the tool in a "global" sense. Algorithms to process the visibility information to select the orientation based on the characteristics of the machining process have been developed.
- A scheme for computing a valid collision free orientation by performing a local search in the neighborhood of the "promising" visible direction using rapid collision avoidance algorithms. Visibility is a necessary but not a sufficient condition for accessibility because visibility cannot account for the geometry of the tool.
- A scheme for determining a smooth interpolation between valid tool postures at the end points. This is not a trivial task because the tool must not interfere with the part while interpolating between valid postures. The visibility cones of the end points are used to compute a "globally" valid interpolatory path and the interpolated postures are then "locally" profiled for access by using rapid collision detection algorithms.

The algorithms outlined in this thesis make it possible, for the first time, to generate 5-axis CNC tool paths automatically from a CAD file. The algorithms have been implemented and tested on several parts.

Thesis committee: **Professor Sanjay E. Sarma**, Chairman, Mechanical Engineering, MIT

Professor David Gossard, Mechanical Engineering, MIT

Professor David Wallace, Mechanical Engineering, MIT

Acknowledgments

The thesis would not have been possible without the advice, encouragement and support of my friend, philosopher and guide, Sanjay Sarma. Sanjay had been very supportive throughout the duration of my research and also helped me to grasp the fundamentals of Computational geometry, CAD/CAM and Computer graphics in a matter of months. Sanjay was very freely accessible for discussion about new ideas in research: I cannot forget the numerous trips to Toscannini and Indian restaurants, where we would have brain-storming sessions to study the feasibility of new ideas. Sanjay also helped me a lot in developing my all round personality. I would like to extend my greatest thanks for him.

I would like to thank Office of Naval Research and Ford Research Laboratories for sponsoring my research. I would also like to thank my thesis committee members Prof. David Gossard and Prof. David Wallace, for their input and direction to the outcome of the thesis.

My association with the Rapid Autonomous Machining Laboratory has been enjoyable both intellectually and personally. I could not have asked for anyone better than Laxmiprasad to do research with. Laxmiprasad laid the framework for this technology and played an important role in the implementation of the technology as well. I would like to thank Stephen Ho and Sriram for developing algorithms that helped me to demonstrate my idea. I am fortunate to interact with my lab-mates Elmer, Taejung, Seung-Kil, Edmund, Niranjan, Yogesh, Marty, Samir, Paula, Ceani, Winston and Stallion.

I could not have gone through all this without the help of ***LORD ALMIGHTY***, my parents and sister's family. Their constant support, prayer, love and encouragement was instrumental in successful completion of the thesis. The note would not be complete without mentioning my friends Harish, Venkatesan, Sreeram, Suvranu, Mahesh, Kripa Kiran and Guru for their constant support during my stay in the U.S.

I thank everyone mentioned above and others whom I may have missed here, for making my stay at M.I.T. pleasant and productive.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1: Introduction

Machining is a broad term to describe removal of material from a workpiece. It covers several processes which can be divided into several categories: cutting, abrasion and non–traditional. Cutting involves use of single–point or multi–point cutting tools with a clearly defined geometry. Abrasive processes involve material removal using small, nonmetallic hard particles having an irregular shape. Non–traditional processes utilize the electrical, chemical, thermal and hydrodynamic methods for material removal. In this thesis, we will use the term machining to describe the milling operation that involves material removal using multipoint cutting tools.

We first describe the importance of machining in today's CAD/CAM in Section 1.1 and the reason for manufacturing industries to increasingly use five axis machines in Section 1.2. Section 1.3 gives an overview of the current state of the art in toolpath generation and reasons out why it is difficult to generate five axis toolpaths that utilize the dexterity of the five axis machine. Inorder to utilize the five axis machines to the fullest extent, we have to look at alternative methods of toolpath generation. With this introduction, we briefly describe our approach in Section 1.4. Section 1.5 outlines the important contributions of the thesis to enable automatic five axis NC toolpath generation.

1.1 Machining in CAD/CAM

The term CAD/CAM implies that an engineer can use computer systems both for designing a product and for controlling manufacturing processes.

Machining is one of the most widely used process to manufacture functional mechanical prototypes of parts. We refer to the initial billet of raw material as the *stock*, the final shape to be machined as the *embedded design*, and the material to be removed as the *delta volume*. The delta volume, ΔV, can be considered to have two portions of interest: the boundary surface, much of which is also the surface of the part, and the internal volume, which constitutes the bulk of the material to be removed. The bulk of the delta volume is removed by a process known as rough machining, or simply, roughing. Roughing is critical to productivity. The surface of the part is usually created by a delicate machining process called finish machining. Finish machining is critical for the functionality of the part. Figures 1.1 (a), (b) and (c) shows the 3D model of a part along with the output at the end of rough and finish machining.

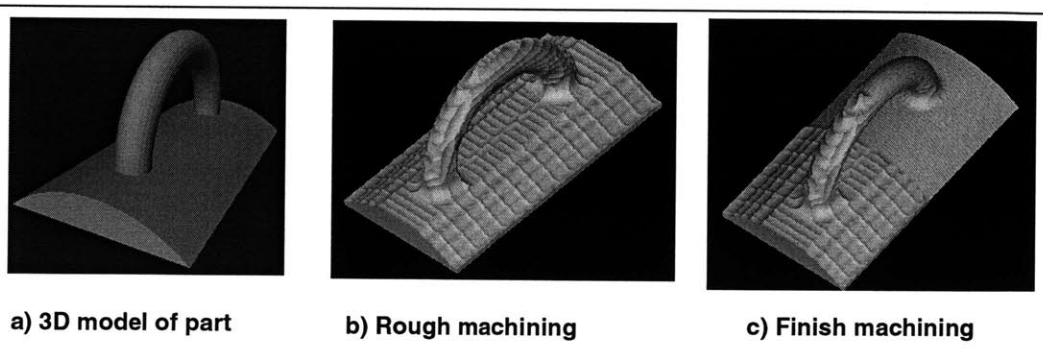


Figure 1.1: Machining stages

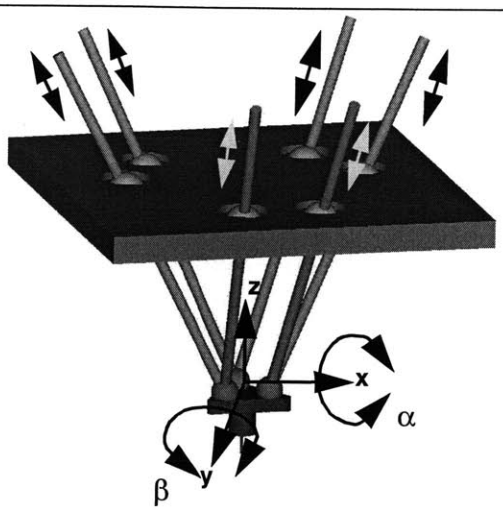


Figure 1.2: Hexapod – Five axis milling

Machining offers many advantages to manufacturing. Firstly, we can achieve closer dimensional tolerances than shaping or metal–forming operations. Secondly, we can manufacture parts with internal and external profiles such as sharp corners and flatness requirements. Thirdly, machining is more economical than manufacturing by other processes, particularly if the number of parts desired is relatively small. However, machining is not usually considered a "rapid" prototyping process because it requires considerable effort and expertise, both intellectual and manual, to plan and operate tools like milling machines and lathes. In recent years this has lead to many attempts to automate machining and integrate it with computer aided design.

1.2 Five axis machines

Manufacturing of parts that have complex contoured regions needs five axis machining capability. Five axis machines have two rotary axes in addition to the standard three translational axes. Figure 1.2 shows a hexapod that has 6 actuators operating on a stuart platform to achieve three translational and two rotational axes at the tip of the cutting tool. Section 1.2.1 gives the reasons for why five axis machines are becoming increasingly popular in manufacturing industries and Section 1.3 summarizes the current state of the art in toolpath generation and Section 1.4 gives the technological challenges associated with generating five axis toolpaths to program the five axis machines.

1.2.1 Need for a five axis machine

In a five axis machine, the tool orientation can be controlled to "access" difficult–to–reach places and machine a surface faster. The ability to reorient the tool relative to the workpiece in mid–cycle enables the machine to mill, drill and tap many different surfaces of a prismatic part, at numerous compound angles, in a single setup. The cuts themselves may demand only X, Y and Z, but extra axes provides the access. Economic considerations that favor five axis machining over a sequence of three axis operations are given below.

1. The radius of curvature may vary for different points on a sculptured surface. Let the mini-

mum radius of curvature of the surface be denoted ρ_{min}. One of the necessary conditions for gouge free contact between a tool and part surface is that the radius of curvature of the tool (ρ_{tool}) must be smaller than the radius of curvature of the surface ($\rho_{surface}$). In three axis machining ρ_{tool} is a constant. Therefore, for three axis machining the tool is sized based on the criterion $\rho_{tool} < \rho_{min}$. Whereas in five axis machining, a flat milling tool of radius (R_{tool}) can be tilted to achieve a wide range of radii of curvature ($0 < \rho_{tool} < R_{tool}$) that matches the surface curvature. Jerard [Jerrard 91] shows that huge performance gains in machining time can be achieved by selecting five axis machining over three axis machining.

2. The ability to perform complete machining in a single setup reduces part handling and increases manufacturing throughput.
3. Cost savings introduced by reduction in the number of machines, tooling and fixturing to achieve the same end result. The increased access can be used to rework on parts and reduce the value of scrap produced.

The potential cost savings must be weighed against the higher initial investment required for five axis machining capabilities. Now that we have seen the reasons for rapid growth of five axis machining in the manufacturing industries, we will look at the key area of generating toolpaths that utilize the "dexterity" of a five axis machine.

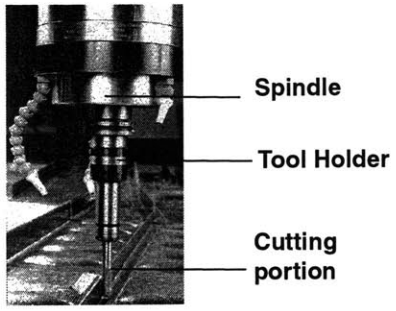
1.3 Five axis toolpath generation — existing approaches

The toolpath generator will generate a sequence of tool positions that can be used to command a milling machine to perform the material removal operation. In Section 1.3.1 we highlight the drawbacks inherent in the state of the art commercial five axis toolpath generators. Section 1.3.2 describes the importance of NC verification for generating collision free toolpaths. We give a brief introduction to the "features" approach in Section 1.3.3 and how it has been used in the context of design and manufacturing. After presenting the limitations of extending the feature based approach to five axis toolpath generation, we stress the fact that one has to look at alternate paradigms of toolpath generation for five axis machining.

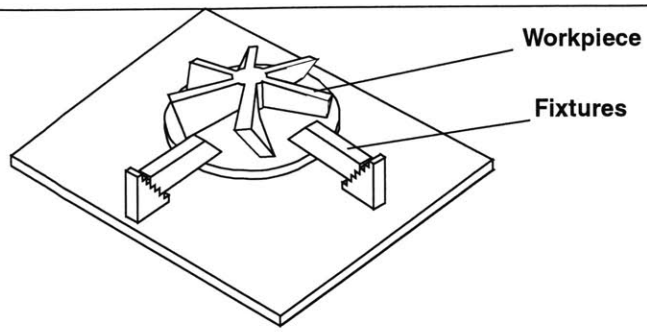
1.3.1 How commercial systems work today

Commercial CAM systems and feature based design systems are based on *3 degree–of–freedom cavity machining techniques*. 3 *dof* machining is an extension of 3–axis machining to complex surfaces. Although, axes may end up moving, the search space is not 5–*dof*. The search is still on a 3–dimensional manifold embedded in a 5–*space*. While many CAM systems like MasterCAM, CAMAX, AlphaCAM and ProManufacture can utilize 5–axis machines, their search space is always limited to three degrees of freedom. The other two degrees of freedom are defined by the orientation mode set by the user. Common modes are surface–normal machining and drive–surface machining. In either case, the problem of path generation is reduced to a search conducted entirely on a three dimensional manifold.

The drawback of this limited search is that the CAM system is incapable of preventing gouges and global interference. That responsibility today lies solely with the user. Furthermore, apart from access issues, the user of commercial CAM systems must also perform additional tasks including: selecting a tool, selecting a cutting strategy, and selecting a cutting order. Operation of commercial CAM systems involves considerable operator skill, which is often difficult to come by. It has been argued that for parts of medium complexity, CAD/CAM may be responsible for up to 20% of



a) A real milling tool [Sutherland]



b) Workpiece along with fixtures

Figure 1.3: Model of tool and workpiece environments

cycle time and a considerably greater fraction of the actual cost [Salmon 96].

1.3.2 NC verification

For the toolpaths to be used in a manufacturing scenario, one has to ensure that there is no collision between any portion of the tool and the workpiece. Figure 1.3 (a) shows a model of the tool [Sutherland]. A tool model can include the cutting portion as well as the non–cutting portions such as shank, tool holder and spindle. The workpiece model shown in Figure 1.3 (b) can also be complex enough to include the effect of fixtures in addition to the 3D model of workpiece. There are several commercial packages such as Vericut, CGCut and NcVerify that simulate and verify four and five axis milling, drilling and turning. In Section 1.3.2.1 we describe the need for NC verification in today's manufacturing environment and Section 1.3.2.2 briefly describes the Z–buffer verification. In Section 1.3.2.3, we describe how the current toolpath generation systems place the onus of collision avoidance on the user under the pretext of NC verification. We conclude this section with the observation that there is a need for a very fast three dimensional collision detection and avoidance algorithm to generate collision free toolpaths.

1.3.2.1 NC verification — needs

In order to generate safe toolpaths for CNC machines it is necessary to check if the toolpaths generated are valid. As the complexity of the machining operation increases, so does the chance for error. NC programming instructions for five axis milling also generally have close tolerance requirements for profiles and surfaces. These factors increase the importance of verifying the NC program. By using NC verification it is possible to identify the following types of errors:

1. Programming inaccuracies due to incorrect toolpath motions and machining the workpiece during the rapid tool motion.
2. Invalid collisions between the tool (cutting portion, tool shank, tool holder and spindle) and workpiece (embedded object and fixtures).
3. Errors in the post–processor output due to bugs in the CAD/CAM program.

1.3.2.2 Z–buffer verification

The most popular approach for verification is the Z–map approach. The Z map model for NC simulation and verification is identical to the Z–buffer approach for hidden surface removal in

computer graphics. The model of the tool, part and the fixtures are loaded into the scene and the tool is swept through the toolpath. The entities are sorted according to the distance from the viewing point and the entities closest to the view point are rendered. The two dimensional projected view is used to make inferences about collision between the tool and the object. This is a passive system where the user has to identify the collisions and tweak the toolpaths to avoid the collisions. Recently, several commercial packages have implemented the verification system using an extended Z–map model. These packages are powerful to compute the work in progress while machining and export the solid model of the machined part. The solid model of the simulated part can be compared with the model of the designed object to determine uncut and gouging locations in the workpiece. It should be noted that the procedure for correction is still passive because the verification package shows the colliding portions of the workpiece in red and the user should manually tweak the G–codes to avoid the collision.

1.3.2.3 How current systems use NC verification

Most of the current toolpath generation systems consider very simple abstractions of the tool and the workpiece. For example, they consider the tool to be an infinitely long, uniform diameter cylinder. This assumption inherently ignores the presence of a larger diameter tool holder and spindle that might possibly come in collision with the workpiece. A workpiece consists of several features and it is imperative to consider the collision of the tool with any feature while generating toolpath for a feature. Moreover, these systems generate toolpaths ignoring global collisions and check for compatibility between local surface geometry and tool curvature to generate local gouge free toolpaths. The task of global collision avoidance is left to the user. Typically, users turn to NC verification systems to visually identify regions of collision. The user must then manually correct the toolpath by tweaking the G–codes or by regenerating the toolpath in the CAM system either with a different drive surface or a different tool tip orientation control curve. This procedure is passive, un–intuitive and iterative. The result is that for complex shapes such as turbine blades, 5–axis toolpath generation can be time–consuming and operator intensive.

1.3.2.4 Collision free toolpaths

An interesting conclusion from Section 1.3.2.3 is the need for an active system that can detect and correct collisions between 3D bodies, namely the tool and workpiece, rapidly. The running time of the collision avoidance routine must be in the order of milli–seconds so that it can be included in the loop of toolpath generation.

The advantages of generating toolpaths using an active system have intrigued several researchers. Li and Lee [Li 94, Lee 95] address this problem and have stressed the importance of a collision detection algorithm in their publications. Li presents an approach for detecting and eliminating collisions between a tool and a triangulated object. Lee presents a two phase approach for checking and eliminating collisions between the tool and the machined surface. The approach assumes a specific representation of the workpiece and is hence limited in applicability. Section 1.4 describes the various stages in our modular approach for determining local and global gouge free five axis tool postures.

1.3.3 The "Features" approach

Features can be viewed upon as information sets that refer to aspects of form or other attributes of a part, in such a way that these sets can be used in reasoning about design, perfor-

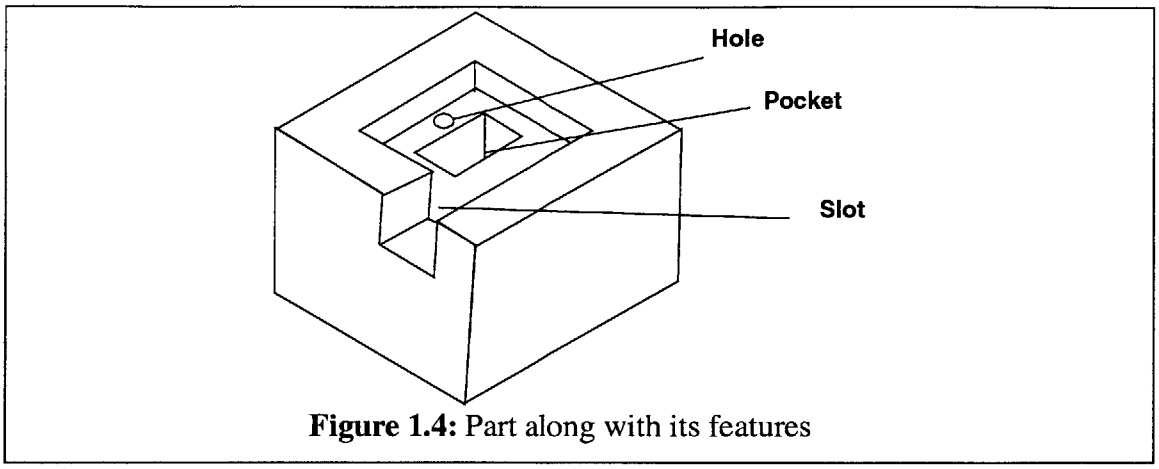


Figure 1.4: Part along with its features

mance and manufacture of the part or the assemblies they constitute. Figure 1.4 shows some examples of features occurring on a part. A product model can be built by using "design" features and is commonly referred to as design by features or feature based modelling. Design features often differ from application "manufacturing" or "machining" features that are required for process planning.

Machining features are 2–1/2 D shape primitives defined in terms of access directions, and mapped to pre–determined, parametrized cutting paths. Typical CAM systems today require input in the form of these features; in turn, they generate low–level cutting instructions by "fleshing out" the details from the parametrized input. Machining features have proved to be convenient because they characterize the capabilities of machining processes such as 3–axis milling and turning fairly well. For example, the important classes of 3–axis cutting operations are end–milling, face–milling and drilling. The machining features that correspond to these operations are pockets, faces and holes respectively.

1.3.3.1 Feature extraction

Designing with "design" features overcomes the problem of lack of modelling freedom in the design with "machinable" features approach. Several commercial CAD systems employ this approach. The problem now shifts focus to the procedure for extracting "machinable" features from the "design" feature. This is referred to by researchers as Feature Extraction [Sakurai 90]. Although this is a simple, obvious statement of fact, there is lot of debate and research going on today to make feature extraction a reality [Salomons 1995]. Appendix D lists the several issues that make feature extraction a very tough problem. Although, there has been a lot of promising research in feature extraction in recent years, no commercially viable solution has yet emerged.

1.3.3.2 Feature based five axis toolpath generator

Features are essentially 2 1/2 D entities that work well in prismatic parts. The full extent of manufacturing capabilities of 4 or 5 axis milling machines cannot be efficiently captured by classical features. Inorder to overcome the limitations, several researchers [Srinivasan 99] are working on the development of five axis feature definitions.

In order to make the machining technology more accessible in today's demanding industrial climate, it is necessary to explore other paradigms which may, in the future, overcome the limita-

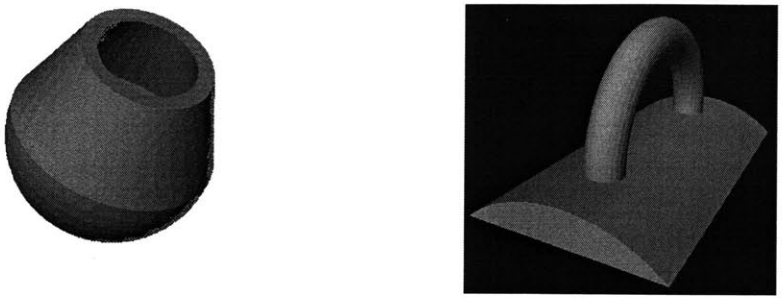


Figure 1.5: Sample parts for demonstrating concept

tions of existing approaches. The problem assumes a lot of interest today because of the rapid increase in the usage of five axis machines in the manufacturing industry and small changes in the design/manufacturing lead times can produce enormous savings in this multi–billion dollar industry.

1.4 How we address the problem

From Section 1.3 we can conclude that it is difficult to generate a "feasible" toolpath quickly from the CAD model of a part. It is mainly because existing systems are extremely time consuming, manual and places the onus of collision free toolpath generation to the user. Inorder to automate the process of toolpath generation, we have to look at alternate paradigms for generating five axis toolpaths by studying the problem from its first principles.

1.4.1 Problem formulation

In this thesis, we outline an alternative way of generating multi–axis machining paths directly from the boundary representation of the geometric object. We refer to this as *generalized 5–axis pocket–milling*. The goal is simple, and is alluded to by several other researchers in the literature. We will attempt to generate *free–form cutting paths* to remove the unnecessary material from the stock while *avoiding local and global interference* with the embedded part shape. Little effort will be devoted to the organization of toolpaths into formal primitives like features. Instead, the goal will be to harness the dexterity of multi–axis machine tools using access arguments.

In order to demonstrate the functioning of the algorithm, we will generate "feasible" five axis toolpaths for a class of parts from their CAD model for which blind folded application of existing toolpath generation methodologies would fail. I will be considering a class of parts whose normal at a point on the surface intersects another feature on the part. Figure 1.5 shows the parts for which toolpaths will be generated to illustrate the algorithms developed in the thesis.

1.4.2 Outline of our approach

Our approach is to generate toolpaths that sweep the delta volume without colliding with the embedded shape of the part. Avoiding collisions can be viewed alternatively as assuring access, and for this reason, our approach can also be referred to as access–based machining. As far as possible, we make no attempt to aggregate or partition the delta volume into pre–determined shape primitives or features. Figure 1.6 shows the procedure for toolpath generation of existing CAM systems and the system proposed by us in this thesis. Our approach for both roughing and finishing

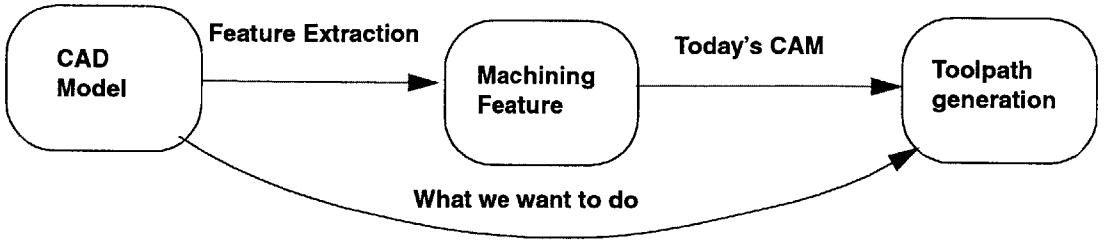


Figure 1.6: Outline of our approach for toolpath generation

consists of three stages: *visibility computation, legal posture definition* and *path interpolation.*

In the first stage, we use the concept of visibility to determine from which directions a point in the delta volume or on the boundary surface is likely to be accessible to an observer located outside the convex envelope of the object. The set of visible directions for a point is referred to as the point visibility cone. The point visibility cone can be used to determine the most "promising" direction to access the point. The procedure for extracting a direction from among the several directions proposed by the visibility cone is different for roughing and finishing. The difference is attributed to the choice of machining strategy selected for roughing and finishing.

Visibility does not ensure accessibility because it cannot account for the diameter of the tool, the tool–holder or the spindle. We describe the second stage of our process referred to as access profiling. We start from the first guess of visibility analysis and determine the size and orientation of the cutting tool that are guaranteed to access a required point in the delta volume without any interference. The major task in posture definition is a local search in the neighborhood of the direction first suggested by visibility analysis. The output of the posture definition step is a set of valid tool postures for every sample point in the delta volume or on the boundary surface.

Finally, all that remains to be done in the third stage is to connect the valid postures into a valid continuous toolpath. This is not a trivial task because the tool must not interfere with the part while interpolating between valid postures. In Chapter 7 we discuss this problem and show how it can be avoided by incorporating the visibility information. Further, we simulate the tool motion along the interpolated toolpath and correct for collisions within the loop of path generation. Figure 1.7 summarizes the three steps in our approach. After covering the theoretical and algorithmic

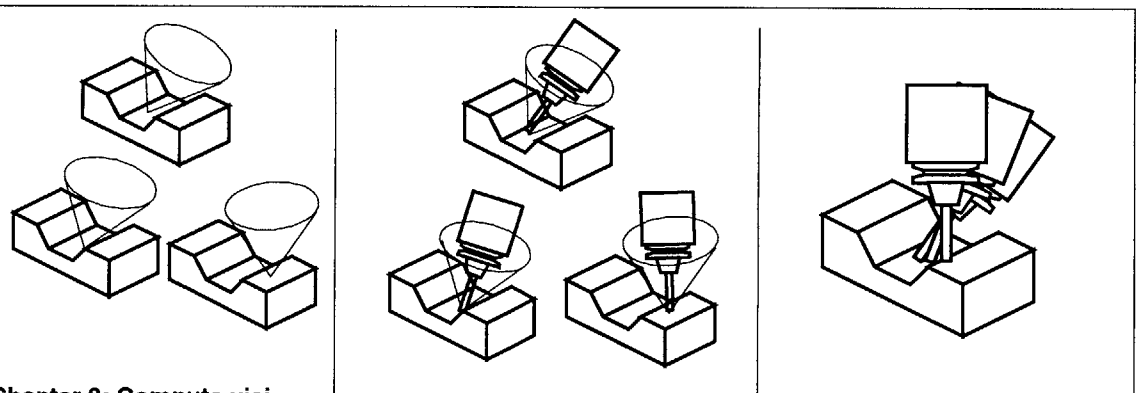


Figure 1.7: Three stages in our approach

bases of our approach in Chapters 3 to 7, we show the application of these techniques to example components in Chapter 8. We conclude the thesis with a summary of the research done along with the future work in Chapter 9.

Depending on the information needed for an application, the users can add constraints to this modular five axis toolpath generation methodology or terminate the procedure after determining the access postures in the second stage. For example, Taejung [Taejung 01] proposes a method for determining the toolpath optimal in the machine actuator space based on tool access postures at various points on the surface.

1.5 Contributions

In this thesis we present several algorithms and heuristics to generate roughing and finishing toolpaths by using "accessibility" of the tool as a driving constraint for the problem. In this Section we will summarize our major contributions.

1. We have developed a framework for generating five axis toolpaths, start to finish, completely automatically from the CAD model of a part. In order to develop the framework we used some of the interesting concepts developed by researchers in the field. However, these concepts served as "islands" that addressed only a part of the larger problem. We have built the "bridges" across the "islands" and in the process invented new concepts and fast algorithms that enable automatic five axis toolpath generation.
2. We have invented an overall scheme based on visibility in place of accessibility. The reason being accessibility information is difficult and time consuming to generate. The premise for our approach is that for non–overhung tools (Appendix A) *accessibility is a subset of visibility.* A new method for computing "dense" visibility information rapidly using graphics hardware has been presented in Chapter 3. This is a practical approach to C–Space planning [Lozano-Perez 83].
3. We have developed new algorithms that process copious amounts of visibility information to generate five axis tool posture assignments for a set of cutter contact points. The driving force behind these algorithms is that visibility closely matches accessibility and can be used to arrive at a "promising" access direction. Algorithms to select the "promising" access directions for roughing and finishing have been presented in Chapter 5 and 6 respectively.
4. Visibility assumes a ray of light to reach a point but accessibility is for a tool of finite geometry. We can expect collisions to occur between the workpiece and the tool positioned along the "promising" orientation. We have incorporated methods from computational geometry to process the results of a collision detection algorithm to move the tool away from collision. The algorithms to convert the "promising" access direction to a "valid" access direction have been presented in Chapter 4.

The algorithms outlined in this thesis make it possible, for the first time, to generate 5–axis CNC toolpaths automatically from a CAD file. The algorithms have been implemented and tested on several parts.

Chapter 2: Background

In this chapter, we summarize the evolution of the research in the areas of access based toolpath generation, collision avoidance, NC verification and feature based machining. We conclude the chapter by presenting a short note on how the issues mentioned above are handled in commercial CAD/CAM software.

2.1 Review of research on Access based approaches

The problem of tool access has been approached from both, a solids perspective, and a surface perspective. Seminal work in the area of visibility and visibility maps was performed by Chen [Chen 92] and Woo [Woo 94]. They introduced the concept of visibility cones for points on a workpiece, which can be mapped on to the unit sphere to create a Gaussian Map. The same authors also show how the Gaussian projection can be extended with a central projection to manipulate access information and minimize setups. The idea of Gaussian maps was used by Wuerger and Gadh [Wuerger 95] to evaluate the separability of dies. The concept of access is also handled in a feature–based approach in [Sarma 96]. The ideas of a visibility cone have influenced surface machining as well. Lee [Lee 95] uses a convex hull based approach to approximate local visibility. An innovative approach to surface accessibility is presented in [Elber 94], in which convex surfaces are mapped to a space in which they become planar. Obstacles to the surface are also mapped into this space, and toolpath generation is carried out in a 3D world. The paths are then inverse mapped back to the original space to obtain 5–axis toolpaths. [Spyridi 90, Henderson 96, Tangelder 96, Gaines 97, Stage 97] explore other concepts pertaining to accessibility.

In recent years, the problem of determining tool accessibility has been approached from the perspective of computer graphics. Spitz [Spitz 00] presents a very interesting and a practical method for performing accessibility analysis using computer graphics hardware. The emphasis of their work is on computing accessible directions for tactile probes used in 3–D digitization with coordinate measuring machines. [Morishige 99] presents an approach for toolpath generation using three dimensional configuration space. The publication highlights the importance of three–dimensional configuration space for determining the optimal cutter location data and proposes an algorithm to determine the three dimensional configuration space.

2.2 Review of research on Collision avoidance schemes

Collision detection procedures determine whether or not two entities are in a state of contact, and are useful in a range of computational geometry applications. A range of collision detection procedures are known today for representations such as convex polytopes and triangulated surface meshes. In addition to determining the boundary of collision of two shells, we need the depth information for computing the force or displacement necessary to perform haptic rendering or collision avoidance during toolpath generation. [Rameau 93] refers to this as penetration analysis.

Lin and Canny present an efficient method of calculating collisions between convex polyhedra [Lin 93]. The method takes advantage of temporal and spatial coherence and operates especially well when the objects rotate at a speed less than one half a rotation per collision detection cycle. In addition to determining a collision event, the method gives proximity information. Application of

the Lin–Canny method is limited to convex objects. Methods to break non–convex objects into a union of convex object could be used, but the additional complexity of these methods limits the practicality of such an approach. I–Collide is a collision detection package written by the computational geometry group at University of North Carolina. I–Collide extends the Lin–Canny approach to an n–body simulation with penetration depth information; however, the objects of the collision detection are limited to convex polyhedra [Cohen 95], and therefore suffer the same limitations of the Lin–Canny method. Mirtich has improved upon the Lin–Canny approach with the V–Clip collision detection method, which enhances the robustness of the Lin–Canny approach and adds penetration depth calculation [Mirtich 98]. Again, the technique deals with convex objects so that non–convex objects must first be decomposed into the union of convex objects.

Triangle set methods represent the surface of each object as a set of triangles and report whether or not any triangle of one set intersects any triangle of the other set in each collision detection query. Triangle sets are capable of representing both convex and non–convex objects. Furthermore, the methodologies do not even require that the object be a coherent solid; any set of triangles is applicable. To increase the efficiency of the triangulated collision detection methods, a bounding volume hierarchy is used. Gottschalk *et al.* introduce the use of Oriented Bounding Boxes in triangle set methods [Gottschalk 96], and Held *et al.* propose the use of k–Discrete Oriented Polytopes [Held 96]. These bounding volumes and their associated bounding volume hierarchical trees enable the collision detection technique to compute the solution more quickly in the average case.

Ho *et. al.* [Ho 1999] presents an interesting approach for collision detection using a heterogeneous representation consisting of an ensemble of implicit equations on the probe side and a cloud of points on the part side. The authors also present a rationale as to why this system has fast performance and how depth information is yielded naturally.

2.3 Review of research on NC simulation and error detection

Simulation of NC machining requires algorithms to model the embedded part, workpiece and the swept volume of the tool. The workpiece model need to be updated dynamically by subtracting the swept volume of the tool. Moreover, in addition to the algorithms being fast the model must be accurate to within several order of magnitude of the desired nominal tolerance of the machined part. The problem of simulation has been approached from a solid modelling perspective as well as a graphics perspective with true solid update facility.

A mathematical approach to simulation based on set theory was first presented by Gossard [Gossard 78]. The feasibility of application of Constructive Solid Geometry to simulation was studied by Voelcker [Voelcker 81] and Fridshal [Fridshal 82]. The solid modelling approach is computationally expensive and requires $O(n^4)$ calculations where n is the number of tool positions to be verified. Each tool position involved computation of intersection between the toolpath envelope and workpiece. In order to decrease the computation time, localization techniques and local updating methods were studied by Woodwark [Woodwark 84] and Bronsvoort [Bronsvoort 89]. Choi [Choi 98] observes that the main problem with the solid modelling approach is intensive nature of verifying the equivalence of the finished workpiece and the desired part.

The Z map model for NC simulation and verification is identical to the Z buffer approach for hidden surface removal in computer graphics. This concept was first introduced by Anderson [Anderson 78] and several researchers have provided for different implementations. Van Hook

[Van Hook 86] developed another version of an "extended Z map" using depth elements called dexels. Most early systems for simulation were three axis based, or were relatively limited in their applicability of predicting gouging in five axis machining. Recognizing this problem, a few researchers in recent years have looked into the simulation of multi–axis cutting: [Wang 86, Oliver 86, Jerrard 89b, Jerrard 91]. Recently, there have been several commercial packages (CGCut, VeriCut and NC Verify) that can perform NC verification using Z maps. These packages can identify the amount of undercut or uncut material in the workpiece and some even export the solid model of the intermediate solid.

2.4 Commercial CAM systems

Most commercial CAM systems, including purported 5–axis systems, are based on *3 degree–of–freedom* (as opposed to 3 axis) *cavity machining techniques*. We use this term because, while many CAM systems like MasterCAM, CAMAX, AlphaCAM and ProManufacture can utilize 5–axis machines, their search space is always limited to three degrees of freedom. The other two degrees of freedom are defined by the orientation mode set by the user. Common modes are surface normal machining and drive surface machining. In either case, the problem of path generation is reduced to a search conducted entirely on a three dimensional manifold. The drawback of this limited search is that the CAM system is incapable of preventing gouges and global interference. That responsibility today lies solely with the user. Furthermore, apart from access issues, the user of commercial CAM systems must also perform additional tasks including: selecting a tool, selecting a cutting strategy, and selecting a cutting order. As a result, 5–axis CAM machining is still very much an acquired skill today.

Growing awareness of these problems has lead to recent interest in a new technology called **Generative NC**. SDRC has recently offered an early version of its Generative NC package. SDRC's generative NC system, *however, is still based on 3–axis machining*, and still requires human input for access–direction selection and tool selection. This proposal deals with the theoretical and practical issues in 5–axis generative NC. There are fundamental theoretical issues that need to addressed before such a system can be created. Yet, without such research, it will be difficult to make full and efficient use of advanced 4–axis, 5–axis and multi–axis machine tools like the Hexapod.

2.5 Robotics

The research that we present here has some parallels to previous work in robot path planning as well. The problems of visibility and accessibility have been addressed in great detail by a number of researchers. Lozano–Perez first attempted to build automatic robot navigation systems [Lozano-Perez 83]. The concept of a configuration space evolved through a series of papers in the early 80's [Udupa 77, Lozano-Perez 81, Lozano-Perez 83]. In the latter paper, Lozano–Perez also introduced the concept of cell decomposition, which is loosely analogous to the voxelized approach presented here. A comprehensive description of later developments in robot planning is presented in [Latombe 91]. An important difference between robotics and machining, however, is that while robotic path planning is concerned with accessing particular points in the configuration space, machining is concerned with sweeping all the points within and on the boundary of the delta volume.

2.6 Review of research on Feature based machining

Although this thesis does not build on feature research, we present a survey of research on features for completeness.

The concept of machining features has been an important step in the development and understanding of manufacturing planning. Machining features have the following advantages: 1) features are a convenient decomposition of machining into handleable units for high level planning; 2) toolpath generating algorithms can be developed and implemented up–front; 3) since features fit the object oriented model well, tool selection and cutting parameter selection can be linked cleanly to knowledge bases; 4) machining features implicitly define access directions and accessibility volumes.

The first mention of features is probably by Krypianou [Krypianou 80]. The concept of manufacturing features first appears in [Arbab 82]. Arbab points out the similarity between the boolean difference operation in constructive solid geometry and the material removal in machining. This lead to the idea of destructive solid geometry (DSG), a design input methodology later refined in a series of papers: [Hummel 86, Kramer 88, Turner 88, Cutkosky 88, Shah 88 and Gindy 89]. In DSG, the user defines a "stock" and *subtracts* primitives (features) to define the part. The development of process planning systems for machining has closely followed the development of features technology. Beginning with early work by Nau [Nau 86], Hayes [Hayes 89], Anderson [Anderson 90] and Cutkosky [Cutkosky 90], to more recent papers by [Yut 95, Gupta 95 and Sarma 96], the use of features has become better understood and more widespread.

Meanwhile, there has been interesting research in feature extraction in recent years. Seminal work on feature recognition was done by Woo [Woo 82]. Later, Joshi [Joshi 88] used graph based heuristics to extract features from adjacency graphs. [Dong 88, Sakurai 90, Finger 90 and Vandenbrande 90] made important contributions to the field. Kim extended Woo's work on convex decomposition [Kim 90]. Gadh introduced the concept of depth filters for feature recognition [Gadh 92]. Nau *et. al.* introduced the idea of generating alternative, optimal machining volumes in [Nau 92]. Recently, Regli has reported a promising new approach to feature extraction in his Ph. D. Dissertation [Regli 95]. His approach is based on the extrapolation of "maximum cover features" for 3–axis machining from the faces of a boundary representation. In general, most feature based approaches have been limited to 3–axis machining.

Features are essentially 2 1/2 D entities that work well in prismatic parts. The full extent of manufacturing capabilities of 4 or 5 axis milling machines cannot be efficiently captured by classical features. Inorder to overcome the limitations, several researchers [Srinivasan 99] are working on the development of five axis feature definitions.

Chapter 3: Visibility Analysis

A key idea in delta volume machining is collision avoidance between the tool and the workpiece. Collision avoidance can be viewed alternatively as access assurance. In other words, how can the cutting tool be oriented to access the delta volume without intersecting with the embedded design? An interference free tool direction will be referred to as an access direction. For a given point in the delta volume of the object, the access direction may not be unique. We therefore talk of access cones, which are contiguous sets of access directions. Section 3.1 describes the computational complexity associated with the procedure for determining the access information of points in the delta volume. We then propose a method of computing the access information in stages by using the fact that visibility is a necessary condition for accessibility for most tools (described in Appendix A). We can therefore also talk about visibility directions and visibility cones, which are conservative approximations of accessibility directions and accessibility cones respectively because visibility ensures that an infinitely long tool of zero diameter can access the region of the delta volume. Section 3.2 describes the evolution of the visibility concept for 3–axis applications and difficulty in extending the results for 5–axis applications. Section 3.3 briefly describes the complexity and time consuming nature of a software based approach to determining visibility. We then explain our procedure for rapidly determining the 5–axis visibility information using graphics hardware in Section 3.4. In Sections 3.5 and 3.6 we address the various implementation issues such as selection of number of viewing directions, relation between the size of the part and the view port and an order of magnitude estimate about the memory requirement for generating the visibility information.

3.1 Difficulty of access determination problem

For the purpose of machining *accessibility* is defined as follows: A point P on surface S is accessible by a tool oriented along orientation O, if and only if the cutting portion of the tool is in contact with point P and the tool is not interfering with the embedded design object. The orientation from which a point can be accessed by a tool can be mapped to a point on the surface of a sphere (S^2) centered at the origin. Maps of orientation vectors to S^2 are commonly referred to as Gauss Maps. For a point in R^3 there are infinitely many orientations from which it could be accessed by a tool. Clusters of visibility or accessibility directions become patches on the surface of the Gauss Sphere. The patches on the Gauss map forms a cone with its apex at the origin and we refer to them as visibility and accessibility cones respectively. As we have defined them, these cones capture the directions of interference free visibility or accessibility for a point in three dimensional space, and we will refer to such cones as *point–visibility* or *point–access* cones. While generating toolpaths, the tool should be given an orientation along which it should be aligned at every point. In order to generate interference free toolpaths, this orientation should be one of the accessible directions for that point. Therefore determining point access cones is of real importance in automating the toolpath generation for machining.

Figure 3.1 (a) shows the point access cone for a point in R^3 space. The point access cone is the entire sphere because the point can be freely accessed from all the orientations. Figure 3.1 (b)

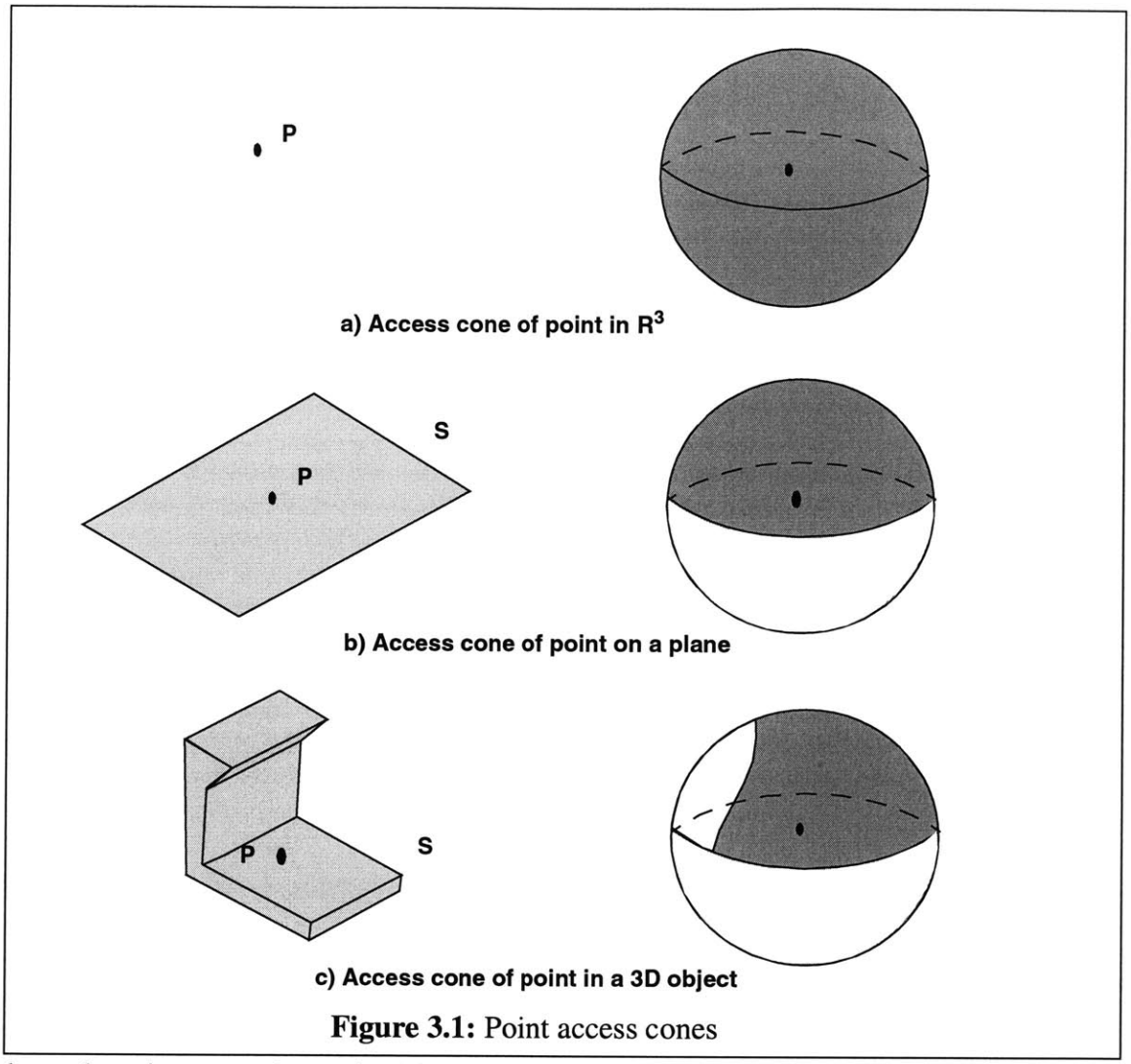


Figure 3.1: Point access cones

shows the point access for a point lying on a plane. The point access cone is the upper hemisphere formed by slicing the sphere with the plane because the point can be reached from along the orientations lying above the plane. Figure 3.1 (c) shows the point access cone for a point lying in an arbitrary object. In the subsequent chapters, we explain the importance of access cones in assigning tool postures for a set of consecutive cutter contact points.

3.1.1 Point and region access cones

The access direction can be determined by using a trial and error approach by positioning the tool along random orientations to reach a point. It is evident that this approach is very time consuming and not reliable. The access cone for the point can be determined as an union of valid access directions. [Tangelder 96 & Roberts 96] use the Minkowski operation to generate accessibility cones. Unfortunately, Minkowski methods tend to be computationally expensive. Our approach is to find the approximate access direction, but a good estimate, very efficiently.

Access can also be generalized for regions. An access cone for a *region*, which we will refer to as a *region–access cone*, is the intersection of the point–access cones for all the points in the region. The access cone for a region R can be defined formally in terms of point access cones as

follows:

$$A(R) = \bigcap_{p \in R} A(p)$$

If we consider a set of points or regions P in cartesian three dimensional space, then the map of access cones for points or regions in P, $P \rightarrow S^2$, is referred to as an *access map.* Clearly, if the access map were available for some portion of the delta volume, it would be an invaluable aid in determining a toolpath. Unfortunately, the access map is very difficult to compute for even a single point, and progressively becomes more prohibitive as the size of the set P is increased.

Visibility is analogous to accessibility. Infact it is a special case of accessibility where the tool can be assumed to be a straight line. Visibility is easier to compute, especially in initial stages when the tool has not been selected. For this reason, like several previous researchers, we will use visibility analysis as a first step in our computation of access.

3.2 Visibility — background

In the past, several research groups such as those lead by Woo, Requicha, Gadh, Henderson and Roberts have investigated the use of visibility maps for path generation in machining. However, because of the difficulties associated with computing and manipulating visibility information, most previous research has been limited to 3–axis applications such as 3–axis machining, 3–axis probing and $2\frac{1}{2}$ D mold parting. Section 3.2.1 summarizes their approach to determine visibility cones for 3–axis applications. The 3–axis situation excludes the possibility of continuous reorientation of the tool in the fourth and fifth axes. Section 3.2.2 presents the difficulty in generating and using the visibility map for 5–axis applications.

3.2.1 Visibility in a 3–axis application

Consider 3–axis machining on a vertical mill. All the machining must be carried out from a small set of discrete orientations that correspond to the fixturing setups. This is because the tool cannot be rotated, and the number of setups must, from a practical stand point, be kept small. To avoid poor finish from fixturing errors, each continuous face of the component *must* be machined in a single setup. Single setup machining implies *that an entire flat face must be treated as if it were a single entity.* This simplifies the visibility map $V{:}P \rightarrow S^2$ substantially, because the set P is now a discrete set of surfaces rather than a continuum of points. It is possible to use a region visibility cone rather than a point visibility cone. The component shown in Figure 3.2 has precisely ten flat faces, and consequently, the visibility information for the entire component, as pertinent to 3–axis machining, can be captured with ten visibility cones. Perhaps more significantly, Wuerger *et. al.* make an observation that can be used to approximate the 3–axis access cone of a flat face on simple components [Wuerger 95]. The basic idea is as follows. The access cone for a single flat face unobstructed by other entities in space is a hemisphere oriented along the normal to the plane of the face, and pointed towards the outside of the solid object. If a face has neighbors, then its access may be obstructed depending on the orientations of these neighbors. If the neighboring faces are also planar, then the obstructed access cone of the face in question is often the intersection of the access cones of that of its neighbors. This is illustrated in Figure 3.2.

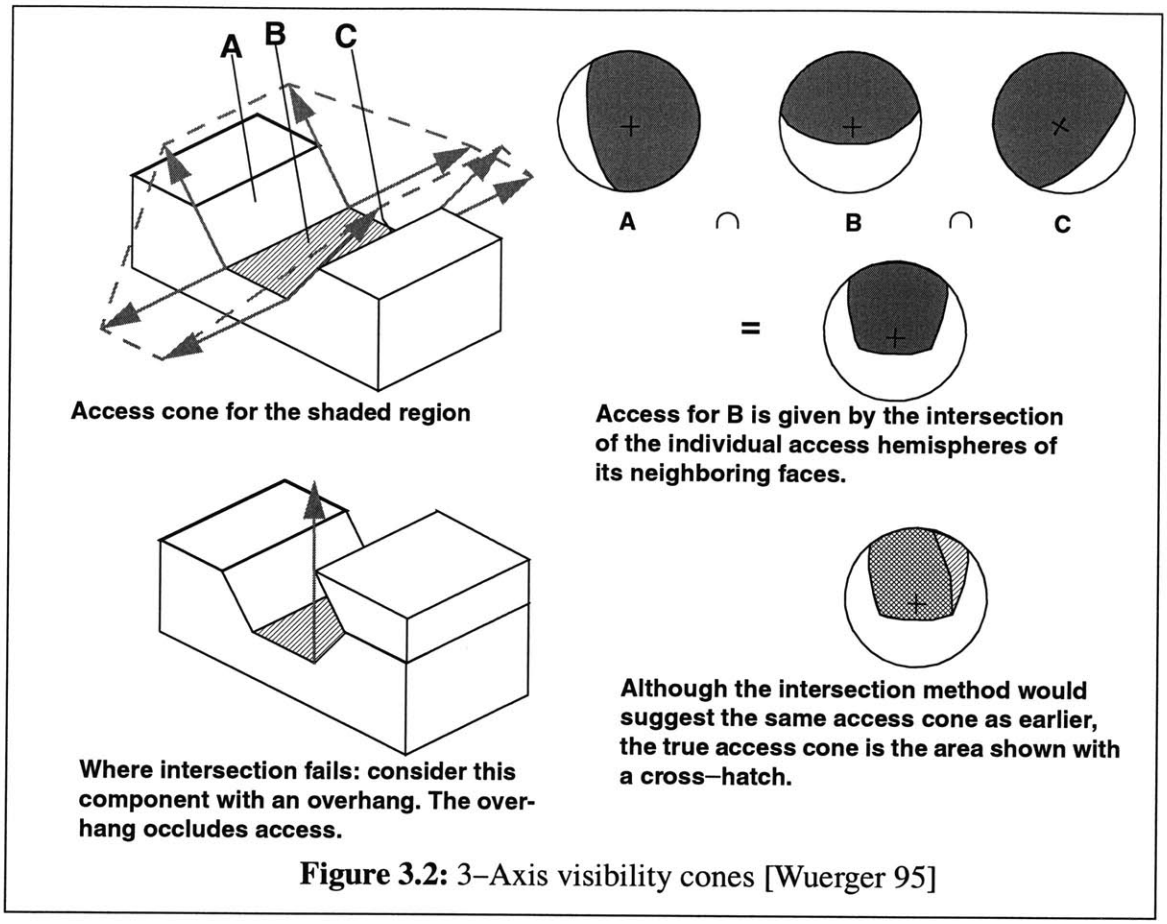


Figure 3.2: 3–Axis visibility cones [Wuerger 95]

Although this method works for simpler shapes, the computation does not consider obstructions caused by other faces that do not actually neighbor the face in question, and therefore it is neither correct nor complete in general. The intersection method is therefore not acceptable for generative path planning in the 5–axis case.

3.2.2 Issues in generating 5–axis visibility cones

There is a fundamental difference between the 3–axis space R^3 and 5–axis space $R^3 \times S^2$, and it is in the latter area that the major contribution of this work lies. The consideration of the rotational axes provided by 5–axis machining complicates the determination of an access map. In 5–axis machining, the search space is greatly expanded by the fact the tool can have an infinite number of orientations at every point in the toolpath, and picking the one that works can be computationally intractable. Therefore the thinking from 3–axis machining *cannot* be generalized to 5–axis situations. The major reasons for this are detailed below.

Firstly, in 5–axis machining the access cone for every point in the delta volume must be computed independently to generate an effective toolpath. This is because continuous fourth and fifth axis control make it possible to vary position and orientation of the tool continually if necessary, even within a single setup. Unlike in 3–axis machining, surfaces need not (and should not in general) be abstracted as integral entities from the point of view of access, and it is not sufficient to compute region access cones for an entire face. If the desired component can be machined with the

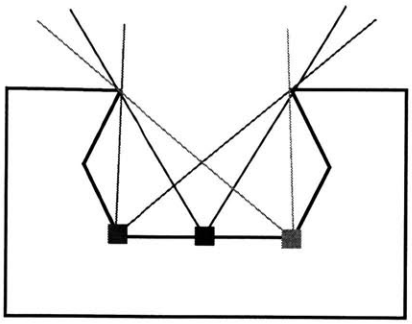
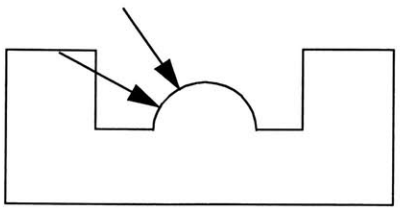


Figure 3.3: 5–Axis visibility cones vary along a single face

access direction kept constant through out a cut, then the machining should probably have been performed on a 3–axis tool, because the continuous control of the fourth and fifth axes haven't been used. This is unlikely to be the case in parts deemed suitable for 5–axis machining. Figure 3.3 shows the 5–axis point visibility cones for several points on the surface of the object. From the figure it is clear that the 5–axis point visibility cones varies along the face of the object.

How do commercial packages deal with this problem? Commercial packages such as SDRC/ CAMAX, MasterCAM do use the continuous fourth and fifth axes. They address the question of tool orientation with a simple adhoc strategy known as surface normal machining: they simply bind the tool orientation normal to the surface. From the point of view of access, this naive strategy often works because the normal direction is often a valid approach direction in shapes that are mostly convex. But it is not necessarily so, as shown in Figure 3.4, and it is here that the crux of the problem lies. Ironically, the search for a toolpath is only being conducted in a 3–dimensional manifold that is "wrapped" around the surface. The commercial approach can, and often does, lead to interference problems with disastrous consequences: broken tools and destroyed parts. The only way to prevent the problem is to verify the cut visually using simulation tools such as Vericut™ and to manually edit the toolpath. This step makes toolpath generation very time consuming, and the resulting "tweaked" toolpaths are usually sub–optimal.

To avoid such problems, it is necessary to compute a general, point–access map, $A_{\Delta V}: \Delta V \rightarrow S^2$, that varies from point to point for the entire region of the delta volume ΔV for 5–axis toolpaths. Unfortunately, compared to the face–access map in the 3–axis situation, the 5–axis generalization causes an explosion in the actual computational burden. Below we discuss how to compute this information in a tractable way using techniques from computer graphics.



In parts that exhibit concavity the surface normal direction need not be a valid accessible direction.

Figure 3.4: Surface normal is not "necessarily" a valid access direction

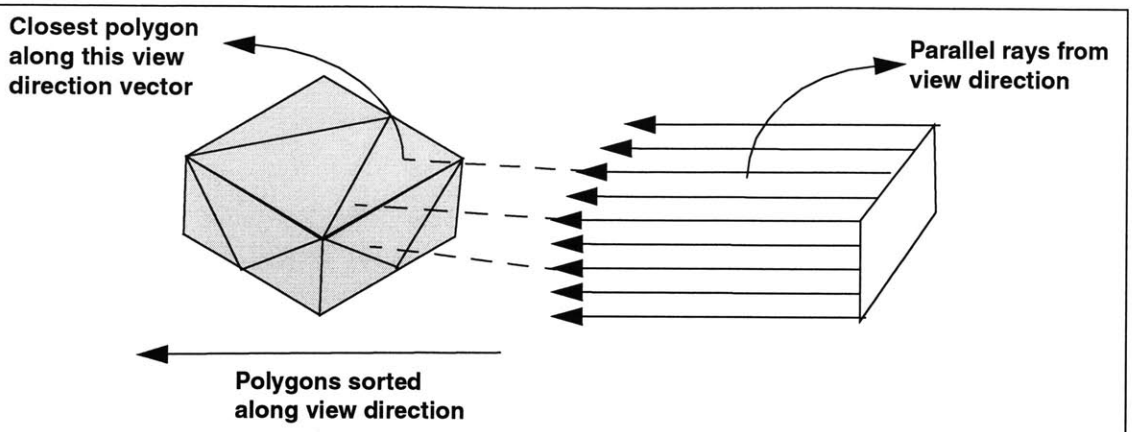


Figure 3.5: Software implementation of hidden surface removal

3.3 Software approach to determine visibility

Approximating accessibility as visibility reduces the task of visibility determination to the classical problem of hidden surface removal. Hidden surface removal has received considerable attention in the fields of computational geometry and graphics. A number of algorithmic solutions have been proposed in [Foley 95]. The surface of the object can be discretized into polygonal entities. The entities are loaded into the scene and sorted according to the distance from the view point.

An orthogonal projection of the part along the view direction is obtained by casting parallel rays along the view direction. By using the sorted list of polygons along the view direction we can quickly identify the polygons that are in the front and hence obtain the visibility information along a view direction. Figure 3.5 illustrates the procedure for computing the visible polygons using the software approach. The complexity of the software approach is in the sorting of the polygons and identifying the faces at the front. In the next Section we will present a hardware approach that uses the advances in graphics hardware to determine the visibility information rapidly.

3.4 Hardware approach to determine visibility

The graphics community have developed hardware solutions like the depth buffer, which make it possible to perform hidden surface removal on up to a million polygons per second in interactive computer graphics. In Section 3.4.1 we describe the advantages of using graphics hardware for hidden surface removal. Section 3.4.2 describes the process of extracting the visibility information along a view direction from the rendered scene.

3.4.1 Using graphics hardware

The depth buffer is a part of video memory used for scan conversion. Each pixel on the screen has a memory address into which the information regarding its color and depth are written. As the polygons are scan converted, the color and depth values of the polygons that are closer to the eye overwrite the existing values, enabling hidden surface removal. This hardware approach helps in building the configuration space of the workpiece very efficiently. In essence, we propose to use 3D graphics hardware as a special purpose solid modeling engine. It should be noted, however,

that the graphics technique is presented here merely as a "trick" to ensure maximum efficiency. The same analysis can be performed through software means.

3.4.2 Extracting visibility information along a view direction

The graphics engine scan converts a model into a scene. In our case, the model is the embedded design. In our visibility analysis, we are interested in "reversing" the process; after computing a scene, we are interested in determining the entities that are visible. One way to do this is to do a inverse screen transformation of all the points in the scene that are visible to get the points in the worldspace coordinates. From these points infer the part of the model that is visible. Unfortunately, this is computationally expensive.

A more efficient method is to determine visibility in the object space. This strategy permits us to do away with the inverse transformation. This is achieved by color encoding *(R, G, B)* each primitive (face) in the model. The visible part of the object is extracted by identifying the colors in the graphics scene. Using the common 24 bit *(R, G, B)* color boards with 8 bits per color 16,777,216 different primitives can be encoded.

Boundary representation (BRep) is an accurate representation of 3D shapes and their boundaries. However, BRep is too coarse a representation for computing point–access cones because faces of a solid are modeled monolithically as single entities. While convenient for solid modeling, it is not sufficient to perform visibility analysis on an entire surface with one color. Visibility of the color assigned to a whole surface does not mean that the entire face is visible. it would merely imply that a portion of the face is visible. This information is not even rich enough to indicate which portion of the face is visible.

Tessellated sampling of workpiece: We are interested in determining the point access cones and it is necessary that we sample the face with several sample points and compute the accessibility cone for each point. One option is to tessellate the surface and check visibility for every individual triangle. Appendix B describes the STL representation and how we store it internally. If the triangles are sufficiently small then this will approximate the condition of point visibility on the surface. Section 3.6 addresses the practical computing issues that arise out of using tessellated representation for visibility analysis. By approximating a tesselation as a sample region for a point access cone, we can use a tessellated representation to generate point access cones for the entire object. Because the tessellation is used as a sample region for a point access cone, the algorithm depends on the aspect ratio and the "size" of the triangle and is independent of the number of triangles in the model (though the memory requirements increase). Any errors resulting from this assumption will be corrected during tool access profiling. Since the size of the triangles is kept small, the errors resulting from this assumption can be easily compensated.

Color encoding the triangles: In a 24 bit *(R, G, B)* graphics board each color is represented as a combination of *R, G* and *B*. Each color occupies 24 bits, with its components *(R, G, B)* occupying 8 bits each and each component is represented internally by an integer between 0 and $255(2^8-1)$. We are interested in a color encoding scheme that is compatible with the internal representation and helps in determining the region pointed to by the color in $O(1)$ computations. We present one such color encoding scheme that uses the index number of the tessellation in the STL representation. The tessellations are numbered sequentially from 1 to a maximum of 16,777,216. The index number of the triangle is converted to a 24 bit binary number. The 24 bit number can be used to assign a mixture of the three primary colors red, green and blue by taking the first eight bits as the

value of the red intensity, the next eight bits as the value of the green intensity and last eight bits as the value of the blue intensity. Figure 3.6 illustrates the procedure of visibility analysis. Algorithm 1 describes the procedure for color encoding the triangles in the mesh.

Algorithm 1: Color encoding the tessellation

Input:

TObject: Tessellated object to be color encoded

numTriangles: number of triangles in mesh

Output:

TObject: Color encoded triangles in tessellation

Algorithm:

```
colorValue ← {0,0,0}
For i ← 1 to TObject.numberOfTriangles
    curTriangle ← TObject.getTriangleWithIndexNumber(i)
    hexRepresentation ← representation of i in hexadecimal form
    colorValue.blue ← hexRepresentation & 0x000000ff
    colorValue.green ← (hexRepresentation & 0x0000ff00) >> 8
    colorValue.red ← (hexRepresentation & 0x00ff0000) >> 16
```

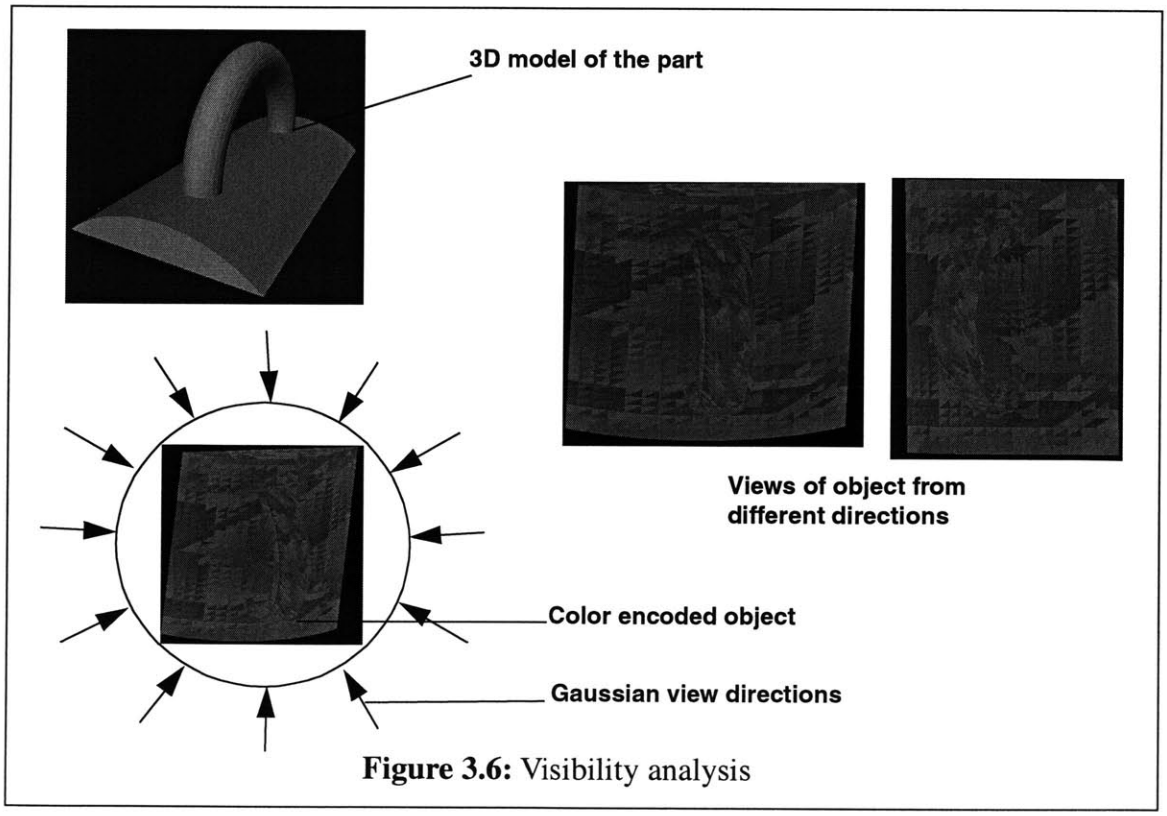


Figure 3.6: Visibility analysis

```
        curTriangle.assignColorValue(colorValue)
    end For
```

For each orientation, the model is rendered and then the *(R,G,B)* buffer is queried to obtain the color values of every pixel in screen space. From the *(R,G,B)* values, the corresponding triangles are identified. The reverse process—i.e., determining the identity of a triangle from its color—is equally straight forward. Concatenate the binary values of the red, green and blue intensities into a single 24–bit binary number. This binary number yields the identity of the triangle. To increase the robustness of the analysis, all the computations are performed in object space. The image space is used to merely to identify the triangles that are visible; no numerical information is taken from the image space. Algorithm 2 describes the procedure for determining the visibility information along a view direction.

Algorithm 2: Determining visibility information along a view direction

Input:

$O(\theta, \varphi)$: Orientation along which visibility analysis should be performed

T : Triangular mesh

X, Y: Size of the image

Output:

VT : Set of object triangles that are visible along the view direction

Algorithm:

```
VT ← {}
render (T, O(θ, φ))
for i ← 1 to X do
    for j ← 1 to Y do
        color ← getcolor_of_image(i, j)
        visibleTriangle ← get_triangle_with_color(color)
        VT← VT ∪ visibleTriangle
    end For
end For
```

In the next section we describe a procedure for determining the point visibility cones for the entire object by accumulating the visibility information from several directions.

3.4.3 Determining point visibility cones

We sample the workpiece from a discrete number of orientations arranged around the Gaussian Sphere. From each of these directions we will ascertain the visibility of the various entities in the workpiece using graphics arguments. The visibility information can be accumulated in a matrix shown in Figure 3.7. The size of the matrix is (m x n), where m is the number of orienta-

	Tri 1	Tri 2	Tri 3	Tri 4	Tri 5	..	..	..	Tri n
Dir 1	1		1				1		1
Dir 2		1		1					
Dir 3	1	1		1				1	
..									
Dir m	1		1		1			1	

Figure 3.7: Visibility matrix

tions in the Gaussian sphere and n is the number of triangles in the object. Section 3.4.2 described a procedure for determining visibility information along a view direction i to fill the visibility matrix row–wise. The point visibility cone for a object triangle j can be interpreted as the set of discrete directions from which it can be accessed. Therefore, the visibility matrix can be interpreted column–wise to obtain the point visibility cones of the object triangles. Algorithm 3 describes the procedure for determining the visibility matrix.

Algorithm 3: Determining visibility matrix

Input:

$OSet(\theta, \varphi)$: Specified set of orientations from which to perform visibility analysis

T : Color Encoded Triangular mesh

Output:

VM(numberOfDirections, numberOfTriangles): Visibility Matrix

Algorithm:

```
numOfDirections ← OSet(θ, φ) .getNumberOfOrientations()
for i ← 1 to numOfDirections
    viewDirection ← OSet(θ, φ) .getViewDirection(i)
    Determine the set of visible triangles VT using algorithm 2
    for j ← 1 to VT.length()
        triangleNumber ← VT.getVisibilityTriangle(j)
        VM(i, triangleNumber) ←1
    end For
end For
```

3.5 Viewing directions

For the algorithm specified in Section 3.4 to yield the point visibility cones, we have to generate a "good" set of view directions. Section 3.5.1 lists the functional requirements of the set of viewing directions and Section 3.5.2 gives a procedure for determining a set of viewing directions

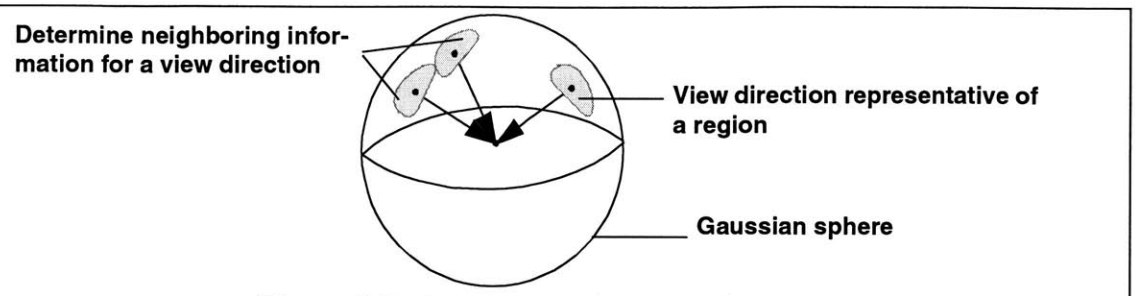


Figure 3.8: Requirement for a set of view directions

that satisfy the requirements.

3.5.1 Requirement for viewing directions

There are two issues to be considered here.

1. It is necessary that we consider a fairly uniform sampling of the Gaussian Sphere. We are dealing with a discrete space of viewing directions. Therefore the selected view direction should be representative of a patch on the Gaussian sphere. This is shown in Figure 3.8, where a view direction is representative of a patch on the Gaussian sphere that corresponds to a group of view directions in its neighborhood.
2. We are collecting the visibility information using a set of discrete orientations and a pixelated object. Because of the discrete nature of our approach, it is possible that we may have some discontinuities in the point access cones. The adjacency information of the Gaussian patches is required to fix the discontinuities based on properties of point visibility cones. Figure 3.8 illustrates this requirement for two neighboring Gaussian patches.

3.5.2 Generating viewing directions

An algorithm to generate the view directions that satisfies the requirements is presented here. An icosahedron that is inscribed in a sphere with 20 triangular facets and 12 vertices is supplied as an input to the algorithm. The triangles with its vertices on the surface of the sphere are called Gaussian triangles to distinguish it from the triangles in the model. The 12 vertices of an icosahedron are given by Banchoff [Banchoff 96]: $(\pm 1, 0, \pm t)$, $(0, \pm t, \pm 1)$ and $(\pm t, \pm 1, 0)$, where the golden ratio t is $(\sqrt{5}-1)/2$. A Gaussian triangle represents the patch on the Gaussian sphere for a view direction from its centroid to center of sphere. A finer distribution of view direction with smaller Gaussian patches can be achieved by repeatedly sub–dividing the triangle into 4 parts and projecting the triangles onto the Gaussian sphere. Algorithm 4 describes the procedure to obtain a uniform sampling of the Gaussian sphere.

Algorithm 4: Gaussian sphere sampling

Terminology:

CreateTriangle(v1, v2, v3): constructs a triangle with the given three vertices

Vertex(T, i): returns the i^{th} vertex of triangle T

Input:

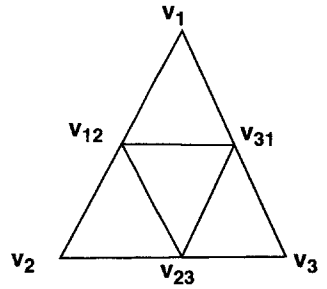


Figure 3.9: Subdivision of triangle on Gaussian sphere

levelSubDivision: Level of sub–division

$T_i\{\}$: Set of icosahedron triangles to be sub–divided

Output:

$T_o\{\}$: Set of triangles approximating a sphere

Algorithm:

```
For i ← 0 to levelSubDivision
    For Each T belonging to T_i{ }
        // Figure 3.9 shows the vertex indices used in the algorithm
        v1 ← Vertex(T, 1)
        v2 ← Vertex(T, 2)
        v3 ← Vertex(T, 3)
        v12 ← 0.5*(v1+v2)
        v23 ← 0.5*(v2+v3)
        v31 ← 0.5*(v3+v1)
        Project v12, v23 and v31 on to the surface of the sphere
        To.ADD(CreateTriangle(v1, v12, v31))
        To.ADD(CreateTriangle(v12, v2, v23))
        To.ADD(CreateTriangle(v3, v31, v23))
        To.ADD(CreateTriangle(v31, v12, v23))
    end For
    Copy To to Ti
    Initialize To
end For
return To
```

Figure 3.10 shows the polygonal approximation of a sphere for 80, 320 and 1280 directions. The "included angle" between adjacent view directions can be approximately determined by assuming that the surface of the sphere is filled with equilateral triangles. If there are n equilateral triangles of length a on the surface of the sphere of radius R, then area coverage gives the relation

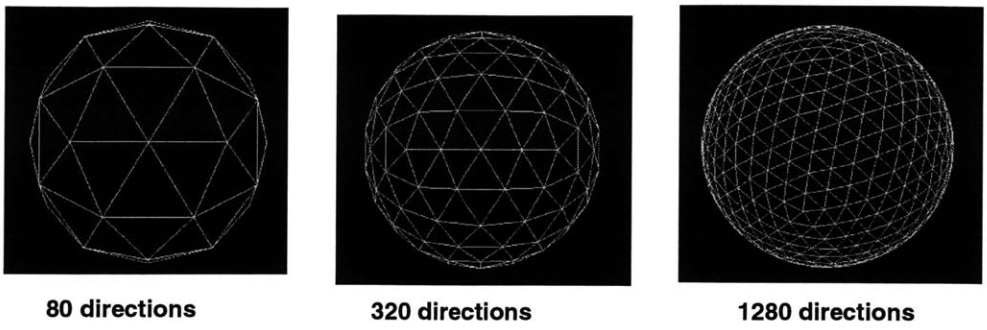


Figure 3.10: View directions

$4\pi R^2 = n\frac{\sqrt{3}}{4}a^2$. The included angle is approximately given by the relation, $\theta \sim \text{atan}\left(\frac{a}{R}\right)$. Table 3.1

Table 3.1: Included Angle as a function of number of view directions

Number of Directions	Angle(Degrees)
80	19.93
320	5.18
1280	1.3

gives the included angle for different values of n. The user can choose the number of view directions based on his accuracy requirements. Moreover, the number of view directions has an important bearing on the memory requirements and this has been discussed in detail in Section 3.6.4.

The visibility cones for a triangle on the object can be represented by a patches of Gaussian triangles from which it is visible. The apex of the cone is at the centroid of the object triangle. Figure 3.11 shows the point visibility cones at several locations in a object. The presence of the handle shaped obstacle will divide the visibility cones of points beneath it. The division of the visibility cone implies that the point can be reached in a direction that is either above or below the handle.

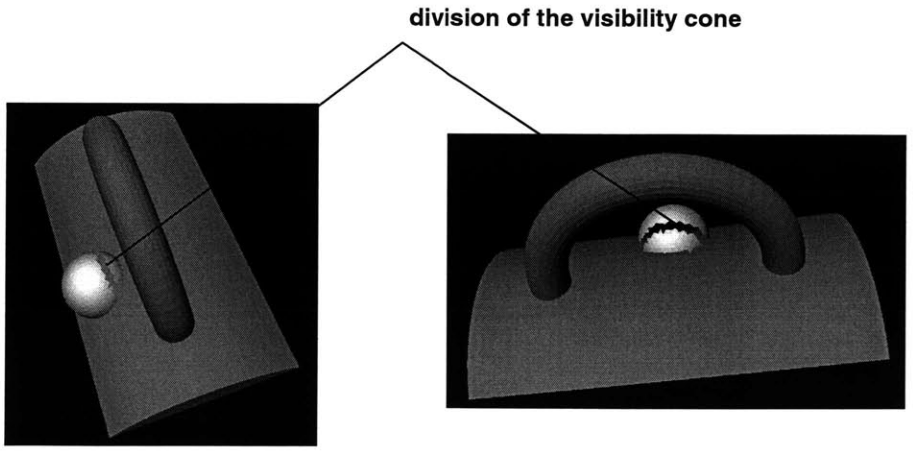


Figure 3.11: Point visibility cones

Figure 3.12: Rendering pipeline

3.6 Issues in generating discrete visibility information

The procedure for determining the discrete visibility cones has been tremendously accelerated by the use of graphics hardware. However, there are several issues that arise out of this approach as opposed to the software implementation of the hidden surface removal. Section 3.6.1 addresses the issue of part size and orientation of the face in the process of visibility analysis. Section 3.6.2 describes the reason for holes occurring in the visibility cones and how we can use the adjacency information in the Gaussian patches to fill these holes using geometric arguments.

3.6.1 Dependence on the size of the part

Figure 3.12 shows a rendering pipeline by which a 3D part is displayed as a 2D image [Foley 95]. The frame buffer has a limitation on number of pixels it can handle, typical sizes of the frame buffer are 1280×960 pixels. The object must to be scaled to fit the viewport where the visibility analysis is performed. The amount of scaling is limited by the fact that we would like the entity to have a minimum number of pixels so that it can be spotted in the visibility analysis. Let,

$\vec{n}$, be a vector representing the outward normal of the entity

$\vec{v}$, be a vector representing the view direction

A, be the area of the actual entity

p_x and p_y be extent of the projected entity on world space coordinates

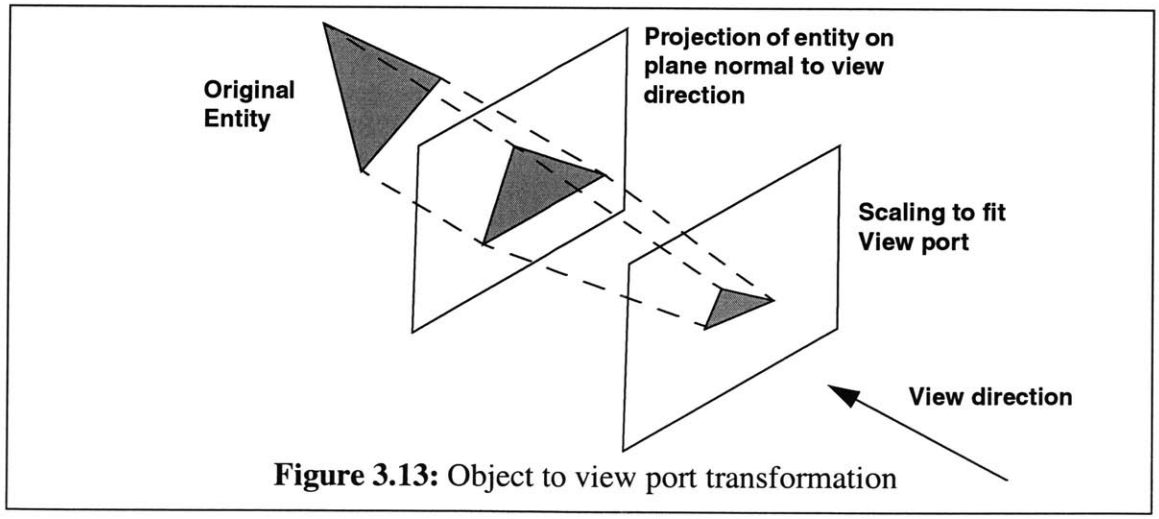


Figure 3.13: Object to view port transformation

w_x and w_y be the extent of the view port

k be the permissible minimum number of pixels on the entity in the view port

The projection of the area on a plane perpendicular to the view direction is given by:

$$A_{normal} = A|\vec{n} \bullet \vec{v}| \tag{3.1}$$

The number of pixels that the entity has in the view port is given by:

$$numberPixels = A_{normal}\left(\frac{w_x}{p_x}\right)\left(\frac{w_y}{p_y}\right) \tag{3.2}$$

The condition *numberPixels* > k, gives the limit on the size of the part based on screen limits and a view direction. Although the use of graphics hardware is not central to our approach, we have presented it here as a means to accelerate the visibility analysis. A potential problem with the graphics approach is loss of resolution in the use of tesselations and pixels. Fortunately, experiments show that inaccuracies in our approach are insignificant, and the graphics approach is indeed viable. For example, consider a part of size 12" × 8" (300mm × 200mm) that has been scaled to fit the viewport of size 1280 × 960. Then, 1 mm on the part space will be represented by approximately 4 pixels in the view port, this means that triangles can be as small as 1 *mm* on the side.

The effect of the orientation of an entity on the number of pixels is captured through A_{normal}. In the mesh generator, we create triangles that are about 50 pixels (say) in area in the viewport space. At a 5° orientation, the same triangle occupies a projected area of only 5–10 pixels. Assuming that the triangle started out as an equilateral triangle (it is important to start with a reasonably well formed triangulation), the minimum width of a triangle oriented at 5° to the viewing direction is at least half a pixel. Therefore, at a 5° orientation, it is very unlikely that we will lose a triangle in the graphics based visibility approach. Our experiments have confirmed this to be the case; thus far losses due to granularity have been insignificant. When losses do occur, it is relatively inexpensive to correct them using algorithms like the side–visibility algorithm described in Section 3.6.3.

3.6.2 Holes in the visibility cone

Figure 3.14 (a) shows the various shapes that the point visibility cones can have in theory. The point visibility cones of a "real" part cannot have islands because obstructions cannot be present "freely" in the object space. Figure 3.14 (b) shows an impossible configuration for a point visibility cone.

Our approach involves several layers of discretization:

1. *Selection of view directions:* We generate a discrete sampling of the Gaussian sphere to perform the visibility analysis. It is possible for the sampling to be coarse so that the view direction is not representative of its Gaussian patch.
2. *Approximating Object triangles:* We approximate the object triangles to represent a small region on the surface. However, it is possible for the mesh generator to fill the surface near

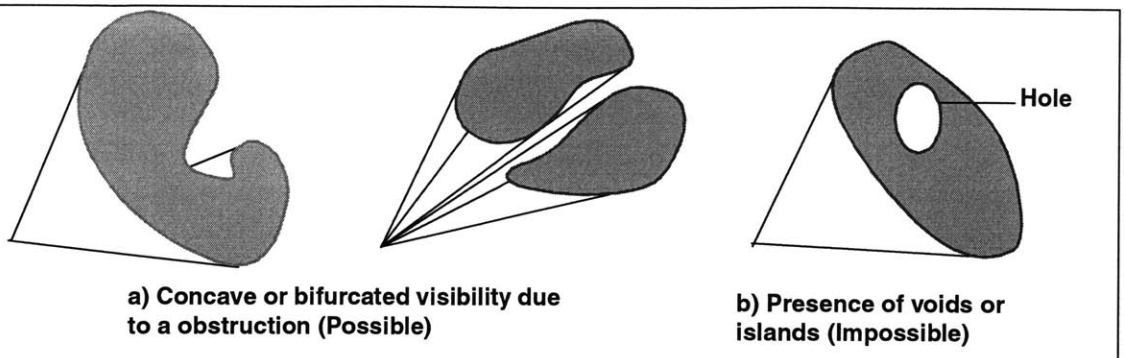


Figure 3.14: Shapes of continuous visibility cones

the face edges with ill–conditioned triangles. At some orientations, the projected area A_{normal} of ill–conditioned triangles can be small and may be "missed" in the visibility analysis.

False negatives and no false positives: In theory the visibility of a point along a view direction can be determined by checking whether a ray shot from the point along a direction opposite to view direction reaches infinity. The Gaussian triangles can be classified into four types based on the "theoretical" visibility of a point from a Gaussian view direction and the results observed in our discrete approach.

1. True positives: Visible in both theoretical and discrete approach.
2. True negatives: Not visible in both theoretical and discrete approach.
3. False negatives: Visible in theoretical approach but not reported in discrete approach.
4. False positives: Not visible in theoretical approach but reported visible in discrete approach.

It should be noted that the discrete visibility cones generated by the visibility analysis can have only false negatives and no false positives. In other words, our approach may miss recognizing a surface triangle as being visible from a Gaussian view direction because of discreteness due to insufficient pixels to represent the surface triangle. Since we can err only by missing "valid" Gaussian triangles it is possible for the discrete visibility cones to have holes.

Filling holes in visibility cones: We can use geometric arguments based on adjacency information to add Gaussian triangles that fill holes in the point visibility cone. We can use the algorithms suggested by Martin [Martin 98] for "fixing" STL files. In our implementation, we have used his algorithm to generate the adjacency information of the Gaussian triangles rapidly. The adjacency information was used to identify a loop of triangles that enclose the holes in $O(n)$ time, where n is the number of Gaussian triangles in the visibility cone. The promising "false negative" Gaussian

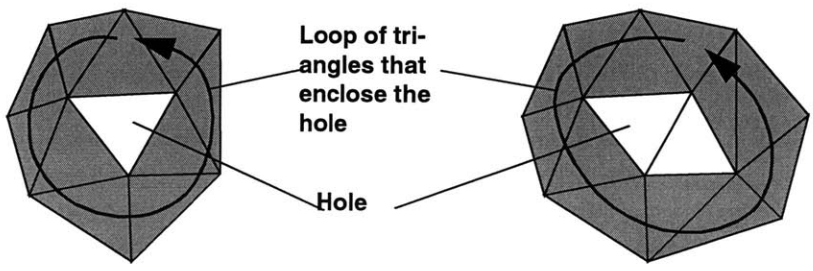


Figure 3.15: Filling holes in visibility cones

triangles can be identified quickly from the adjacency information of the Gaussian triangles that form the loop. We can add the promising "false negative" Gaussian triangles after checking if it was infact missed due to the discreteness in our approach. Figure 3.15 shows a discrete visibility cone with a hole due to a missing Gaussian triangle/s. The loop of triangles that enclose promising "false negatives" is also shown in Figure 3.15.

3.6.3 Sideways visibility

One problem with using the visibility argument is that faces parallel to the visibility direction are not usually visible from that direction. Yet, in machining, a parallel face may be accessible from a parallel direction. Laxmiprasad [Laxmiprasad 98] suggests a side correction algorithm that adds Gaussian triangles to the point visibility cone based on neighborhood information and the inclination of the object triangle with respect to the orientation corresponding to the Gaussian triangle.

3.6.4 Memory requirements and computations

In this section, we will make an estimate of the memory requirements for performing the discrete visibility analysis for an object mesh with m triangles and n viewing directions. Following are the stages in the visibility determination:

Formation of the visibility matrix: The visibility matrix is computed rowwise by determining the visible object triangles for every view direction. Since the process of generating visibility information along a direction is independent of the previous direction, we can refresh the memory at the end of every stage and the memory requirement does not scale up with the number of view directions. However, the total computations will increase linearly with the increase in number of view directions.

Computing point visibility cones: The visibility matrix generated in the previous stage is used to determine the point visibility cones. The Gaussian sphere is discretized up front and is held in memory. For every object triangle, we store a pointer to the Gaussian triangle from which it is visible. Moreover, the number of Gaussian triangles in a point visibility cone can atmost be $n/2$ because an object triangle can be seen from directions that lie in upper hemisphere defined by the outward triangle normal. If our implementation takes p bytes of data to store a pointer in memory, memory requirement on an average will be $O(p \times n/2 \times m)$. For a real world application; $n \approx 1280$, $m \approx 10{,}000$ and $p \approx 8$. The memory requirement in the worst case evaluates to around 50 MegaBytes. The number of computations will be $O(n \times m)$.

The trade–off involved in selecting the number of view directions for performing the visibility analysis are higher accuracy versus the memory requirements. For the purpose of toolpath generation, 1280 view directions are sufficiently accurate to capture the visibility information. Chapter 4, describes the procedure for access profiling that takes the visibility information and generates valid access directions for tools. Any errors due to discretization in the visibility will be corrected during the access profiling stage.

The highlight of our approach is that the point visibility cones can be generated for a portion of the object. However, we have to include the entire object for performing the visibility analysis. Fortunately, it turns out that "occlusion" properties of a triangle is not affected by its size. Therefore, we can save on memory requirements by sampling the region of interest with "small" trian-

gles and fill the remaining portions of the object with "large" triangles. Therefore the advantages are two–fold:

1. Decrease in memory requirements due to reduction in number of object triangles for which the visibility cones are determined.
2. The scene can be rendered faster by the graphics hardware because the number of entities in the scene has been reduced considerably.

Chapter 4: Accessibility determination

Chapter 3 gave a methodical procedure to determine the visibility cones for points on the surface of the object. Visibility does not ensure accessibility, because visibility assumes a ray of light to access the point whereas accessibility is for a tool of finite geometry trying to access the point. In this chapter we are concerned with determining guaranteed legal access directions, or postures, from which a tool can access a given region in the surface or delta volume without collision. Section 4.1 discusses the importance of the access determination problem, the difficulty in obtaining the cones of accessibility and how visibility cones could be used as an approximation for accessibility in a "global" sense. Section 4.2 lists the requirements of the collision avoidance algorithm to convert a "promising" direction suggested by the visibility cone into a feasible access direction. Sections 4.3 and 4.4 describe our implementation for access determination using a very fast collision detection algorithm.

4.1 Need for access determination

Collision avoidance can be viewed alternatively as *access assurance*. In other words, how can the cutting tool be oriented to *access* the delta volume without intersecting with the embedded design? An interference free tool direction will be referred to as an *access direction*. The access direction may not be unique. We therefore talk of access cones, which are contiguous sets of access directions. The access cones give a rich information for planning the toolpaths based on a machining strategy, but it is very difficult to generate access cones directly and reason for the difficulty has been described in Chapter 3.

For the class of non-overhung tools described in Appendix A, the point access cone is a subset of its point visibility cone and the difference between them is attributed to the geometry of the tool. Figure 4.1 shows a 2D view of a part along with a point visibility cone for point x in the delta volume of the part in red. The accessibility cone for a tool trying to access the point is shown in blue. It should be noted that the access cone is a subset of visibility cone and for the purpose of machining we are interested in determining a "good" access direction to reach a point. Therefore, we can arrive at a "promising" access direction by treating the visibility cone as a rough approximation for its accessibility cone. At every stage of the toolpath generation we have to first check for collisions between the tool positioned along the "promising" orientation and the part and then eliminate it to determine a collision free "valid" access direction.

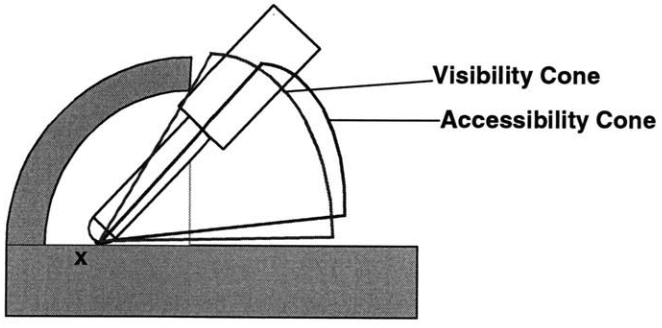


Figure 4.1: Visibility and Accessibility Cone

4.2 Requirements for rapid access determination

The visibility analysis gives the "most promising" access direction for accessing a point in the delta-volume of the part. As explained in Section 4.1, the access is not guaranteed to be collision free. The following are the main requirements of the collision detection and correction algorithm.

1. Detect collisions between the tool and the workpiece and interpret the profile of the penetration to move the tool away from the collision.
2. After the correction, the cutting portion of the tool should be in contact with the surface and the non-cutting portion of the tool should be well off the surface or any obstruction.
3. The algorithm should be scalable with respect to the tolerance specification on collision that is permissible between tool and the workpiece.

Requirement 1 is important from the point of view of implementation of the system. Inorder to compute the correction when a collision is reported by the algorithm, we must be able to extract the penetration information rapidly. Since the collision detection and correction procedure would be invoked at every point of the toolpath during the generation process, it is absolutely essential for this invocation to run in the order of 1 milli-second per collision detection and correction. Infact, this turns out to be an important requirement for real-time haptic rendering of interaction between 2 objects [Massie 96]. Requirement 2 is important to prevent the tool from missing some of the important features on the surface of the object. Requirement 3 is of practical importance for a commercial system.

4.3 Rapid collision detection

As a proof of concept we first implemented the system using a collision detection library called RAPID to convert the "promising" access direction into a "valid" access direction. Section 4.3.1 describes how the input was modelled and how the results outputted by the collision detection algorithm were interpreted as a penetration. Though the implementation demonstrated the concept, the solution was not scalable when higher accuracies were sought. Section 4.3.2 presents another implementation using the HIPS collision detection algorithm and the reason why it was scalable.

4.3.1 RAPID [Gottschalk 96]

RAPID is a *robust and accurate polygon interference detection* library for large environments composed of unstructured models. It is applicable to polygon soups - models which contain no adjacency information, and obey no topological constraints. The models may contain cracks, holes, self-intersections, and nongeneric (e.g. coplanar and collinear) configurations. The algorithm is numerically robust and not subject to conditioning problems. The fundamental data structure underlying RAPID is the OBBTree, which is a hierarchy of oriented bounding boxes [Gottschalk 96].

The STL representation of the object is used as the model for collision detection from the object side and the tool is modelled as a series of triangulated slabs. We refer to the triangle on the object and tool as a object triangle and tool triangle respectively. Figure 4.2 (a) shows a tool modelled as a series of triangulated slabs. Figure 4.2 (b) shows a tool positioned along the "most prom-

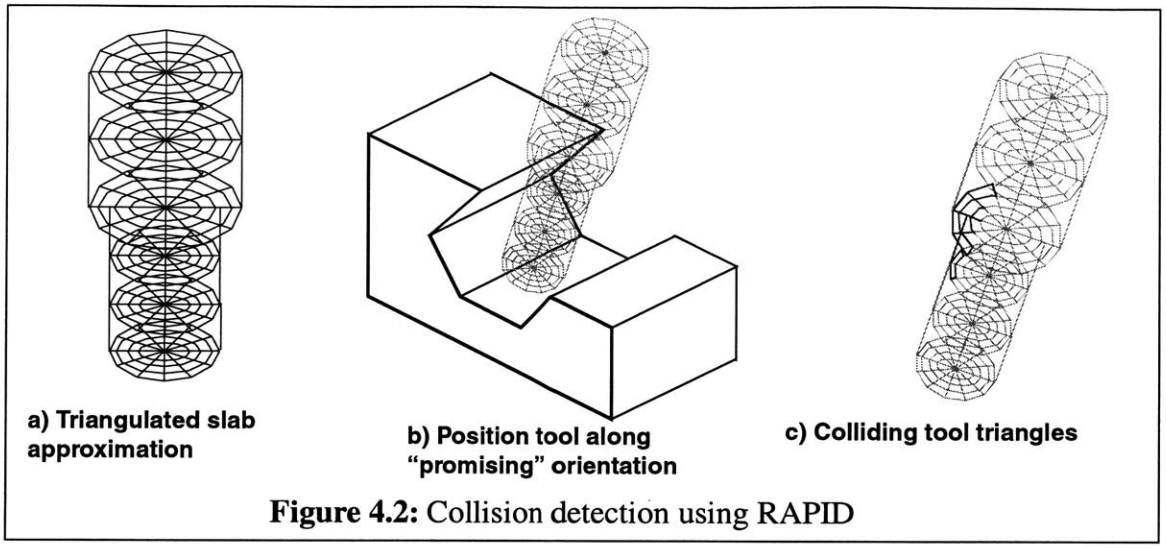


Figure 4.2: Collision detection using RAPID

ising" orientation suggested by the visibility analysis. The RAPID collision detection algorithm would return pairs of object and the tool triangles that are in collision. The penetration information can be determined by processing the result set of the RAPID procedure. Figure 4.2 (c) shows the intersecting tool triangles in red. Inorder, to estimate the penetration we made an approximation that the centroid is representative of a tool triangle in collision. Section 4.4.1 describes the procedure for correcting the tool so that it comes out of collision.

The desired accuracy of the correction scheme determines the number of triangles in the tool model because the characteristic length of a triangle should be of the same order of the desired accuracy. Infact, the number of triangles increases non-linearly with the increase in accuracy and this led to problems with scalability. An alternative was to compute the exact penetration between the object and tool triangles but this involved a lot of additional computation and slowed down the phase of penetration determination. The next section describes the implementation using HIPS algorithm [Ho 1999].

4.3.2 HIPS [Ho 1999]

In most previous work in collision detection, the same level of representation is used for both the objects in the collision. This is necessary because most previous approaches have been devel-

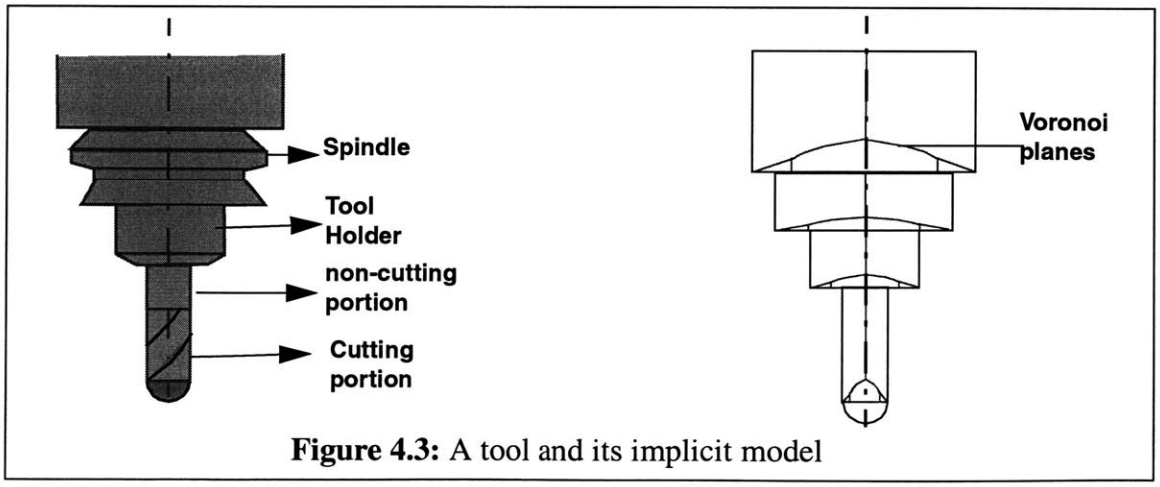


Figure 4.3: A tool and its implicit model

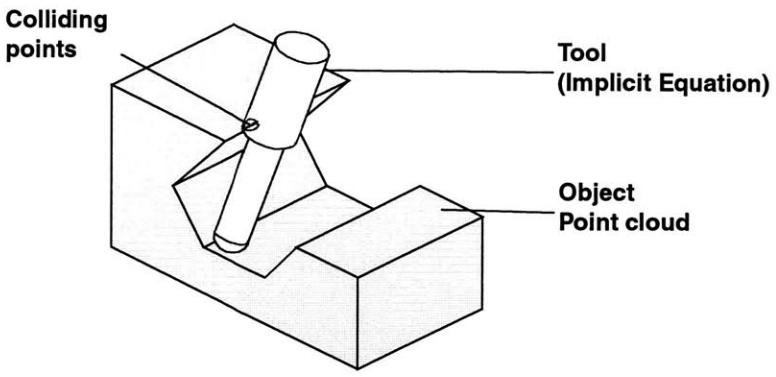


Figure 4.4: Collision detection using HIPS

oped with very general applications in mind. In the HIPS approach, different representations are used for the two objects: a high level implicit representation and a low level point cloud representation. The reasoning is as follows: For applications such as gouge detection and avoidance during toolpath simulation, the tool object is usually well known in advance. However, the workpiece needs to be very general to accommodate a wide variety of shapes. We can therefore pre-compute the implicit representation for the known tool and use the general point cloud representation for the unknown terrain. Figure 4.3 shows a real tool and its implicit representation for the purpose of collision detection.

Figure 4.4 shows the collision between the "implicit" tool and the "point cloud" object when the tool is positioned along the "promising" orientation suggested by the visibility analysis. The points of the object that are in collision with the tool are shown in red.

Scalability of the collision detection algorithm: With regard to determining penetration depth, we note that as long as one of the representations is a full solid, it does not matter if the other representation is not topologically complete. Furthermore, if one of the solids is a full implicit representation, then depth information can be inferred very simply and accurately from the interfering point cloud. For these reasons, our use of heterogeneous representations is suited for applications where one of the object is known in advance.

4.3.3 Effect of discrete representation on achievable tolerance

It should be noted that discrete representation have been used by several authors [Angleton 89,

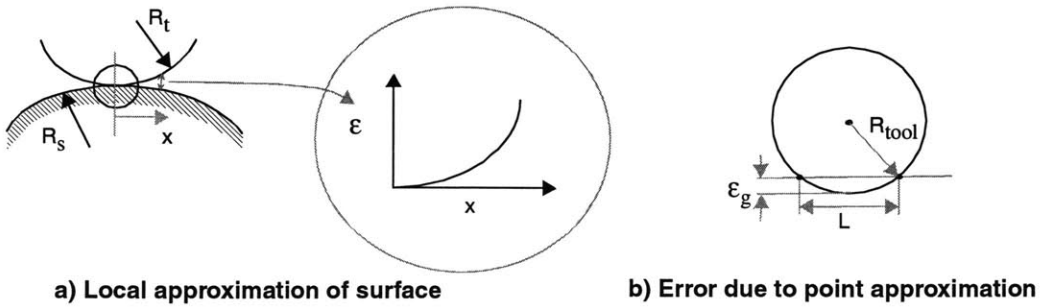


Figure 4.5: Error bounds due to discrete representation

Jerard 89a, Saito 91, Choi 94] for the purpose of toolpath generation. The machined surface must be produced to within the specified finish tolerance. Choi [Choi 98] states that the specification that is relevant to sculptured surfaces is the profile tolerance. The tolerance zone is created by the envelope surfaces created by sweeping a sphere of radius($\varepsilon_{tolerance}$) along the surface. The accuracy of the collision detection is determined by the largest gouge that goes undetected. Therefore, it is imperative to determine the bound for the undetected gouge based on the tool and discrete representation of the surface. Figure 4.5 (a) shows a cross section of the surface and the tool in a direction perpendicular to the feed direction. The separation between the two surfaces denoted by ε can be represented in terms of the transverse distance x [Marciniak 01]. By defining the effective curvature (κ_{eff}) as $\kappa_{eff} = \frac{1}{R_{tool}} + \frac{1}{R_{surface}}$ and using a taylor second order approximation for the surfaces, we get: $\varepsilon = \frac{1}{2}\kappa_{eff}x^2 + O(x^3)$. We can reduce the error desired by decreasing the spacing between the points. Consider the case of a sphere of radius R_{tool} resting on two points that are separated by L. The ε can be determined by substituting $R_{surface}$ as ∞ and x as $L/2$, we get $\varepsilon = \frac{1}{2}\left(\frac{1}{R_{tool}}\right)\frac{L^2}{4} + O(x^3)$. It should be noted that this is the error coming because of the discrete approximation ignoring the effect of surface curvature.

From practical point of view, Drysdale [Drysdale 89] determines the inter-point distance for achieving 90% of the tolerance with the remaining 10% being accountable to curvature effects. For an accuracy of 50 microns using a tool of diameter 25 mm, we get the inter-point distance to be 3.1 mm. Assuming an uniform distribution of points on the surface of the object, the number of points depends on the surface area of the object. For example, achieving a this translates to uniformly distributing approximately 6,000 points on a cube of side 0.1 m. For the examples shown in the thesis, we have uniformly distributed more than 30,000 points on the surface whose area is lesser than that of the cube given above. Moreover the collision detection algorithm uses a bounding volume hierarchy and the performance is not greatly affected by the increase in number of points on the object side.

4.4 Correction schemes

A particular motivation for determining access using local profiling is that checking for local intersections between a well known tool model and a workpiece is very much a tractable problem. The collision detection algorithm returns the points of the object that are in collision with the tool. For a given state of collision we do not have prior information about the shape of the set of object points in collision with the tool. Therefore it is very difficult to find the shape of penetration to determine a single correction vector that brings the tool out of collision. This difficulty forces us to pursue a different approach to determine the correction vector for the tool. The correction vectors of all colliding points can be determined rapidly by using the position of the point in the local tool coordinates and then combined to obtain a single correction that brings the tool partially or completely out of collision. Section 4.4.1 describes different types of correction schemes based on the location of the colliding point in the local coordinates of the tool. Each colliding point will be assigned a "point correction" vector that brings it out of collision based on the overall correction scheme for the tool. In Section 4.4.2 we describe the procedure for determining a single "tool correction" vector from a set of "point correction" vectors that would bring the tool out of collision. This procedure is a pseudo-gradient search in the neighborhood of the most promising access

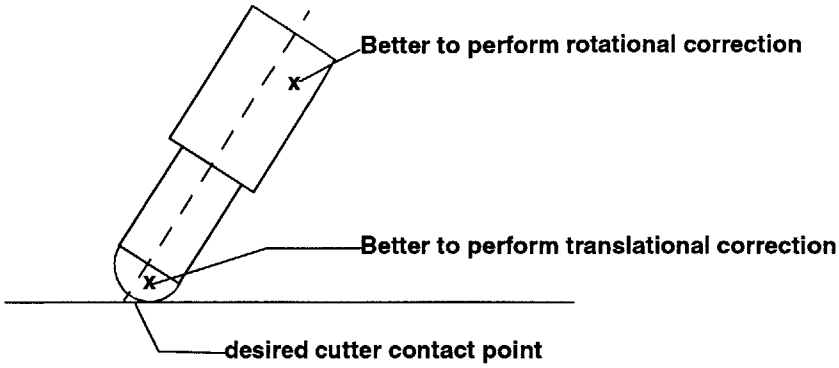


Figure 4.6: Trade–off between translational and rotational correction

direction derived from visibility analysis.

4.4.1 Types of correction schemes - translation and rotation

There are two ways in which the tool may be corrected: translational or rotational. We are interested in determining the hybrid motion that can bring the tool out of collision. We list the advantages and disadvantages of translational and rotational correction to facilitate selection of a correction type.

Invariance of cutter contact point: In the toolpath generator, the cutter contact points are generated under the assumption that the tool will be oriented to access the "pre-determined" cutter contact points. It is desired to maintain the same cutter contact point even after performing the correction. A rotational correction can be pivoted about the cutter contact point so that the contact point is not shifted by the correction. The translational correction moves the entire tool away from the collision and hence shifts the cutter contact point. Though this motion is desirable to accommodate for errors made by the toolpath generator, it will result in lack of control over the position of final cutter contact point after the correction.

Efficiency of correction: The efficiency of a translational or a rotational correction in moving away from a collision is dependent on the distance of the intersecting point from the cutter contact point. As shown in Figure 4.6, collisions close to the desired cutter contact point can be effectively corrected by using translation rather than a rotation. Collisions far away from the cutter contact point can be more effectively corrected by rotation. The user must prudently determine a cut-off distance parameter to perform a translational or a rotational correction. In our implementation for a ball tool, translational correction was applied for collisions in the hemispherical portion and rotational correction for collisions in the cylindrical portion.

4.4.1.1 Determining exit vector for a colliding point

When there are collisions in both the hemispherical and the cylindrical portions of the tool the correction scheme is determined by the set of colliding points. It should be noted that if there are colliding object points that can be "effectively" corrected by a translational correction then the entire tool must translate to come out of collision. Therefore the assignment of a translational correction scheme takes precedence over rotational correction scheme.

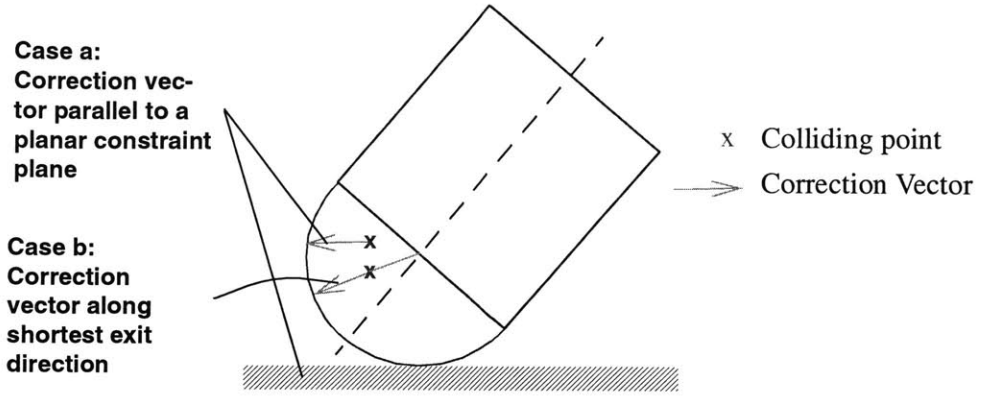


Figure 4.7: Translational correction

The correction vectors are computed individually for the set of points $\{P_{\text{colliding}}\}$ that determined the type of correction. Every colliding point in $\{P_{\text{colliding}}\}$ is assigned a correction vector that brings the point in isolation out of collision along the shortest distance between the point and the tool surface. For the case of translational correction, there can be additional constraints on the direction of an exit vector due to machining considerations. For the case of roughing, which will be discussed in greater detail in Chapter 4, the lowest portion of the tool should remain in the plane for which the cutter contact points have been generated. Figure 4.7 shows the two cases for determining translational correction vectors. The two cases differ due to the presence of a constraining plane. Case (a) shows the translational correction for a colliding point that is constrained to be in a plane parallel to the sliceplane. Case (b) shows the translational correction for a colliding point that is free to exit through the closest tool surface.

A rotational correction for a colliding point is defined by a normalized axis vector and a correction angle. The axis of rotation is perpendicular to the plane containing the tool center (for example, the center of the ball in a ball end-mill), the colliding point and the radial exit point of the colliding point. The magnitude of the correction angle is obtained from the length of the radial exit vector and the distance of the point from the pivot point. Figure 4.8 shows the parameters of rotational correction for a colliding point.

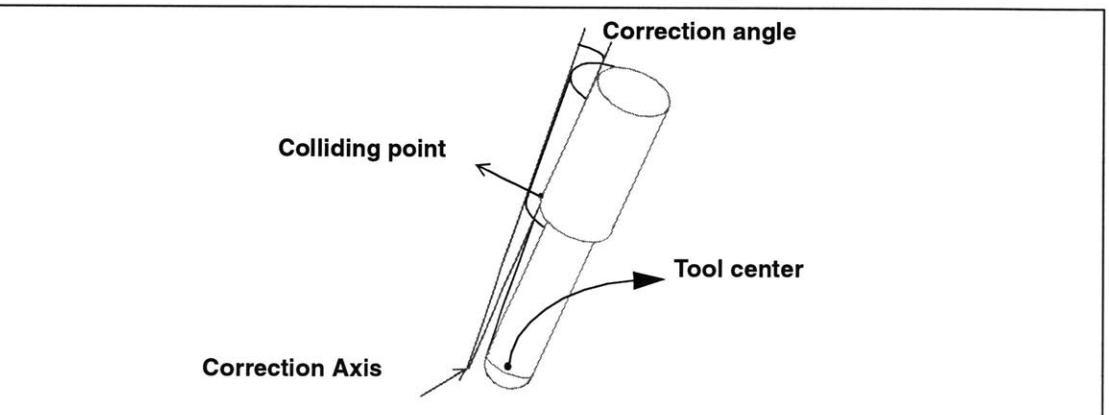


Figure 4.8: Rotational Correction

4.4.2 Composite/Iterative correction

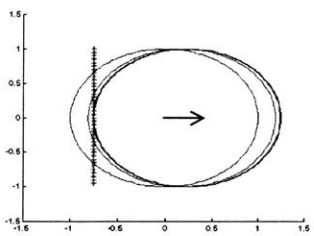
Section 4.4.1 described a procedure for determining the correction type and a set of correction vectors represented as $\{d_i, \vec{v}_i\}$ for all the colliding points. For the case of translation, $\vec{v}_i$ represents the unit vector along the exit direction and d_i is a non-negative scalar representing the translation distance. For the case of rotation, $\vec{v}_i$ represents the unit vector along the axis of rotation and d_i is a non-negative scalar representing the rotation angle.

The composite correction can be computed by assuming the tool as a rigid body and being acted upon by either pure forces ($\vec{F}_i$) or pure torques ($\vec{\tau}_i$). The $\vec{F}_i$ and $\vec{\tau}_i$ can be determined from $\{d_i, \vec{v}_i\}$ by using a linear constitutive relation between the force or torque and the penetration. By applying Newton's II law of motion, the direction for composite correction will be along either $\sum_{i=1}^{n} \vec{F}_i$ or $\sum_{i=1}^{n} \vec{\tau}_i$, where n is the number of points in collision. This is identically equal to $(constant)\sum_{i=1}^{n} d_i\vec{v}_i$. In our case, we need a correction vector such that the tool is not over-corrected because over-correction will result in loss of important features on the part. In order to prevent over-correction, we compute a weighted correction vector with the weights as the magnitude of individual correction vector. The weighted correction vector is given by:

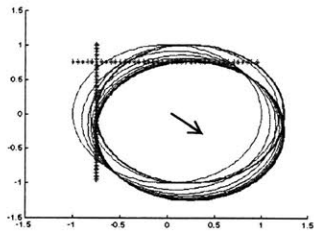
$$\frac{\sum_{i=1}^{n} d_i(d_i\vec{v}_i)}{\sum_{i=1}^{n} d_i} \tag{4.1}$$

The weighted correction is not guaranteed to bring the tool out of collision completely. Therefore, we have to repeat this process for each successive tool position till the tool comes out of collision. It should be noted that the iterative solution can be a series of corrections of different types. For example, the sequence of iterative corrections could be *translation* → *translation* → *rotation* → *translation* → *rotation* → *rotation*. We iterate between translation and rotation to nudge the tool away from the collision region using a pseudo-gradient search. In other words, we move the tool, ensure that the amount of collision has reduced, and move it again.

Convergence of iterative correction: The convergence of the approach depends on the contact tolerance between the tool and the surface and the penetration profile of the colliding points. In our experience, convergence is typically achieved in 5 - 20 iterations for a tolerance of 10^{-4} units. Figure 4.9 illustrates the iterative scheme of correction for a 2-dimensional case. A circle is initially in collision with a set of points shown in red and the position of the circle at the end of every iteration are also shown. The arrow indicates the direction of gross motion of the circle as it comes out of



Tolerance: 0.0001 units
Number of iterations: 5



Tolerance: 0.0001 units
Number of iterations: 14

Figure 4.9: Iterative correction in 2D

collision. The iterative approach ensures that the tool tip is in contact with some part of the surface of the object. In order to prevent an occurrence of an infinite loop, a parameter called `ITERATION_LIMIT` is set to limit the number of iterations. In our work we use an iteration limit of just 15 cycles. If the algorithm exits without converging, the cutter contact position is tagged as being inaccessible, and the system appeals to the user for help.

4.4.3 Asymptotic number of calculations

For a given state of collision if there are n points in collision, then we will try to estimate the approximate number of multiplications and divisions needed to bring the tool out of collision.

- Determining the type of correction $\approx O(n)$
- Assigning individual correction vectors $\approx O(n)$
- Computing the composite correction from the individual correction vectors for every iteration as a step in the iteration $\approx 2 \times n + 1 \approx O(n)$
- Maximum number of calculations for `ITERATION_LIMIT` iterations $\approx$ `ITERATION_LIMIT`$\times O(n)$, which is still $O(n)$ because `ITERATION_LIMIT` is a constant.

Chapter 5: Roughing toolpath generation

Roughing is a rapid material removal process, generally involving heavier but less accurate tools like hog mills. Roughing operations are used to remove the bulk of the material before finishing operations. In the previous chapters, we described techniques for determining point visibility cones to aid global interference avoidance. In this chapter we show how roughing can be performed regardless of the individual features. The basic idea of global roughing is illustrated in Figure 5.1, where the part to be machined is sliced to form 2.5D slab and five axis toolpaths are generated to machine every slab individually. Section 5.1 describes the procedure to determine the point visibility cones for the delta volume rapidly using the coarseness of the rough machining process. The heuristic for selecting a valid access direction from the visibility cone for points in the delta volume is described in Section 5.2. We then interpolate the tool access directions to get a smooth continuous toolpath and Section 5.4 addresses the various concerns in toolpath interpolation.

5.1 Visibility analysis for roughing

The procedure for determining the point visibility cones for surface segments have been described in Chapter 3. The accessibility information of the 3–dimensional delta volume is necessary to generate toolpaths within the delta volume. A "naive" way of determining this information is by subdividing the 3 dimensional volume using intermediate surfaces and computing the accessibility information of the intermediate surfaces using concepts presented in the previous chapters. For roughing, the slice planes that determine the 2.5D slabs can be taken as the intermediate surfaces.

Figure 5.2 shows a 3D object that has been stratified into n slabs. The toolpath for slab 1 is generated by performing the visibility analysis for the intermediate solid that would remain after machining slab 1 and this process is repeated for the other slabs till the entire object is machined. The repetitive procedure ensures that we have some surface elements to approximate the delta vol-

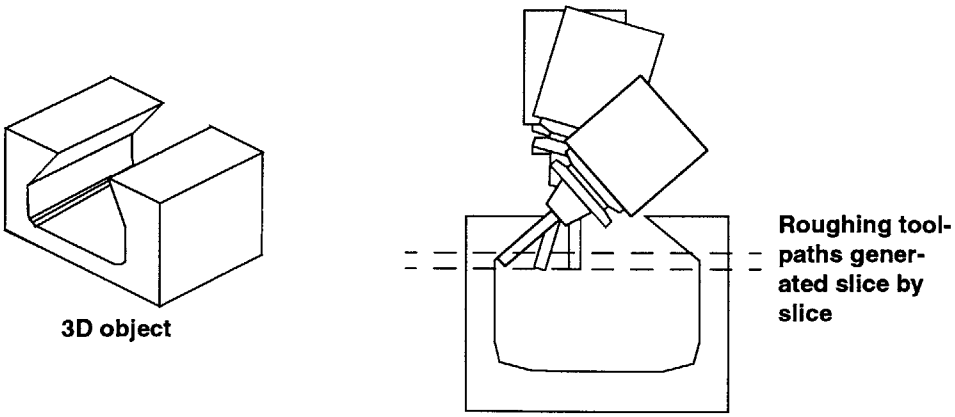


Figure 5.1: Global Roughing

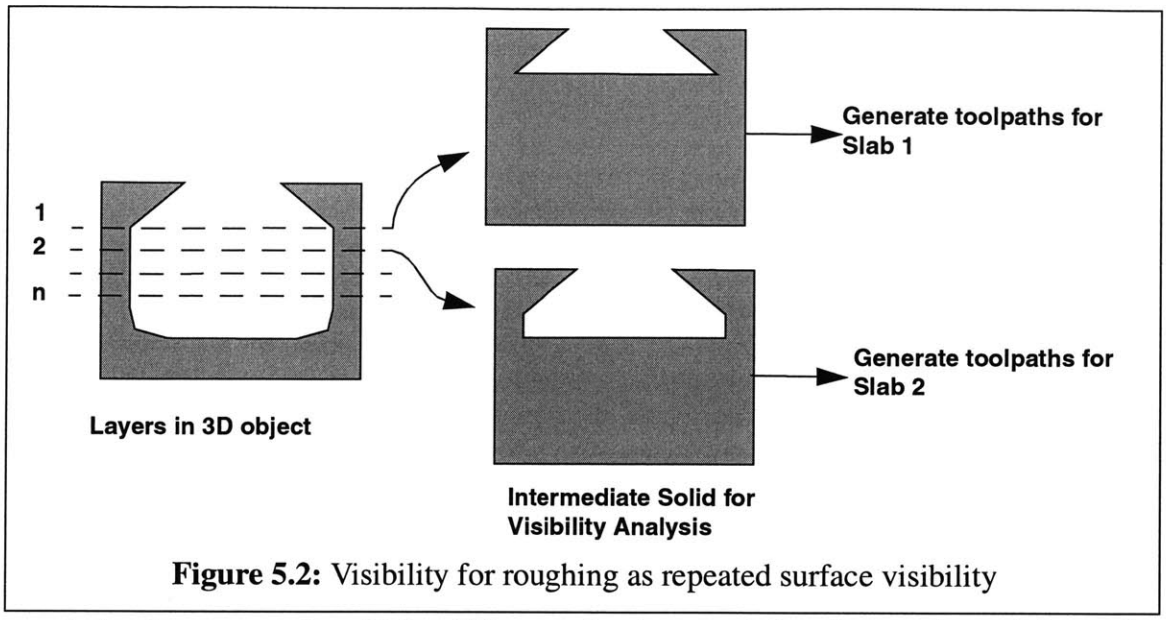


Figure 5.2: Visibility for roughing as repeated surface visibility

ume during the process of machining. This procedure is computationally intensive and the number of times the visibility analysis need to be performed is dependent on the number of layers in the object. Section 5.1.1 describes the procedure for determining the visibility cones for points in the delta volume rapidly by using the properties of the roughing process.

5.1.1 Determining visibility for roughing from surface visibility information

The delta volume is a continuous 3–dimensional region. It is necessary to digitize or sample this space in order to map visibility. We use a simple three dimensional space enumeration of the delta volume, also referred to as a voxelized representation. Just as in the case when we approximated the area occupied by a tesselation on a surface to represent a "point" for the purpose of visibility, the volume of the voxel will approximate a point in the delta volume. Section 5.1.2 describes the effect of increasing the number of voxels on accuracy, memory requirements and computational complexity. A "practical" and "reasonable" value for the number of voxels can be determined by using the coarseness of the roughing process, size of the tool used for roughing and memory requirements. We convert B–Rep to voxels by a simple scan–conversion algorithm [Foley 95, Samet 90]. Figure 5.3 shows a 3D object and its voxelized model.

From the visibility analysis, each point on the surface of the workpiece has a set of orientations along which it is visible. Consider a ray of light reaching point on the surface of the object. The ray

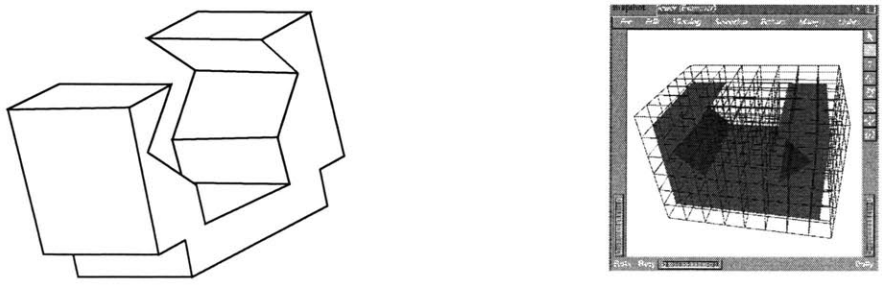


Figure 5.3: 3D object and its voxelized model

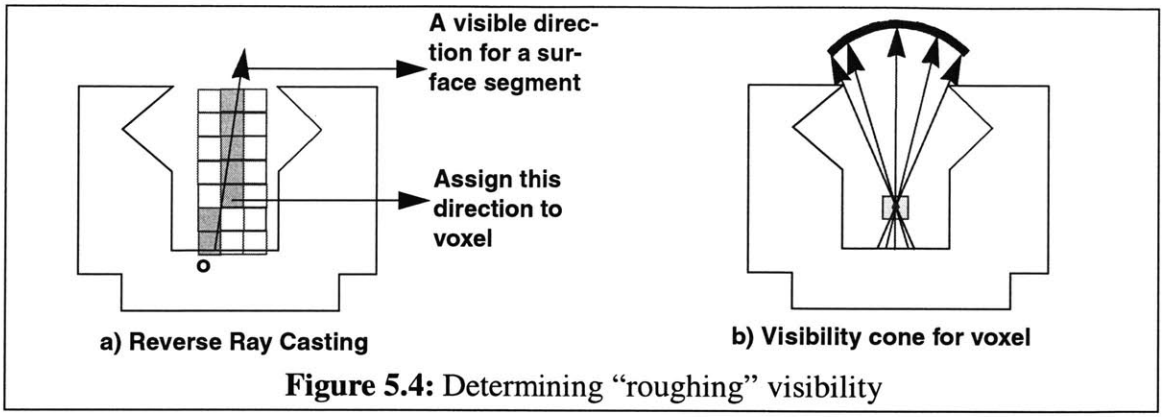


Figure 5.4: Determining "roughing" visibility

intersects several voxels as it traverses the object to reach the point on the surface from infinity. It is intuitive to assign the intersected voxels with this direction because a tool accessing along the visible direction will be able to access these voxels. Figure 5.4 (a) shows a surface point o and a ray along the reverse direction from which it was visible. All the voxels that are intersected by the ray are tagged with this access direction. We refer to the process of shooting rays from all surface segments along its visible directions as "Reverse Ray Casting". Figure 5.4 (b) shows the "roughing" visibility cone for a voxel by accumulating the direction of rays that intersected it during the reverse–ray casting. Figure 5.5 shows the "roughing" visibility cone at a voxel for an extruded surface and a teacup that has an obstruction in the form of an handle. It is clearly seen that the presence of the handle divides the "roughing" visibility cone into two parts.

By taking advantage of coherence along a line, "roughing" visibility can be computed with minimal computational effort. A direction for reverse ray–casting is selected for a surface triangle from its visibility cone. The ray originates from the centroid of the surface triangle and we deter-

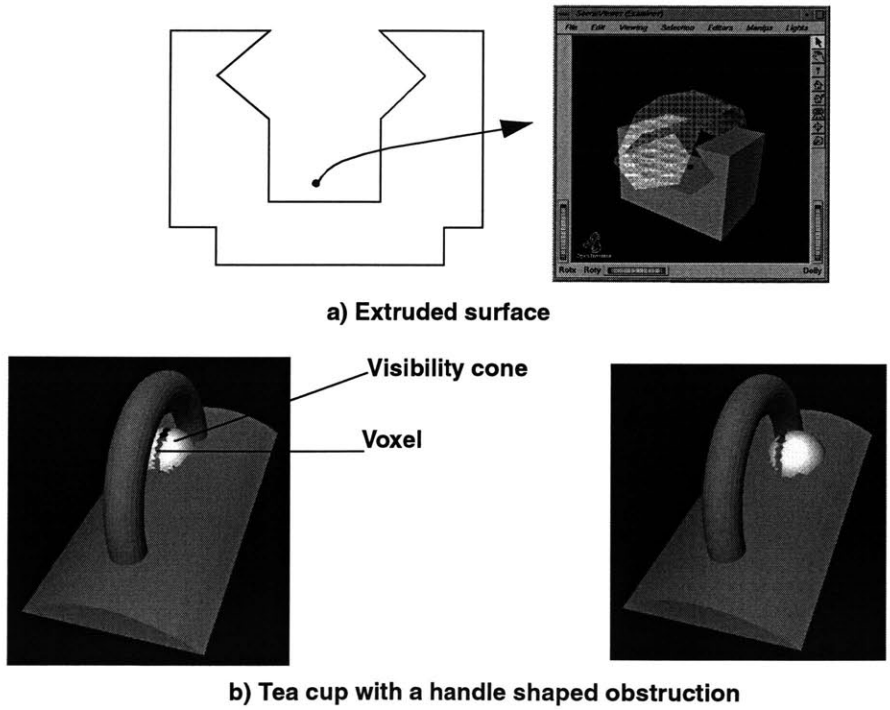


Figure 5.5: Point visibility cones for roughing

mine a point P by stepping along the direction by a length that is a characteristic of the voxelized discretization. The voxel containing point P is tagged with this direction and this step is repeated till point P exits the bounding box of the part. Algorithm 5 describes the procedure for performing "roughing" visibility.

Algorithm 5: Determining "Roughing" visibility

Input:

T: Triangular mesh

GS: GaussianSphere

$VT[noSurfaceTriangles][noGaussianTriangles]$: Visibility Matrix (from visibility analysis)

$VOX[x, y, z, \theta, \phi]$: Voxel array data structure

Output:

$VOX[x, y, z, \theta, \phi]$: Filled voxel array data structure

Algorithm:

```
voxel−size ← getMinimumVoxelDimension(VOX)
(X_min, Y_min, Z_min) ← getMinimumCoordinatesFromBoundingBox(T)
(X_max, Y_max, Z_max) ← getMaximumCoordinatesFromBoundingBox(T)
for i ← 1 to T.getNumberOfTriangles() do
    (X, Y, Z) ← T.getTriangleWithIndex(i).getCentroid()
    for j ← 1 to noOfGaussianTriangles do
        O(θ, φ) ← getDirection(GS, j)
        C_x ← (voxel−size)sin(θ)cos(φ)
        C_y ← (voxel−size)sin(θ)sin(φ)
        C_z ← (voxel−size)cos(θ)
        while X_min ≤ X ≤ X_max & Y_min ≤ Y ≤ Y_max & Z_min ≤ Z ≤ Z_max do
            tagVoxelContainingPointWithDirection(X, Y, Z, VOX, O(θ, φ))
            X ← X + C_x
            Y ← Y + C_y
            Z ← Z + C_z
        endWhile
    endFor
endFor
```

5.1.2 Asymptotic complexity in determining visibility for roughing

We will make an estimate of the memory requirements for determining the "roughing" visibility cones for an object mesh with m triangles and n viewing directions. Following are the stages in the visibility determination:

Computing visibility matrix of surface segments: Section 3.6.4 outlined the memory requirements and number of computations needs for determining the visibility matrix of the surface segments.

Computing "roughing" visibility cones: The visibility matrix generated in the previous stage is used to determine the "roughing" visibility cones. The gaussian sphere is discretized up front and is held in memory. The model is also discretized into (n_x, n_y, n_z) voxels. For every voxel, we store the pointers to the gaussian triangles from which it is visible. In the worst case, the number of gaussian triangles in a visibility cone of a voxel can be n. If our implementation takes p bytes of data to store a pointer in memory, memory requirement on an average will be $O(n_x \times n_y \times n_z \times n)$. For a real world application; $n \approx 1280$, (n_x, n_y, n_z) $\approx$ (20, 20, 20) and $p \approx 8$. The memory requirement evaluates to around 82 megabytes. This is however the worst case estimate and the amortized memory requirement in our implementation was around 40 megabytes.

In our implementation there are 2 loops, the first loop goes over all the surface triangles and the second loop goes over all the view directions. For every visible direction in a surface triangle we iterate atmost the diagonal of the bounding box and this is given by $\sqrt{n_x^2 + n_y^2 + n_z^2}$. Therefore, the total number of computations will be $O(n \times m \times \sqrt{n_x^2 + n_y^2 + n_z^2})$.

The data generated by voxel visibility is five dimensional $(x, y, z, \theta, \varphi)$. To store this data in a ordinary data structure is expensive. A hierarchical data structure like a 5–d tree can be used. Currently, memory limitations demand that we use only a coarse voxel grid. As a future work, we can develop hierarchical data structures to process the information more efficiently.

5.2 Tool posture selection

Visibility analysis yields a set of cones of visibility for regions of the delta volume. The question we ask is which is the most effective direction from the point of view of access? In this section we are concerned with determining guaranteed legal access directions, or postures, from which a tool can access a given region within the delta volume without collision. Our strategy consists of two stages: cone thinning and tool profiling. Section 5.2.1 describes cone thinning procedure for determining the most "promising" direction from each visibility cone. The selected posture is then profiled for access using concepts discussed in Chapter 4.

5.2.1 Cone thinning

We argue that in the absence of any information about the cutting tool, the best access direction is in some sense the "center" of the visibility cone. The are several concepts of a center in geometry. One approach would be to find the "center of mass" or "centroid" of the cone. However, as shown in Figure 5.6, the centroid would not work because the centroid of a complex shape might lie outside the boundary of the shape.

A more appropriate "best direction" is the skeleton of the visibility patch on the Gaussian sphere. The skeleton may be obtained by thinning the tessellated representation of the visibility cone on the Gaussian Sphere. The thinning shown in Figure 5.7 (a) is similar to that used in com-

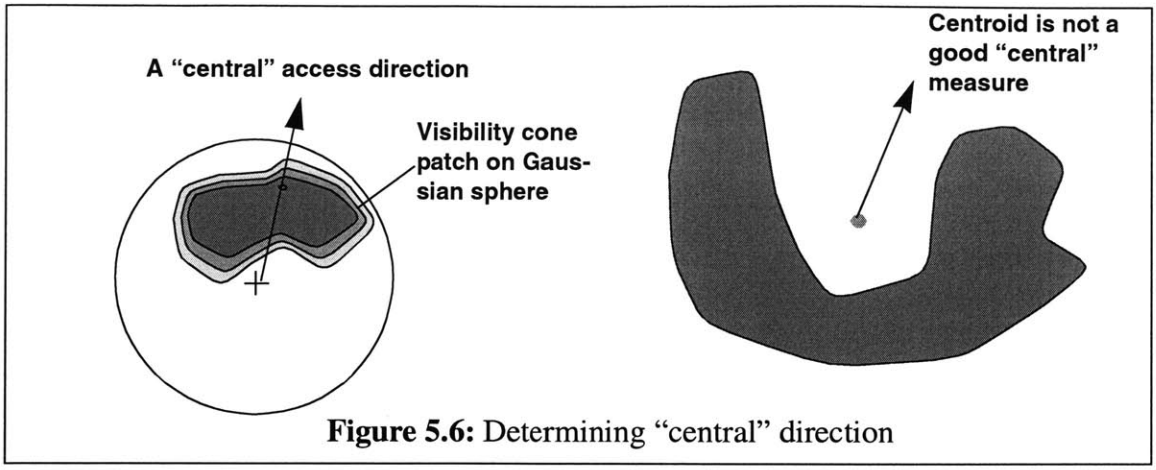


Figure 5.6: Determining "central" direction

puter vision applications with two important differences: firstly, we are using triangular rather than square cells, and secondly, the thinning is being carried out on the surface of a sphere. The algorithm consists of iterative thinning steps with a stopping condition. The iterative step consists of starting from the original visibility cone and shrinking the outer boundary inwards one layer at a time. We define a layer as all the triangles that contact the boundary. After each shrinking step we redefine the thinned boundary and repeat the process iteratively. We stop the iteration when the next step will reduce the shrinking region to an empty set. Figure 5.7 (b) shows the numbering of gaussian triangles on the visibility cone based on the "depth" from the boundary of the visibility cone. Figure 5.7 (b) also shows a snapshot of our implementation where the different sets of colors indicate the layers of the visibility cone during cone thinning. In practice, we can achieve this by continuing the iteration until the shrinking region goes to an empty set and then back up one step. The entity that remains at this stage is the skeleton center.

5.2.2 Accounting for machine limits — Restricted thinning

Machines have motion limits. In any setup, there are limits on the orientations that a 5–axis machine can achieve. For example, most 5–axis machines with trunnion tables have a 110° limit on α axis rotation. Similarly, most rotating heads have a limit of about 70° of β rotation. These limits must be considered during machining. We do so in the cone thinning stage as shown in Figure 5.8 using a process known as *restricted thinning*. Restricted thinning is performed on a primary contour, but restricted to a secondary contour. The steps in restricted thinning are identical to those

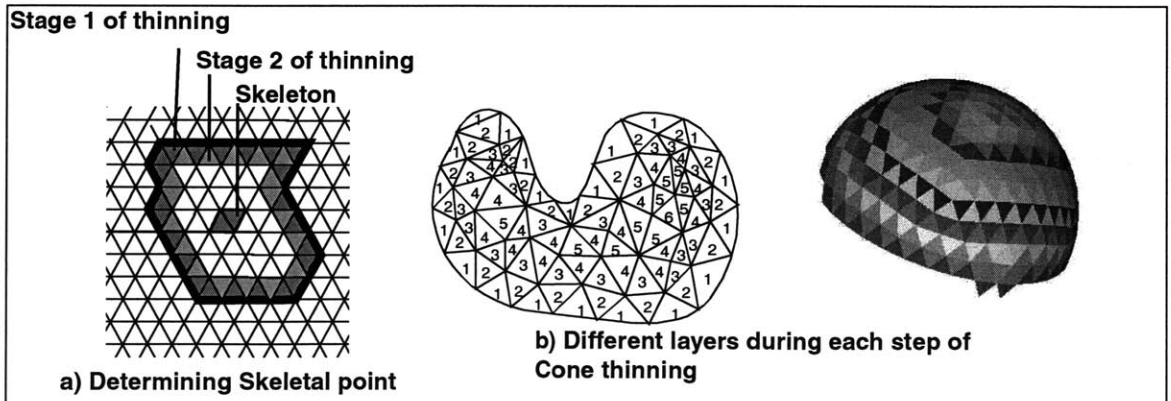


Figure 5.7: Discrete Cone thinning

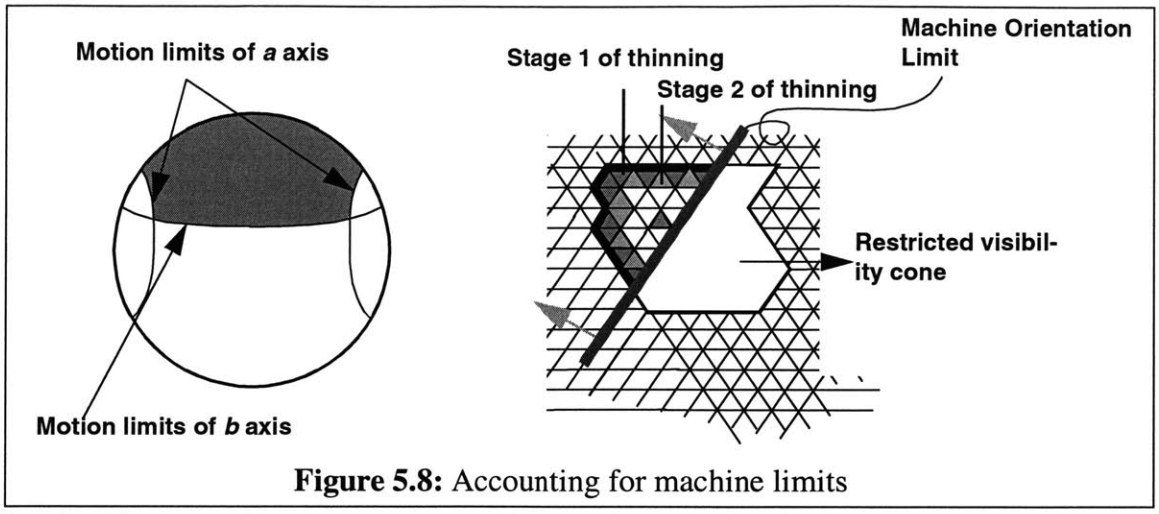


Figure 5.8: Accounting for machine limits

described in the section above. However, the stopping condition is different. Instead of stopping just before the shrinking region vanishes, we stop just before the intersection of the shrinking region with the secondary contour region vanishes.

5.2.3 Collision Proofing the posture

Chapter 4 describes the procedure for determining access using a pseudo–gradient search around the most "promising" access direction obtained from the visibility analysis. Our general strategy will be based on decomposing the delta volume into slabs perpendicular to the setup direction and generate for every slab. An important requirement for slab–wise machining is that no portion of the tool should penetrate the lower plane of the slab it is machining. This requirement places a constraint that the translational correction should be parallel to the plane. Figure 5.9 shows translational correction vectors (parallel to the slice plane) that would bring the point out of collision. The procedure for computing the rotational and composite correction for all the colliding points has been described in Chapter 4.

It should be noted that the iterative tool correction brings the tool out of collision, but we desire a tool access direction where the cutting portion of the tool is in contact with the part while the non–cutting portions are well off the surface of embedded part. This can be achieved in our implementation by simply over–sizing the non–cutting portions of the tool. Even the cutting por-

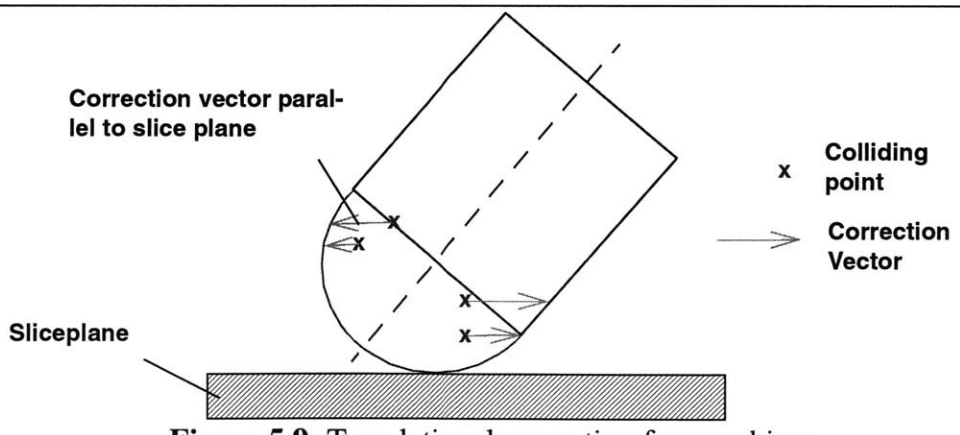


Figure 5.9: Translational correction for roughing

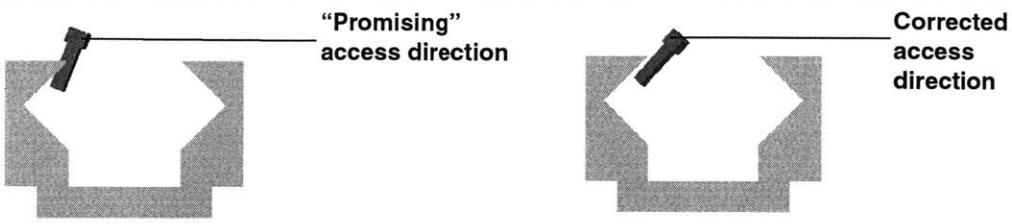


Figure 5.10: Access profiling a colliding tool postures

tion of the tool can be oversized during roughing to leave out some material to be removed during finishing. Figure 5.10 shows the process of collision proofing where a tool posture that resulted in collision has been corrected to a collision free access direction.

5.3 Generating sample points for assigning tool postures

Section 5.2 described the procedure to determine a collision free access posture for a point from a most "promising" orientation. All that remains is to traverse the three dimensional volume and determine the sequence of points that needs to be machined. There are several ways to traverse the three–dimensional volume of the delta volume. Our general strategy will be based on decomposing the delta volume into slabs perpendicular to the setup direction and generating zigzag or Voronoi based area filling paths [Held 1991]. Figure 5.11 (a) shows a slice through the object and the voxels that are intersected by the slice are shown in Figure 5.11 (b). Figure 5.11 (b) also shows the visibility cones and the "promising" access directions for every voxel in the slice.

We generate a roughing pattern that removes material from the outer layers inwards so that the delta volume in the upper slab is removed completely before proceeding to the next slab. There can be several discontinuities in the accessibility space of the voxels at a slab. The discontinuities can be in the form of inaccessible (dead) voxels or due to absence of a smooth or direct transition between neighboring voxels. We will talk more about identification of the discontinuities in Chapter 9. We then fill the slice with zigzag toolpaths such that the dead voxels and the discontinuities are avoided. This is a fairly straight forward task with parallels in scan conversion and we do not enter into detail here. As the tool tip proceeds along this path, our task will be to assign an orientations to the cutting tool in the 4^{th} and 5^{th} axis such that it does not intersect with the embedded design. Figure 5.12 shows a zigzag roughing pattern on a slice along with discontinuities due to dead voxels and transitions.

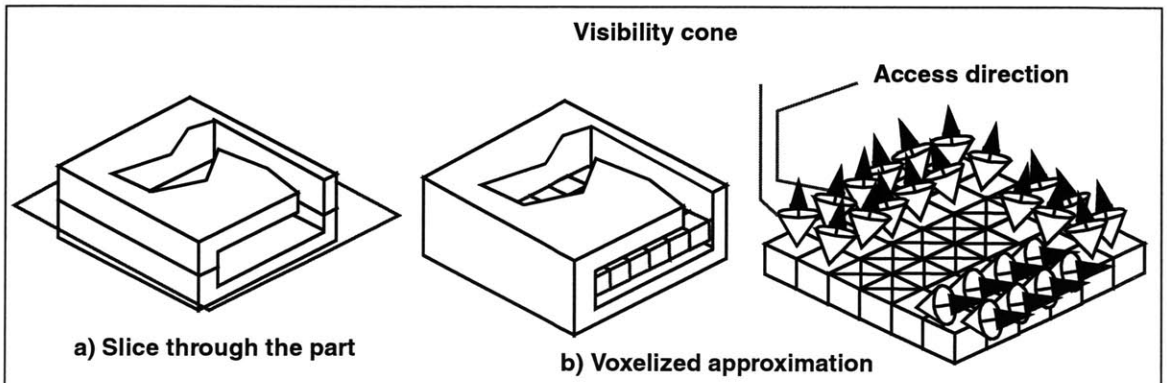
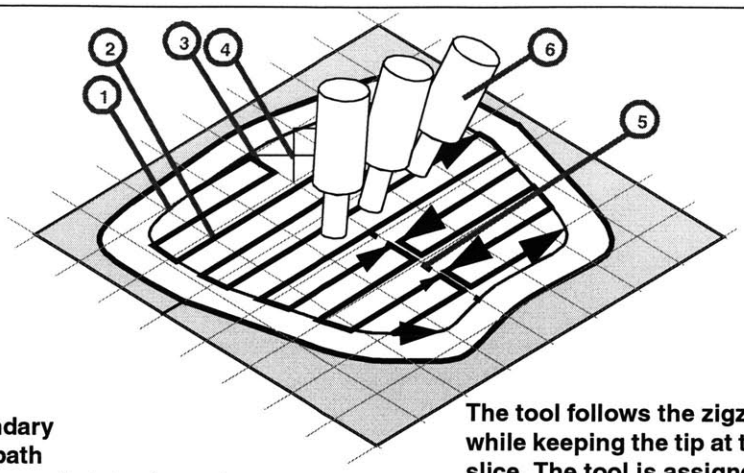


Figure 5.11: Access information at a slice



1. Offset Boundary
2. Zigzag toolpath
3. Direction reversal at dead voxel
4. Dead Voxel
5. Direction change at a discontinuity in the configuration space

The tool follows the zigzag toolpath while keeping the tip at the base of the slice. The tool is assigned the "most promising" orientation from the visibility analysis.

Figure 5.12: Sampling the slice for roughing

5.4 Tool posture interpolation

The set of *all possible* legal postures of the tool is referred to as the configuration space and the postures we have computed at the points in the delta volume is certain to lie within this configuration space. What we seek now is a toolpath which sweeps the entire delta volume without leaving the configuration space. Our approach will be to do so by interpolating between the legal postures we have computed. Interpolation of position coordinates between postures can be performed in a straight–forward fashion using linear or spline interpolation. Similarly, orientations represented as quaternions can be interpolated by using quaternion algebra. Chapter 7 describes the procedure for spline interpolation of position and orientation of the tool postures [McCool 98].

Although the legal postures computed from visibility and accessibility do form a promising basis from which a toolpath can be interpolated, it is not necessary that the interpolated path is itself completely legal. It is infact possible that the interpolated path leaves the configuration space. An example to illustrate the problem is shown in Figure 5.13, where the overhang in the middle of the pocket divides the configuration space into two portions. The two portions can the viewed broadly as the region that can be accessed from the left and that which can be accessed from the right. The free configuration space is shown on the right as a region on a cylinder which is a cartesian product of Euclidean 1–space R and a circle S. The cylinder $S \times R$ represents the possible orientations θ (which correspond to access arcs, the analog of access cones in 2D) of the tool as the tip of the tool is translated along the x axis in some slice as shown. The free configuration space is shown in grey. Our representation here can be viewed as a 2D slice through the more general 5D configuration space. Chapter 7 addresses the various issues associated with interpolation and describes a procedure for determining a valid interpolation using the visibility cones of the end points and access profiling.

Section 8.1 presents several figures to illustrate the concept of access based roughing toolpath generation. Illustrations are provided for every stage of the toolpath generation process, namely

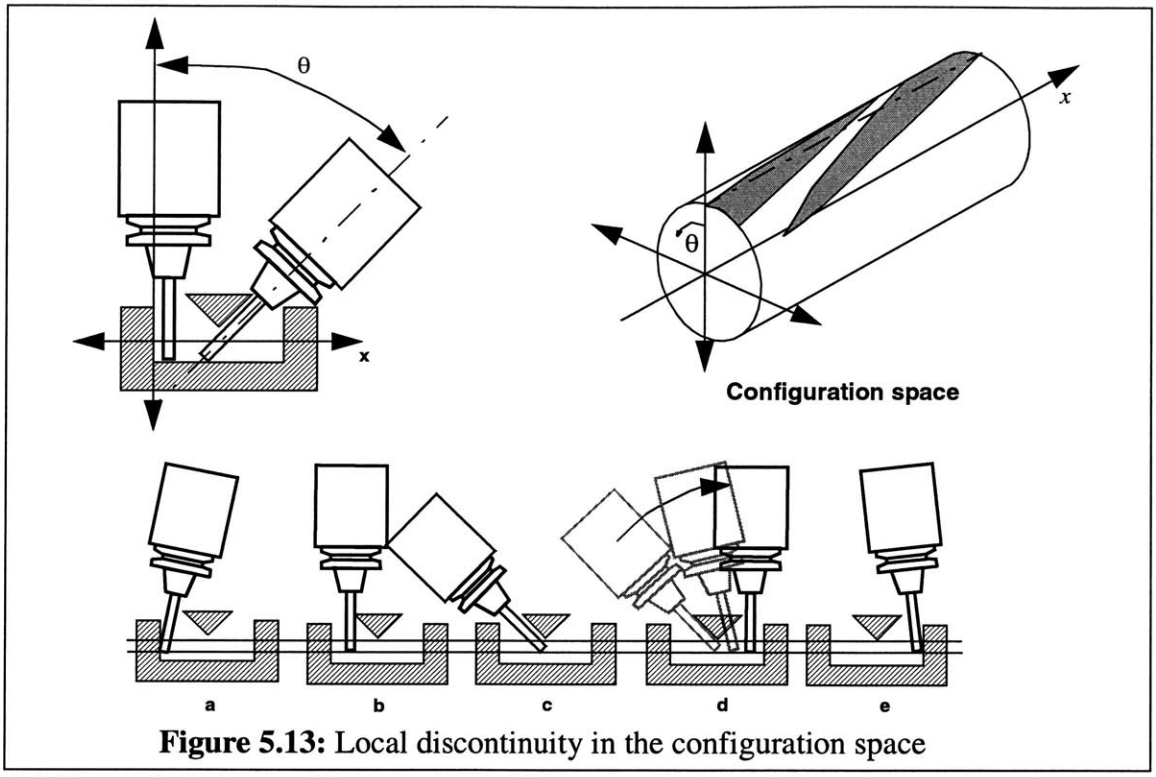


Figure 5.13: Local discontinuity in the configuration space

visibility analysis for roughing, access profiling, numerical simulation of toolpaths and manufacturing.

Chapter 6: Toolpath generation for surface finishing

Free form surface machining is an important operation in the automotive, airplane and ship building industries. Suh [Suh 95] suggest that multi–axis NC machining is key to cope with the industrial demands for geometric variety and high precision. Lee [Lee 97] suggests that the surface finishing of dies in the aerospace industry can take as long as tens to hundreds of hours to machine a free–form part from a block. The surface of the part is usually created by a delicate machining process called finish machining. Finish machining is very critical to achieve the tolerances that determine the functional characteristics of a part. In many cases, it may take weeks to generate NC toolpaths and the main obstacle for generating the toolpaths automatically is the absence of interrogation tools that help in determining the access direction of the tool at various points on the surface of the object. The problem of access determination gets complicated because the tool model can be generic to include the cutting and non–cutting portions such as the tool holder and spindle, while the workpiece model can include fixtures and other obstructions that form a part of the machining environment.

In this chapter, we propose a modular system to determine the accessibility information of tools rapidly. The access determination methodology can be integrated with existing toolpath generation schemes to generate global and local gouge free five axis finishing toolpaths for a free–form surface. The "heuristics" will generate toolpaths that are guaranteed to be collision–free between the workpiece and tool model.

6.1 Background of existing methodology

The problem of determining the approach directions for the tool has received considerable attention from the research community. Tseng [Tseng 91] proposes a computationally efficient method for determining feasible tool approach directions for a bezier surfaces only. [Elber 94, Woo 94, Suh 95] propose the concept of visibility cones and computational algorithms to determine the cones for an object. The information content in the visibility cone of an object has evoked a lot of interest in the researching community. The visibility cones were used to determine the minimum number of setups to machine an object. [Morishige 99] proposes a method for determining the 3–dimensional C–Space of an object. This 3D C–Space can be used to assign access direction for a tool tracing a particular path. The approach is however limited to ball end mills and the does not consider the effect of collisions between non–cutting portions of the tool and workpiece. [Choi 93] proposes a method for determining tool access direction using the concept of cutter–orientation maps, However these maps consider the effect of local gouging and do not take into account the global tool interference. Lee [Lee 95] proposes an interesting and a fast method for avoiding global tool interference in 5–axis machining in 2 phases. The method checks for collisions between the tool and the convex polytopes of the workpiece. However, the method was limited to NURB representation of the surfaces. [Vickers 89, Li 94] have given the advantages of using a flat–end mill for 5–axis surface machining and have also presented algorithms to determine 5–axis toolpaths devoid of local gouging. [Choi 98] gives a compendium of approaches from several publications in determining the CC points, step–length and path interval for the toolpaths. In this paper, we would like to propose an algorithm that would aid the determination of global and local collision free tool access directions. This can be integrated with the existing methodology of generating CC points described in [Choi 98] to get 5–axis surface finishing toolpaths automati-

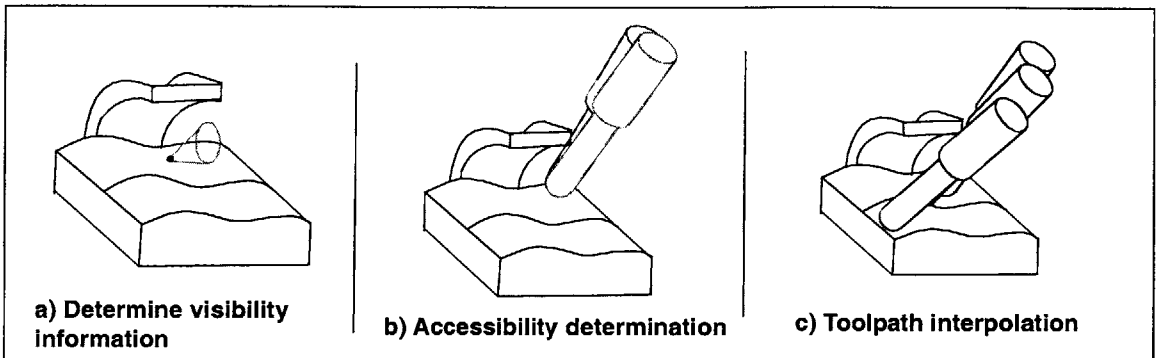


Figure 6.1: Stages in our approach for generating surface finishing toolpaths

cally. More background information will be provided at the Sections where the concepts are being introduced.

6.2 Our approach

Figure 6.1 shows the three stages in our approach: **visibility computation**, **legal posture definition** and **path interpolation**. In the first stage, we use the concept of visibility to determine from which directions a point in the delta volume is likely to be accessible to an observer located outside the convex envelope of the object. Section 6.3 illustrates the procedure for computing the visibility cones for a tessellated object, whose triangles define the segments on the surface of the object. These visibility cones provide us with "global" access information about the workpiece environment. In the second stage, we use the visibility information to determine the posture of the tool for the cutting operation. This is not a trivial step because, we have to select a sequence of access directions that satisfies the machining strategy selected by the user. For example, the machining strategy could be to maintain "almost" a constant machining strip width [Lee 97, Choi 98] or the tool orientation should not change rapidly along a path or a combination of above strategies. Section 6.4 presents heuristics to select a sequence of access directions from the visibility cones for a tool tracing a path based on the machining strategy. Moreover, the visibility does not guarantee accessibility because visibility checks whether a ray of light can reach a surface segment, whereas accessibility requires a tool of finite diameter to reach a surface segment. Therefore, we correct locally for collisions between the tool and the workpiece by using a very fast collision detection algorithm. In the third stage, we present algorithms to interpolate smoothly between collision free access directions and ensure that there are no collisions between the tool and the object at the interpolated orientations. The collision–free interpolation scheme has been discussed in detail in Section 6.6. The issues involved in developing a robust implementation have been described in Section 6.7.

Therefore, we are presenting a set of heuristics and "interrogating" tools that provide the toolpath generation routine with collision free tool postures for accessing the surface segments of the object. These algorithms can be seamlessly integrated with the toolpath generation routines formulated by previous researchers to develop a system that is capable of generating surface finishing toolpaths automatically.

6.3 Visibility analysis

Chapter 3 described the procedure for determining the visibility cones for points on the surface

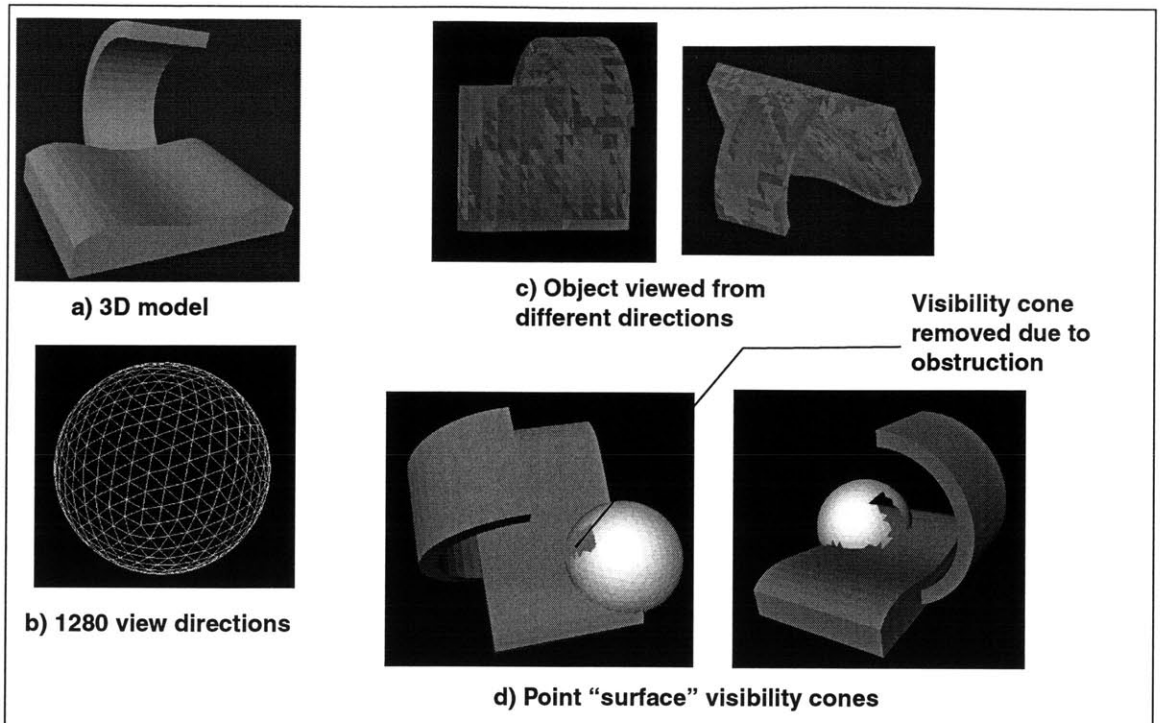


Figure 6.2: Visibility analysis of part with overhang

of the object. Figure 6.2 (a) shows the 3D model of the object for which the visibility analysis needs to be performed. The 3D model of the object includes an overhang, which in reality could be an obstruction due to a fixture or some portion of the object. Figure 6.2 (b), shows a gaussian sphere discretized into 1280 triangles, whose centroids determine the view directions along which we have to perform the visibility analysis. Figure 6.2 (c), shows the part as viewed from several directions. The discrete visibility cone is constructed by combining all the gaussian triangles from which the surface segment was visible. The overhang will obstruct the visibility of surface segments that are lying in its shadow along a view direction. This is evident from the Figure 6.2 (d) where a portion of an "otherwise" hemispherical visibility cone has been removed because of the presence of the overhang.

6.4 Tool posture selection

In this Section we will present heuristics for selecting a "promising" tool posture from the visibility information and justify the validity of the heuristic. The visibility cone suggests many approach directions, but we are interested in selecting only one access direction. Since surface finishing is a very delicate operation, it is desirable to give the control over methodology of orienta-

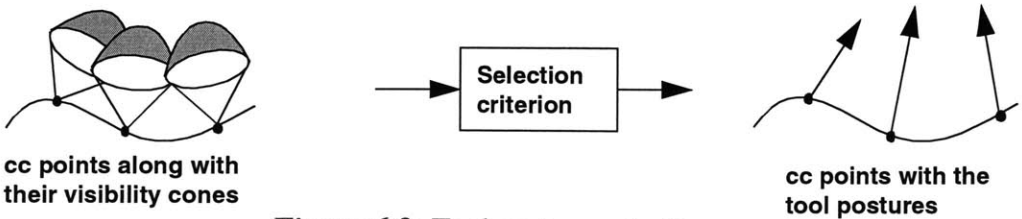


Figure 6.3: Tool posture selection

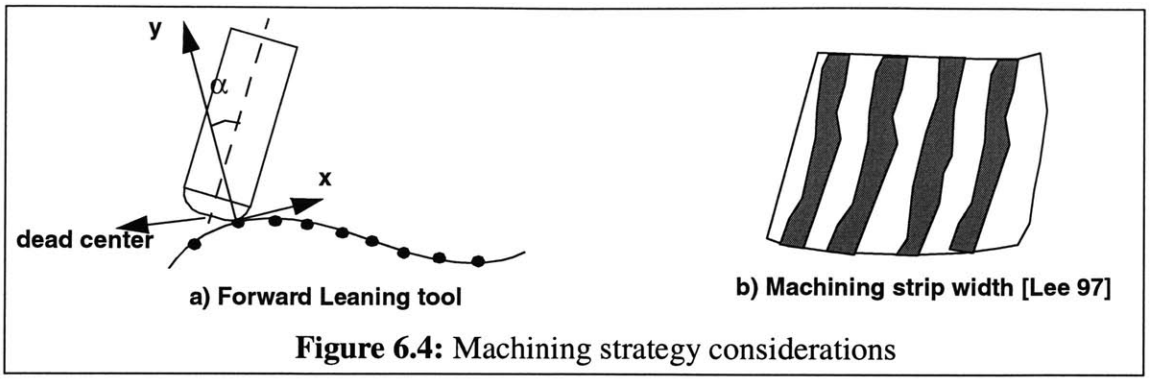


Figure 6.4: Machining strategy considerations

tion selection to the user. The user can specify a machining strategy which can be mapped into a selection criterion. Figure 6.3 shows a schematic of selecting access directions from the visibility cones for a set of cutter contact points based on an user defined selection criterion. Section 6.4.1 lists some of the commonly used machining strategies for five axis finishing.

6.4.1 Common machining strategies

Forward Leaning tool: Figure 6.4 (a) shows the dead center [Choi 98] of a tool where the cutting velocity is zero. Inorder to prevent the tool from dragging along the surface instead of cutting, it is imperative to avoid the dead center to come in contact with the surface. This can be implemented by specifying a constraint on the value of α (forward leaning angle) in the local coordinate system defined by the feed direction and surface normal at the cutter contact point [Lee 96].

Constant Machining Strip Width: Lee [Lee 97] describes a procedure to evaluate the machining strip width for 5–axis machining. Figure 6.4 (b) shows a surface and its machining strips. The machining strip width is used in conjunction with the cusp height to determine the side–stepping distance for generating cutter contact points. For a faster coverage of a surface it is desirable to maintain a constant machining strip width. If not, then we have to conservatively take the minimum machining strip width for the previous set of cutter contact points to determine the side–stepping distance.

Minimize the jump in orientations: Surface finishing is a very delicate operation and it is therefore desirable to minimize the jumps between the tool orientations at adjacent cutter contact points.

In this thesis, we have adopted a combination of the machining strategies. We will try to minimize the jump in orientations while trying to maintain a constant machining strip width.

6.4.2 Initial attempts on determining "promising" access direction

Initially, we tried to use the "cone thinning" heuristic described in Chapter 5, for selecting the most promising access direction from the visibility cone. However, this procedure had the following drawbacks:

1. *Dependence on Machining Strategy*: This procedure does not give the user any control over selection of access direction based on a machining strategy.
2. *Jumping of Orientations*: Since we are performing the cone thinning in a discretized space, the orientations selected would vary a lot for adjacent cutter contact points. As a result of

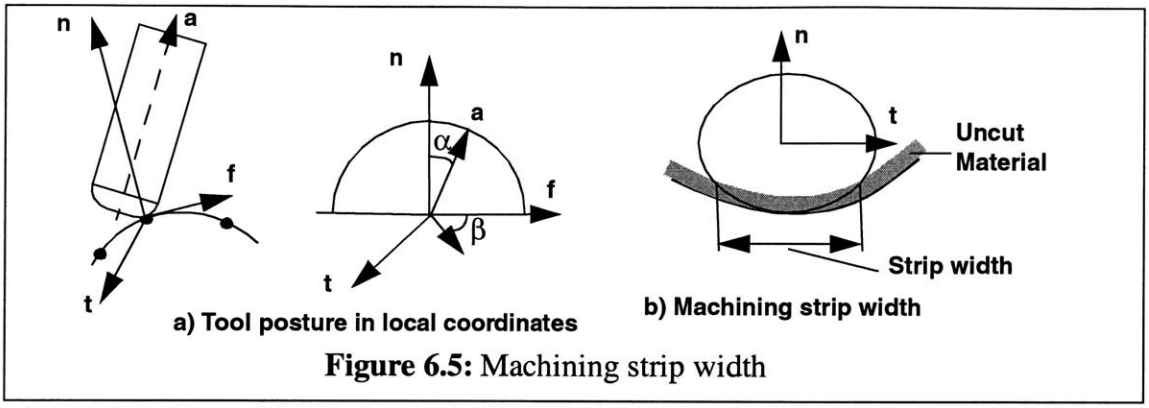


Figure 6.5: Machining strip width

which we obtained an unacceptable sequence of tool posture assignments. The jumping in the orientations can be reduced by increasing the number of gaussian triangles on the sphere, but this would increase the memory requirement and the number of computations. Therefore, it would be ideal to have the access extraction procedure that is independent of the discretization in the gaussian sphere.

The cone thinning heuristic was modified to reduce the "jump" between orientations by adding planar constraints based on the feed direction. Though the planar constraints allowed for specification of an upper bound for the orientation jump, it still did not give a handle to select access directions based on a machining strategy. Moreover, addition of planar constraint reduces the solution search space and this "greedy" approach may return an empty solution set even when there is a feasible optimal solution.

6.4.3 History information for selecting orientation

Inorder to determine a "promising" access direction from the visibility cone based on a machining strategy, we have to study the effect of tool inclination angle on the machining strategy. Moreover, it would be ideal to have a selection "heuristic" that is not affected by the discretization of the gaussian sphere.

The dependence of elliptical cutting shape on tool inclination has been elaborately studied by Lee [Lee 97] and Choi [Choi 98]. Figure 6.5 (a) shows a local coordinate frame defined at the cutter contact point. The *local–x* axis is along the feed direction (*f*) of the tool, the *local–y* is along the surface normal (*n*) at the cutter contact point and *local–z* axis is determined by $f \times n$. The angle α

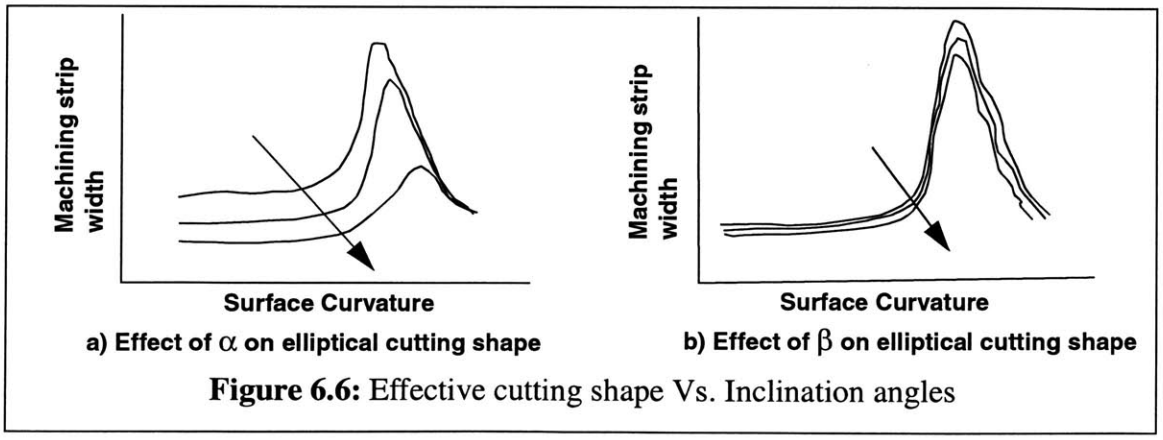
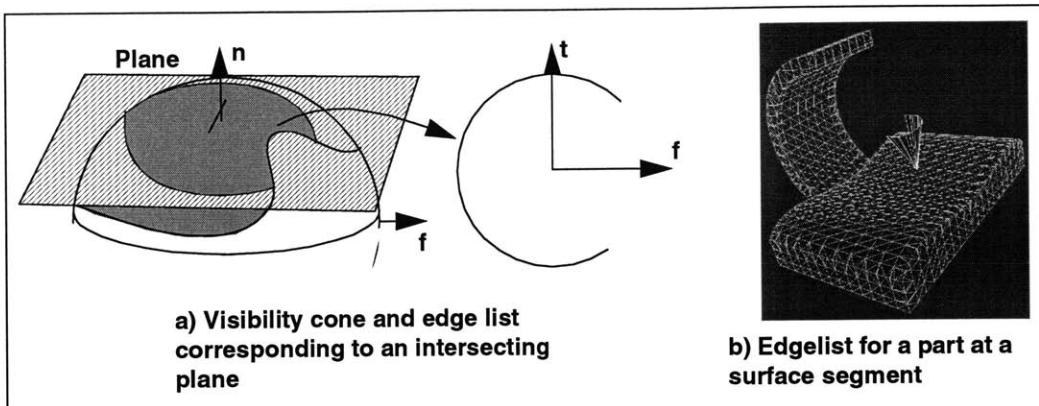


Figure 6.6: Effective cutting shape Vs. Inclination angles



a) Visibility cone and edge list corresponding to an intersecting plane

b) Edgelist for a part at a surface segment

Figure 6.7: Edgelist determination for a visibility cone

is the forward leaning of the tool with respect to the surface normal (*local–y*) and angle β is the out–of–plane orientation of the tool in the *local* x–z plane. The amount of material removed by the tool can be determined by considering the cross–section of the tool in the *local* y–z plane and the thickness of the uncut material. Figure 6.5 (b) shows an elliptical cross–section of the tool in the local y–z plane along with the thickness of uncut material. Lee [Lee 97] shows that the elliptical cutting shape is largely dependent on α and less dependent on β. For the sake of completeness, we show the trend curves obtained by Lee [Lee 97] on the dependence of cutting shape on inclination angles in Figure 6.6. Inorder to maintain a constant machining strip width, it is prudent to first search for probable candidate orientations that have the same α and then continue the search in β to correct for collisions.

In the local coordinate reference frame, the set of valid orientations with a constant α will lie on the visibility cone and a plane parallel to *local* x–z at a height $\cos(\alpha)$. The intersection of a discrete visibility cone with the plane gives an "edgelist" that represents tool inclinations that result in "almost" same machining strip width. By selecting an access direction from this edgelist, we can achieve almost achieve a constant machining strip width. Figure 6.7 (a) shows a schematic for determining an "edgelist" by intersecting a plane with the visibility cone. Figure 6.7 (b) shows the constant α "edgelist" for a point on the surface of a 3D object. It is clearly seen that the infeasible access directions due to the presence of the overhang have been pruned off the "edgelist". The "edgelist" determination step is just an one time operation that depends on the surface normal at the point. The "selection" heuristic will use this "edgelist" to determine a sequence of postures such that the jump between orientations at adjacent cutter points is minimized.

A "global" orientation vector on the edgelist ($\vec{a}_i$) corresponding to a forward leaning angle α can be written in terms of local coordinates as:

$$\vec{a}_i = \cos(\alpha)\vec{n} + \lambda_i\vec{f} + \mu_i\vec{t}, \text{ where} \tag{6.1}$$

$$\lambda_i^2 + \mu_i^2 = (\sin(\alpha))^2 \tag{6.2}$$

For a well–behaved surface, we can show that minimizing the orientation jumps in the local coordinates of the tool would result in minimizing the orientation jumps in the global coordinates. We can frame a selection heuristic that selects a $(\lambda_i, \mu_i)_{current}$ from a $(\lambda, \mu)_{prev}$ corresponding to the posture at a previous cutter contact point. A simple and a good selection heuristic can be based on

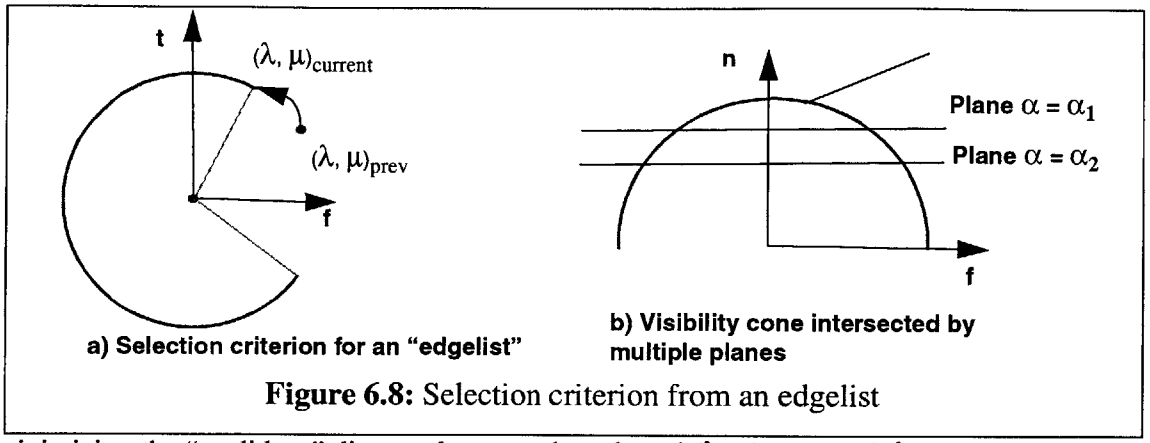


Figure 6.8: Selection criterion from an edgelist

minimizing the "euclidean" distance between the selected $(\lambda, \mu)_{current}$ and $(\lambda, \mu)_{prev}$. The heuristic for selecting a $(\lambda, \mu)_{current}$ that minimizes its distance from $(\lambda, \mu)_{prev}$ is shown in Figure 6.8 (a). The user can implement any selection criterion that fits the machining requirements. The idea of minimizing the distance in (λ, μ) can be extended to include selection from a set of edgelists corresponding to several forward leaning angles $(\alpha_1, \alpha_2, .., \alpha_n)$. Figure 6.8 (b) shows a visibility cone intersected by planes corresponding to different α's. The selection criterion has to be framed carefully taking into account the trade–off between changing α and β on the effective cutting shape of the tool.

Figure 6.9 illustrates the procedure for assigning "promising" access orientations from the "edgelist". Figure 6.9 (a) shows a sequence of cutter contact points along the feed direction of the tool on a 3D object. Figure 6.9 (b) shows the points along with the "edgelist" corresponding to an inclination angle α. The access selection process is initiated by taking a valid $(\lambda, \mu)_1$ and then marching forward to determine $(\lambda, \mu)_i$ based on the minimum distance heuristic. For the purpose of illustration, a smooth curve passing through the selected $(\lambda, \mu)_i$ has been drawn. The selected (λ, μ) for the four cutter contact points have been marked on their respective edgelist. Figure 6.9 (c) shows the selected tool postures at the cutter contact point. The tool initially starts out with an inclination towards the overhang, as it nears the overhang the tool changes its orientation by vary-

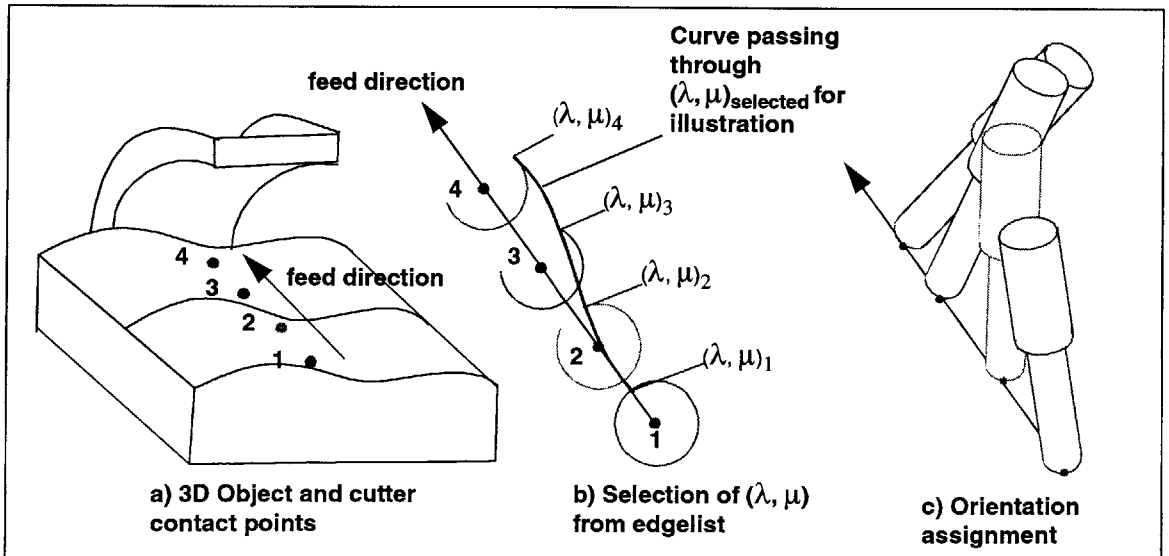


Figure 6.9: Orientation assignment using minimum distance heuristic

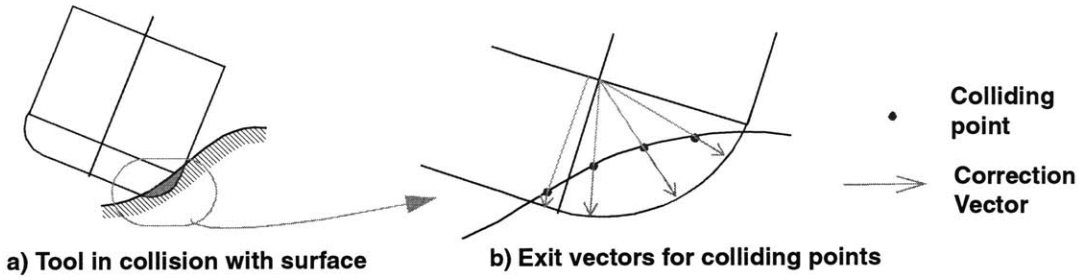


Figure 6.10: 3D Translational correction

ing the β angle so that it will be able to reach under the overhang without colliding with the overhang.

6.4.4 Collision proofing the posture

Chapter 4 describes the procedure for determining access by using a pseudo–gradient search around the most promising access direction obtained from the visibility analysis. The translational correction is performed when the collisions occur close to the cutter contact point where the rotational correction is ineffective. Unlike translational corrections in roughing, the tool can correct itself by performing a 3D translation. Figure 6.10 (a) shows a tool in collision with a surface and Figure 6.10 (b) shows the exit vectors associated with the colliding points for computing the composite translational correction. The procedure for computing the rotational and composite correction for all the colliding points has been described in Chapter 4. Figure 6.11 shows collision free access directions for 2 parts.

6.5 Generating sample cutter contact points

The inputs to the 5–axis finishing problem are: the surface to be manufactured and the accuracy requirements [Choi 98] of the machined part. The requirements will be specified by the following parameters: in–tolerance (gouge), out–tolerance (uncut) and maximum allowable cusp height that are permissible for the final part surface. The spacing between adjacent cutter points that are lying in the same feed direction is determined by the in–tolerance, out–tolerance parameters and local curvature of the surface. Figure 6.12 (a) shows two cutter contact points and the intermediate positions are interpolated by linear interpolation. The permissible amount of material clipped by the tool as a result of this interpolation is specified by in–tolerance. Similarly, there is a specification for out–tolerance where the intermediate positions of the tool between 2 cutter contact points leaves some uncut material on the surface. The spacing between cutter points in a direc-

Figure 6.11: Collision free access direction

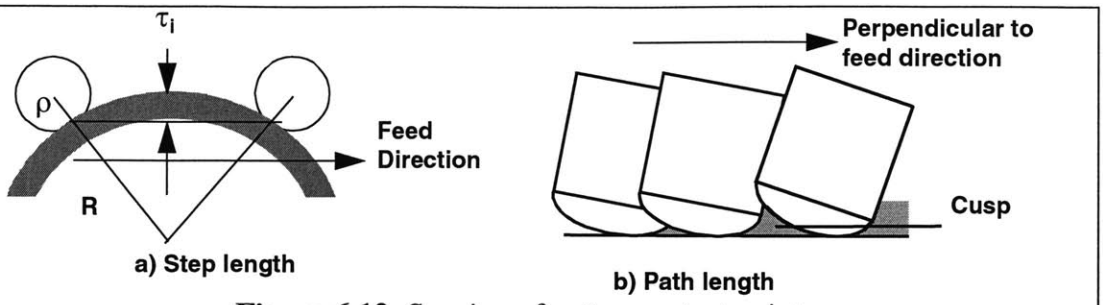


Figure 6.12: Spacing of cutter contact points

tion perpendicular to the feed direction is determined by the maximum allowable cusp height, shape of the cutting portion of the tool and local surface curvature. Figure 6.12 (b) shows 3 instances of tool that are moving into the plane of the paper and the uncut material that remains on the surface inbetween 2 passes of the tool is specified by the allowable cusp height. Choi [Choi 98] gives a compendium of approaches proposed by several researchers for determining the cutter contact points on surface of the part. Our aim is to integrate our modular tool posture determination routine from the visibility information with the existing toolpath generation procedures.

There are 3 ways of generating cutter contact points: iso–parametric marching, guide plane marching and APT method of using drive planes. Figure 6.13 (a) shows the iso–parametric approach where we move along an iso–parameter line and determine the sequence of cutter contact points. Figure 6.13 (b) shows the guide plane approach which is a variant of the iso–parametric approach. The surface can be projected to the guide plane so that there is a unique point on the surface for every point in the guide plane. We then select a marching direction in the guide plane and generate a sequence of cutter contact points. Figure 6.13 (c) shows the APT method of cutter contact point generation. A drive plane intersects the surface and the cutter contact points are selected along the intersection curve. The step–length between adjacent cutter contact points are determined based on the in–tolerance and out–tolerance parameters. The spacing between drive planes can be determined from the machining strip width and maximum allowable cusp height. We have used the APT method of cutter contact point generation in our implementation.

6.6 Tool posture interpolation

The set of *all possible* legal postures of the tool is referred to as the configuration space, and

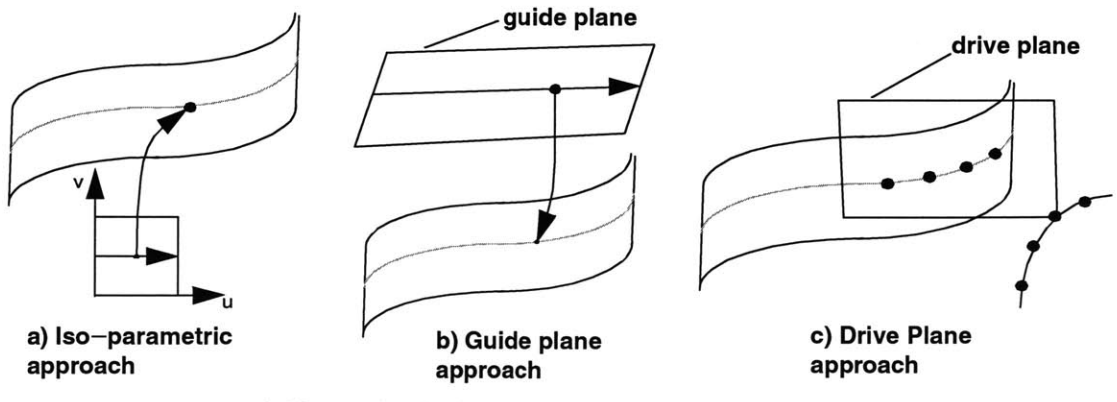


Figure 6.13: Methods for generating cutter contact points

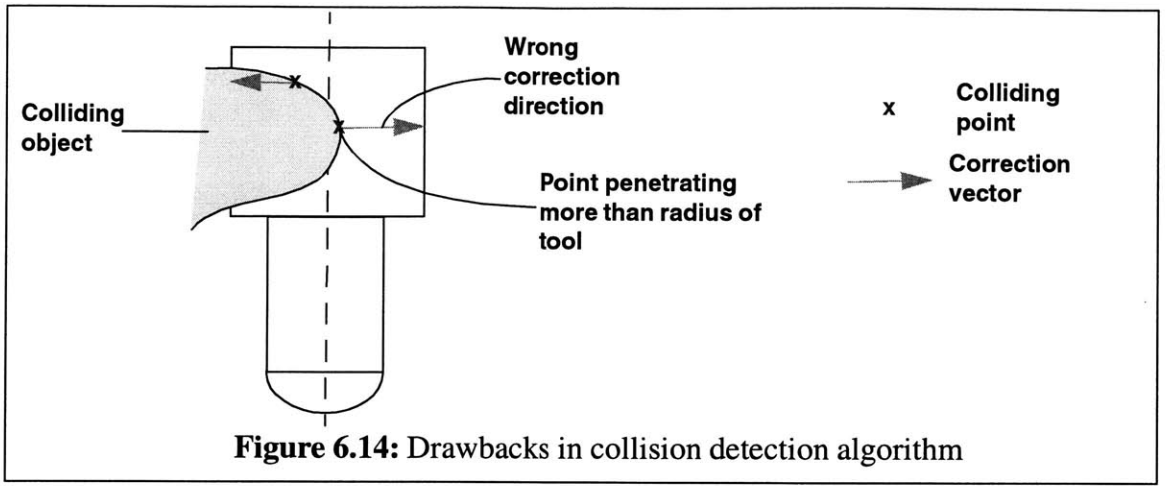


Figure 6.14: Drawbacks in collision detection algorithm

the postures we have computed at the points on the surface is certain to lie within this configuration space. What we seek now is a toolpath which sweeps the entire surface without leaving the configuration space. We can compute the intermediate tool postures by interpolating the legal postures computed at the end points.

Section 5.4 gave the reasons for why this problem is not straight forward. Chapter 7 addresses the issues associated with interpolation and describes a procedure for determining a valid interpolation using the visibility cones of the end points and access profiling.

6.7 Implementation issues

Chapter 8 shows several examples to illustrate the procedure for generating surface finishing toolpaths. We faced some interesting problems during the implementation of the system due to the limitations in the component subroutines. Section 6.7.1 describes a complication due to the collision detection algorithm and how we overcame it on our implementation. Section 6.7.2 describes the dependence of toolpath generator on setup orientation of the part.

6.7.1 Accommodating the deficiencies in collision detection

The collision detection routine used in our implementation [Ho 1999] modelled the tool as a symmetric solid. As a consequence of which points that have penetrated more than the radius of the tool will be interpreted incorrectly and corrected in a direction that will push it more into the solid. Figure 6.14 shows a colliding point and how it would be interpreted by the collision detection routine.

In surface finishing, it is possible to have collisions with penetration depth greater than radius of the tool. The access direction proposed by the visibility analysis assumes that a ray of light is accessing the point on the surface, therefore we can expect atmost half of the tool placed in that orientation to be in collision with the part. Moreover, while accessing a cutter contact point the center of the tool has to be translated by some distance to account for its geometry. Figure 6.15 shows a case when this translation results in penetrations greater than the radius of the tool. The additional penetration may result due to discrete nature of our approach.

On analyzing the reason for excessive penetrations, we found that the problem was mainly due

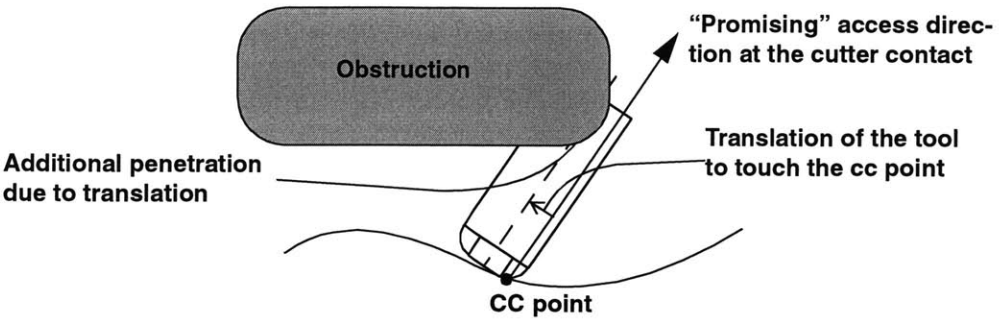


Figure 6.15: Reasons for larger penetration depth

to selection of orientation directions corresponding to the gaussian triangles at the boundary of the visibility cone. It would be ideal to select a promising orientation that is "far away" from the boundary. We can avoid this excessive penetration problem by removing a "few" outer layers of the visibility cone and then determining the promising orientation of the tool from the modified visibility cone. The heuristic for trimming the visibility cone is referred to as "Selective Cone Thinning". The number of layers to be trimmed can be estimated from the tool geometry and the angle subtended by a gaussian triangle. Figure 6.16 illustrates the procedure for determining the maximum angle subtended by the constituent cylinders at a common reference CL point. If $\delta_{gaussian}$ represents the angle subtended by a gaussian triangle, then the number of layers to trim is given by $\lceil \gamma_{max-subtended} / \delta_{gaussian} \rceil$. For the examples shown in Chapter 8, we have trimmed the visibility cone by 4 layers.

6.7.2 Dependence on Setup Orientation

The finishing tool postures assigned to a cutter contact point depends on its visibility cone, surface normal and feed direction. The point visibility cone is a characteristic of the part and is independent of the setup. The machine limits define a space (M) of possible orientations of the tool tip. The setup orientation of the part determines the exact location of the space M on the point visibility cone. Section 5.2 described the effect of machine limits on the determination of the roughing tool postures. The effect of machine limit can be incorporated in the posture assignment stage by pruning off the "edgelist" based on the "feasibility" of an orientation in machine space.

Section 8.2 presents several figures to illustrate the concept of access based finishing toolpath generation. Illustrations are provided for every stage of the toolpath generation process, namely visibility analysis, access profiling, numerical simulation of toolpaths and manufacturing. It

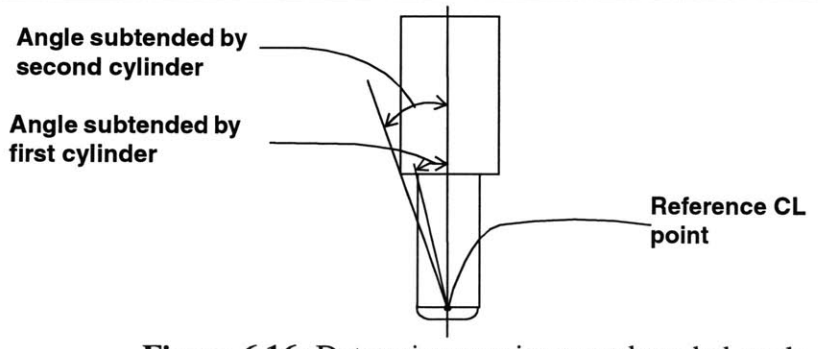


Figure 6.16: Determine maximum subtended angle

should be noted that the finishing toolpaths shown in the example does not consider the effect of machine limits.

Chapter 7: Toolpath interpolation

The heuristics for generating access based finishing and roughing toolpaths will determine the access orientations at discrete points in the delta volume or the part surface. The important aspect of toolpath generation that is yet to be addressed is the transition between valid tool postures. It is not obvious whether interpolating between valid tool postures will still be valid. Moreover, it is desired to have a smooth interpolation, so that the machine does not have several start–stop motions at the end of every toolpath segment. Therefore, we need an interpolation that is atleast C_1 continuous in both position and orientation.

Section 7.1 describes a method for generating valid interpolating tool postures. The interpolation of tool postures involve both translational and orientation interpolation. Section 7.2 describes the issues involved in interpolating the orientations and presents a cubic spline orientation interpolation scheme.

7.1 Validity of the interpolated orientations

It is not necessary for the tool postures interpolated from valid postures to be necessarily valid. Figure 7.1 (a) shows tool postures at 2 points on the surface along with their visibility cones. An obstruction has been introduced to illustrate that straightforward interpolation will result in an invalid tool posture. Figure 7.1 (b) shows the visibility cones of both the surface segments. It is clearly seen that direct interpolation from posture 1 to posture 2, results in orientations that are not lying in the visibility cone of the intermediate points. Therefore we have to interpolate through the common intersection region of both the visibility cones to ensure validity of the interpolated posture. In addition to checking the validity of the interpolated posture through visibility cones, the collision detection and correction algorithm should be invoked at every interpolated posture to ensure that it is collision free. The maximum distance between interpolated tool postures got by sub–sampling can be determined from the surface curvature, tool geometry and the amount of permissible collision. We can use the algorithms similar to the ones used for determining the step length between cutter contact points to estimate the maximum distance between interpolated tool postures.

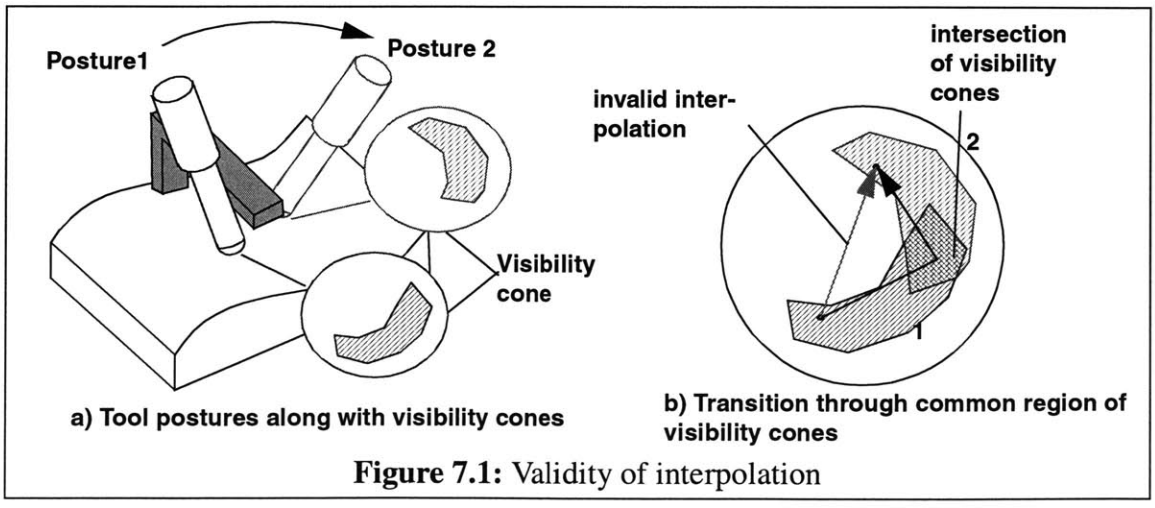


Figure 7.1: Validity of interpolation

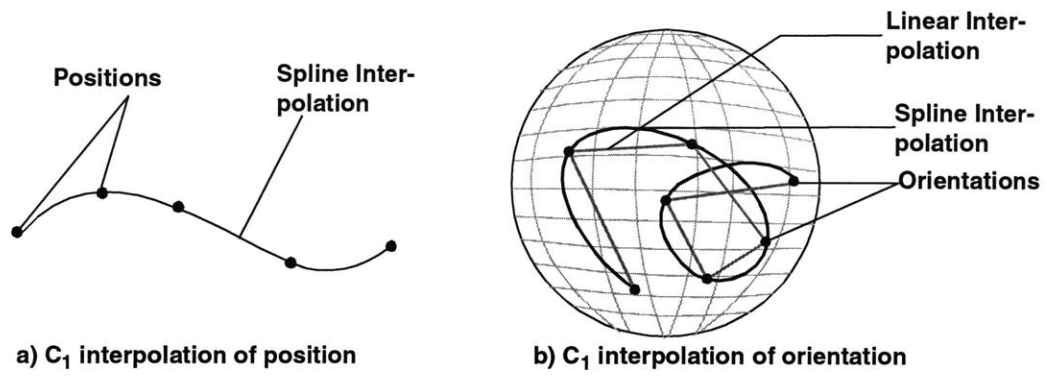


Figure 7.2: C_1 interpolation of position and orientation

The discontinuities in the configuration space of the object can make direct transitions between two successive tool postures infeasible. Under such circumstances, the tool has to be retracted and then re–inserted at the new location. Appendix C addresses the various issues associated with interpolating tool postures through a retraction and a re–insertion point.

7.2 Implementing cubic quaternion interpolation

C_1 interpolation of position coordinates is straightforward. Figure 7.2 (a) shows a set of points representing positions and an interpolatory cubic spline fit through the set of points. However, C_1 interpolation of orientation is trickier and has been studied extensively by several researchers. Figure 7.2 (b) shows points on a sphere (S_2) representing orientations and their interpolatory cubic spline. Shoemake [Shoemake 85] has suggested the use of quaternions to represent rotations and how it could be used in animations. [Kim 95, Horsch 98, Wilhelmy] suggests several schemes for interpolating orientations with higher order continuity.

Wilhemy [Wilhelmy] gives an insight into quaternion arithmetic and its similarity to position arithmetic. For the sake for completeness, we briefly describe his approach to determine a cubic spline interpolation of orientations using quaternions. A quaternion $q = (s, \vec{v})$ consists of a scalar and a 3D vector. The length of the quaternion can be determined by using dot product and all quaternions which represent a rotation have unit length. The quaternion interpolation formula can be used to find the shortest path from one quaternion to another on the surface of the unit sphere. The orientations at the interpolatory points are represented as quaternion rotations of a default vector (*z–axis*). We are interested in fitting an interpolatory quaternion spline through the set of quaternions. It would be useful to realize the equivalence of linear interpolation between two positions P_1 and P_2 with *SLERP* between two quaternions q_1 and q_2. We can use the SLERP function to interpolate between 2 orientations, in the same way as we draw a line between points.

$$P(t) = LinearInterpolation(P_1, P_2, t) = P_1 \times (1-t) + P_2 \times t \tag{7.1}$$

$$q(t) = SLERP(q_1, q_2, t) = \frac{\sin((1-t)\times\alpha)}{\sin(\alpha)} \times q_1 + \frac{\sin(t\times\alpha)}{\sin(\alpha)} \times q_2 \tag{7.2}$$

In the above equation t is the parameter and α is the angle between the quaternions. The

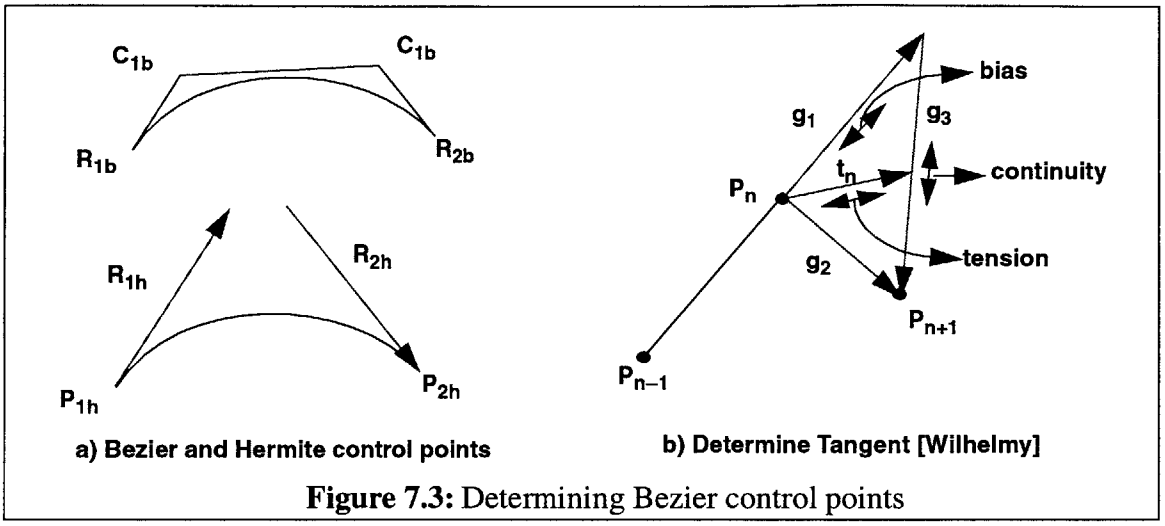


Figure 7.3: Determining Bezier control points

approach would be to determine the equations for position interpolation and then transform the results for interpolating the orientations.

A Bezier spline can be represented by 4 control points [Hoschek 93]: P_{1b}, C_{1b}, C_{2b} and P_{2b}. The Bezier spline passes through P_{1b} and P_{2b}, while C_{1b} and C_{2b} can be determined based on tangency conditions at P_{1b} and P_{2b}. A Hermite spline is a spline through two points P_{1h} and P_{2h} with given tangent vectors R_{1h} and R_{2h}. Figure 7.3 (a) shows a Bezier and a Hermite representation of a spline. Castaljieu suggested a method for evaluating the points on a Bezier spline by just using a sequence of linear interpolations. Since linear interpolation in position space maps to SLERP in rotation space, we can use the castaljieu method on quaternion Bezier control points to evaluate orientations on the interpolatory Bezier spline.

Our approach will be to determine the common tangent at the interpolatory point and then fit a Hermitian spline that satisfies the boundary conditions. We can use the equivalence between Hermite spline and Bezier spline to determine the Bezier control points C_{1b} and C_{2b}:

$$P_{1b} \Leftrightarrow P_{1h} \tag{7.3}$$

$$C_{1b} \Leftrightarrow P_{1h} + R_{1h}/3 \tag{7.4}$$

$$C_{2b} \Leftrightarrow P_{2h} - R_{2h}/3 \tag{7.5}$$

$$P_{2b} \Leftrightarrow P_{2h} \tag{7.6}$$

Determining common tangent: The tangent t_n at the interpolatory point P_n is determined as a linear combination of g_1 and g_3, where g_1 is the vector from P_{n-1} to P_n, g_2 is the vector from P_n to P_{n+1} and g_3 is a vector from g_1 to g_2. Figure 7.3 (b) shows the effect of tension, bias and continuity parameters in determining the tangents at the interpolatory point.

The above methodology can be easily extended to orientation space by just replacing the interpolatory points with quaternions representing rotation of a *z–axis* vector to the desired orientation vector. The Hermitian spline that satisfies the position and tangency boundary conditions is determined. The equivalence between Hermite and Bezier spline can be used to determine the quaternion Bezier control points. We can then evaluate the intermediate quaternions on the Bezier curve

at a parameter t by using the Castaljieu method. These intermediate quaternions can be applied on the *z–axis* vector to determine the interpolatory orientation vector at parameter t. In view of the brevity of description, we refer the readers to [Wilhelmy] for further implementation details.

Chapter 8: Examples

The preceding chapters explained the concept of generating access based roughing and finishing toolpaths. In this chapter, we will present the results of applying the concepts to various parts. The components were created using ACIS^{TM} geometric modeler and very fine tessellations were generated by using their Mesh generation routines. The total time for mesh generation varied from 3 – 10 seconds depending on the complexity of the model. Since the ACIS^{TM} mesh generation husk provides control over the aspect ratio of the triangle, the maximum length of an edge of a triangle and the maximum deviation between the true and the meshed surface — we can obtain a triangulation that is consistent with our assumptions. Section 8.1 presents four examples to demonstrate the various stages rouging toolpath generation. Section 8.2 presents two examples to demonstrate the various stages in finishing toolpath generation. For each example shape, we display graphical results of each salient step along with their execution times. We report real execution times on a SGI Octane Workstation with two 250 MHz R10000 processors as opposed to CPU time.

8.1 Roughing examples

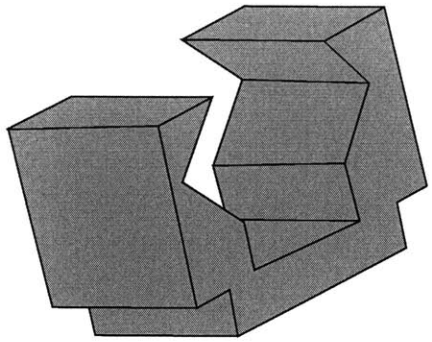
Section 8.1.1 shows the results for a simple object with an undercut, referred to here as an undercut channel. Section 8.1.2 shows the output of our system for a vase–like object consisting of a hollow frustum with a hemi–spherical shell bottom. Section 8.1.3 shows the results for an impeller blade component Finally, Section 8.1.4 shows the results for roughing a teacup in the presence of an handle.

8.1.1 Undercut channel

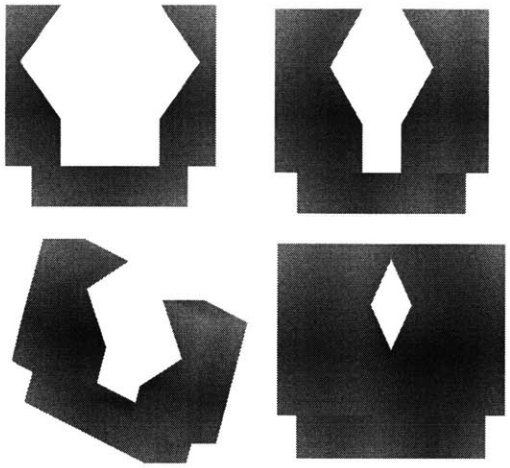
Figure 8.1 (a) shows a 3D model of the undercut channel which has been discretized into 1004 triangles. Figure 8.1 (b) shows the process of generating the visibility matrix by viewing the part from different orientations. This takes around 30 minutes for obtaining the visibility information along 1280 directions. Figure 8.1 (c) shows the "roughing" visibility cone for a point in the delta volume of the part. The process of determining the roughing visibility takes around 30 seconds for 1000 voxels. Figure 8.1 (d) shows the access profiling of a tool posture for machining a 2.5D slab. Figure 8.1 (e) shows the collision proofed tool postures at various points in the 2.5D slab.

8.1.2 Vase

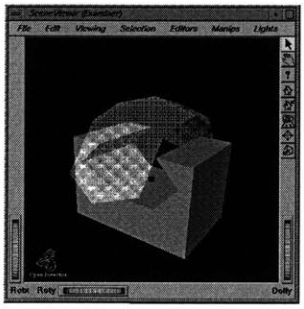
Figure 8.2 (a) shows a 3D model of the vase which has 2 portions: conical and a hemispherical frustum. We are interested in generating toolpaths to machine the conical portion of the vase. The part has been discretized into 2424 triangles. Figure 8.2 (b) shows the process of generating the visibility matrix by viewing the part from different orientations. This takes around 30 minutes for obtaining the visibility information along 1280 directions. Figure 8.2 (c) shows the "roughing" visibility cone for three voxel positions in the delta volume of the part. The process of determining



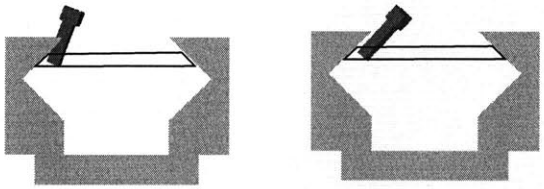
a) 3D model



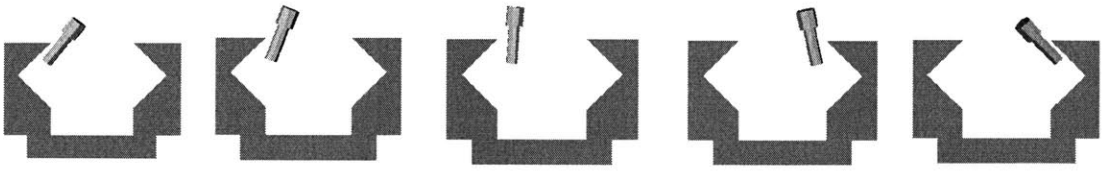
b) Performing Visibility Analysis to determine Visibility Matrix



c) "Roughing" Visibility cones

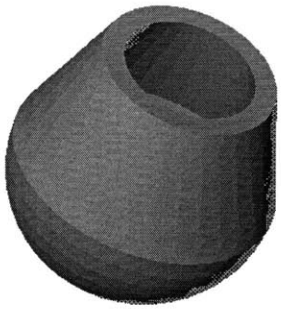


d) Tool access profiling

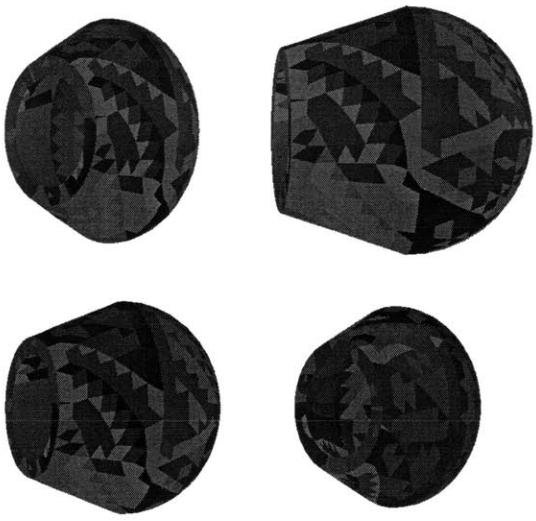


e) "Roughing" Tool postures at several points in a 2.5D layer

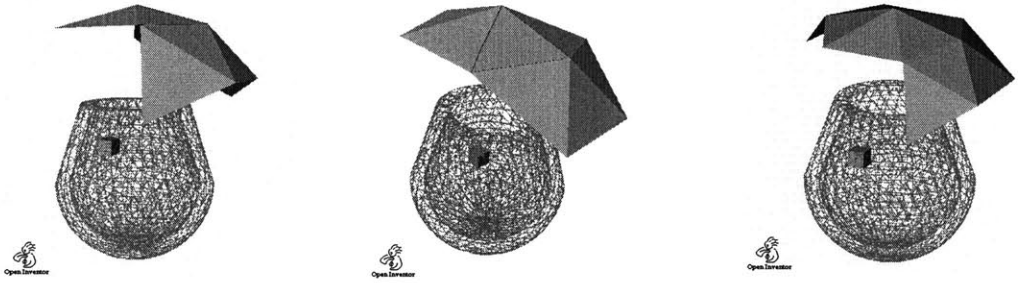
Figure 8.1: Stages in roughing toolpath generation for Undercut channel



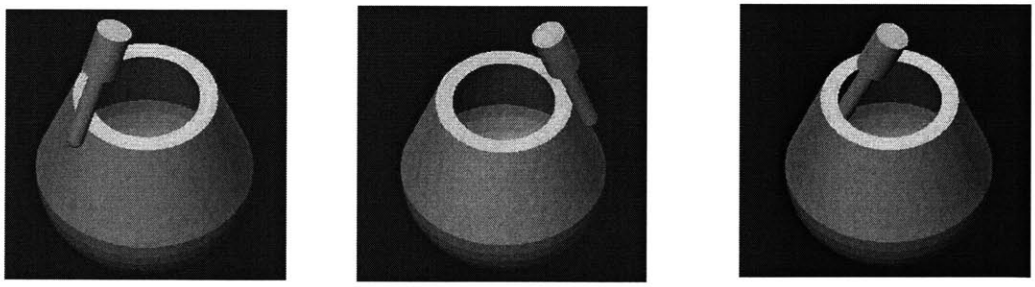
a) 3D model



b) Visibility Analysis



c) "Roughing" visibility cones



d) Tool postures at various points in the 2.5D slab

Figure 8.2: Stages in toolpath generation for vase

the roughing visibility takes around 45 seconds for 1000 voxels. Figure 8.2 (d) shows the collision proofed tool postures at various points in the 2.5D slab.

The part was "virtually" machined using the toolpaths generated by our algorithm. The "virtual" machining simulation was performed with NCVerify^{TM} [Sirius]. The input to the software is the 3D model of the stock, tool model and the toolpath. The program sweeps the tool through the toolpath and removes material corresponding to the tool sweep volume from the stock. Figure 8.3 (a) shows the part during various stages of machining. The toolpath was "posted" onto a 5 axis machine to manufacture an actual part. Figure 8.3 (c) shows snapshots of the "actual" machining process. Figure 8.3 (d) shows a snapshot of the part that remains after the machining.

8.1.3 Impeller blade

Figure 8.4 (a) shows a 3D model of an impeller which has 6 vanes. We are interested in generating roughing toolpaths to remove material from a cubical stock. The 3D model of the vane has been discretized into 10,665 triangles. Figure 8.4 (b) shows the process of generating the visibility matrix by view the part from different orientations. This takes around 35 minutes for obtaining the visibility information along 1280 directions. Figure 8.4 (c) shows the "roughing" visibility cone for a set of voxel positions in the delta volume of the part. The process of determining the roughing visibility takes around 90 seconds for 1000 voxels. Figure 8.4 (d) shows the collision proofed tool postures at various points in the 2.5D slab.

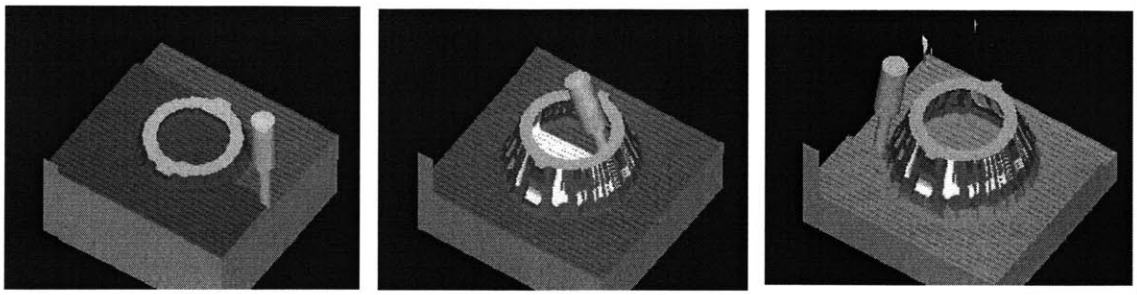
8.1.4 Teacup with handle

In order to demonstrate how our approach accounts for the presence of obstacles, we considered machining the portion of a teacup that has a handle. Figure 8.5 (a) shows a 3D model of an teacup surface along with the handle. We are interested in generating roughing and finishing toolpaths for manufacturing the part. In this Section we will illustrate the various stages involved in generating roughing toolpaths to machine the part from a cubical stock and Section 8.2.2 illustrates the procedure for generating finishing toolpaths. The 3D model of the vane has been discretized into 4,344 triangles. Figure 8.5 (b) shows the process of generating the visibility matrix by viewing the part from different orientations. This takes around 30 minutes for obtaining the visibility information along 1280 directions. Figure 8.5 (c) shows the "roughing" visibility cone for a set of voxel positions in the delta volume of the part. The process of determining the roughing visibility takes around 120 seconds for 12000 voxels. Figure 8.5 (d) shows the simulation of material removal by the collision proofed toolpaths using NCVerify^{TM}.

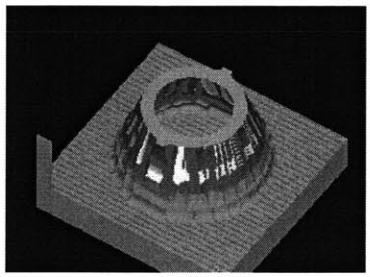
8.2 Finishing examples

The capability of this system for generating surface finishing toolpaths is well illustrated by the considering example parts that have an obstruction. The obstructions in the workpiece could be a design surface or a fixture. Section 8.2.1 shows the various stages involved in generating the toolpath for a surface with an obstruction in the form of a "diving board". Section 8.2.2 shows the various stages involved in generating the toolpath for the surface of the teacup and the handle. In this thesis, we have adopted a machining strategy to minimize the jump in orientations while trying to maintain a constant machining strip width.

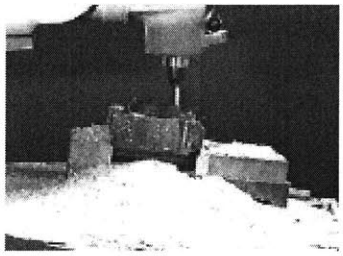
8.2.1 Part with a diving board obstruction

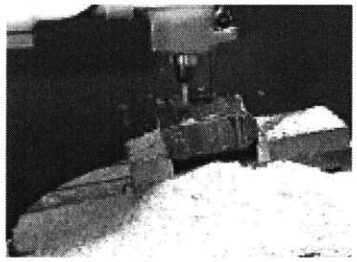


a) Simulation of the toolpath in NCVerify™

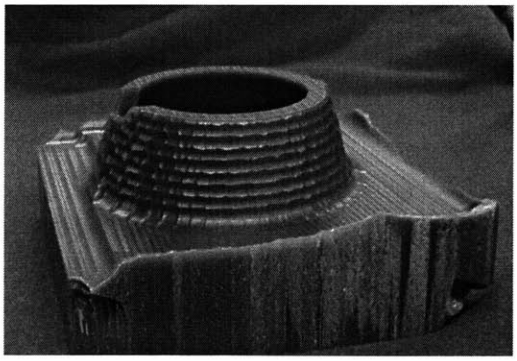


b) 3D model of the part after simulation





c) Machining the part in a five axis machine



d) Machined part

Figure 8.3: NC Simulation and machining the vase

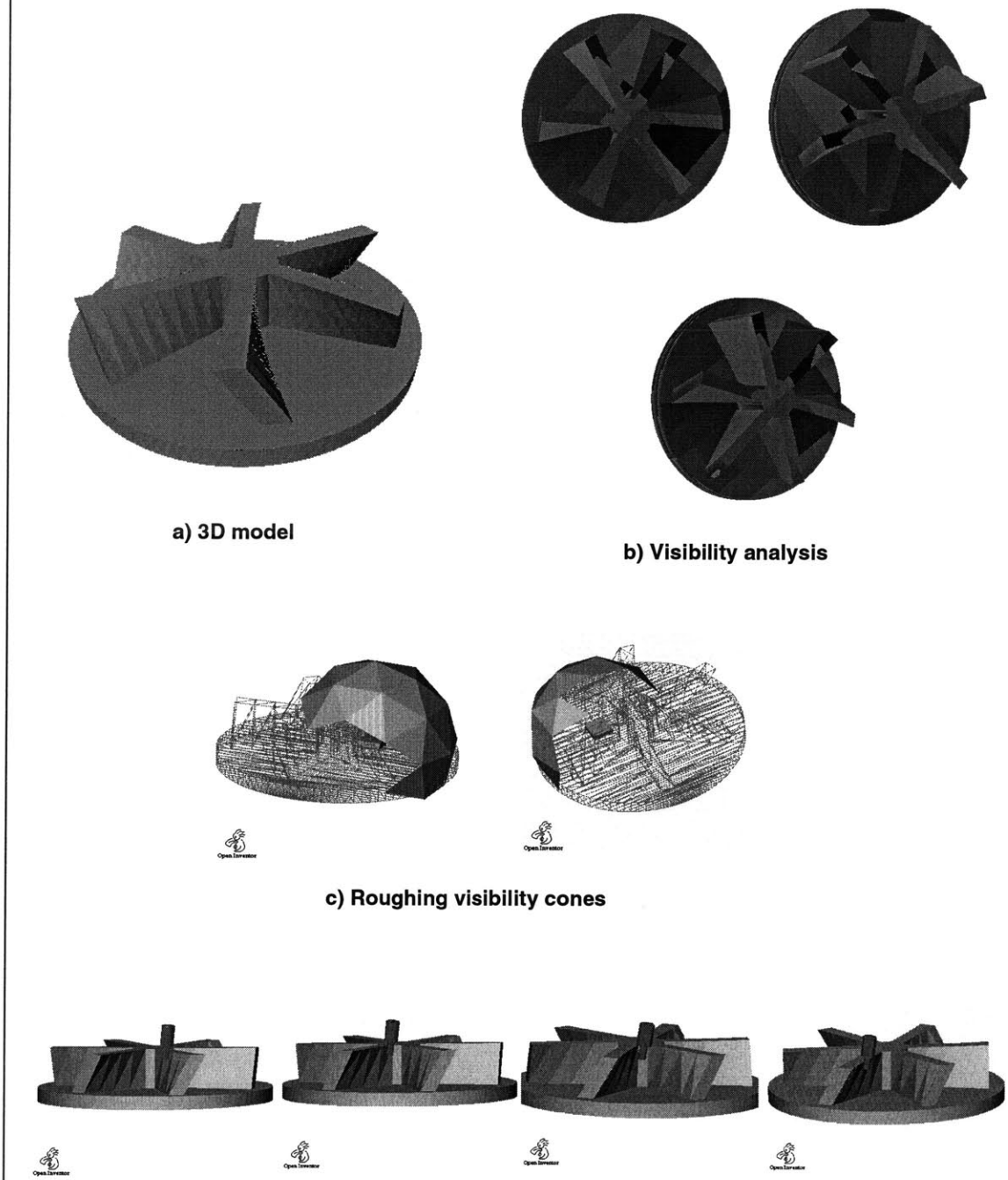
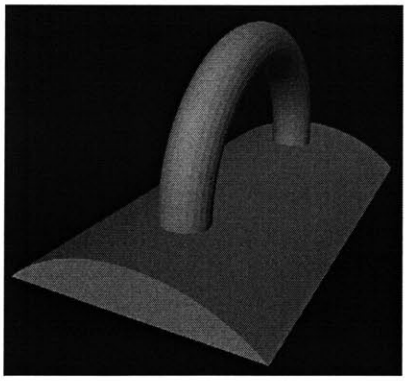
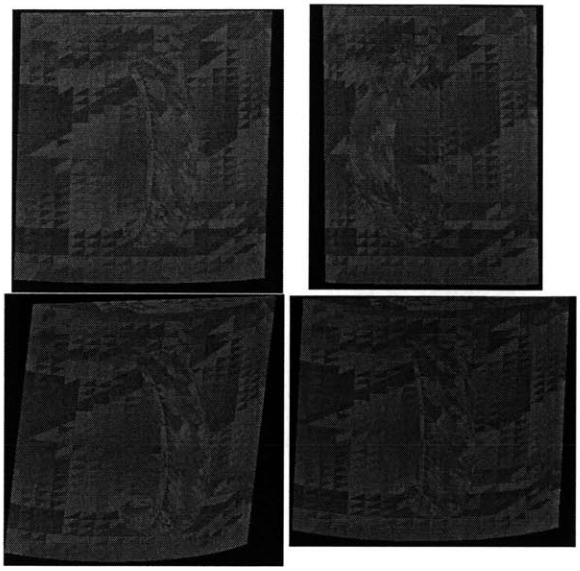


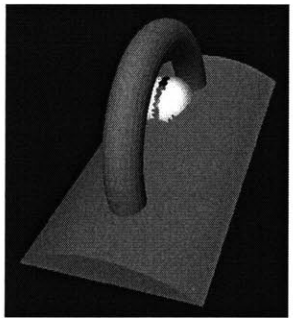
Figure 8.4: Stages in generating roughing toolpaths for an impeller

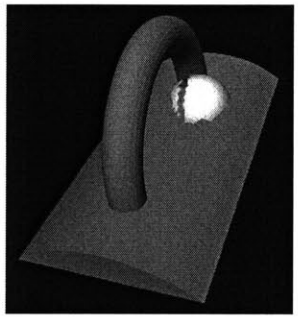


a) 3D model

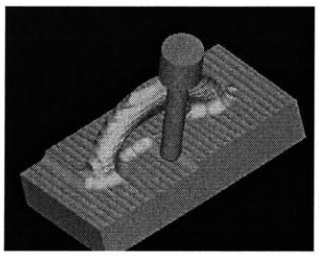


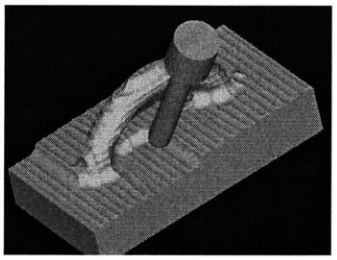
b) Visibility Analysis

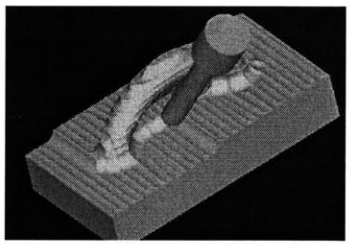


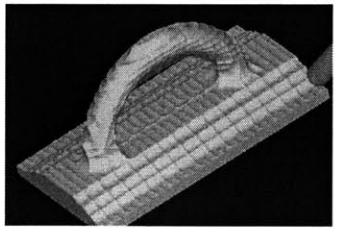


c) Roughing Visibility cones









d) Simulation of material removal using NCVerify™

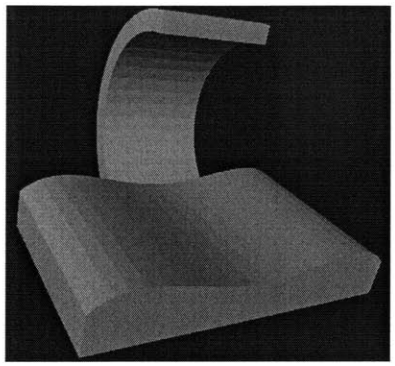
Figure 8.5: Stages in generating roughing toolpaths for a teacup

Figure 8.6 (a) shows the 3D model of the part along with the model of obstruction in the form of a "diving board". The part has been discretized into 3730 triangles and Figure 8.6 (b) shows the process of generating the visibility matrix by viewing the part from different orientations. Figure 8.6 (c) shows the visibility cones for three points on the surface of the part. Figure 8.6 (d) shows the edgelist formed by intersecting the visibility cones with a plane corresponding to constant α. The "promising" tool postures for surface finishing are selected from the "edgelists" using the selection criterion described in Section 6.4.3. The "promising" access directions are corrected locally for collisions and Figure 8.6 (e) shows the collision free access postures. The collision free tool access postures were interpolated to obtain a smooth and a valid toolpath. Figure 8.7 (a) shows the simulation of material removal at the surface NCVerify^{TM}. Figure 8.7 (b) shows different views of the machined part after the simulation.

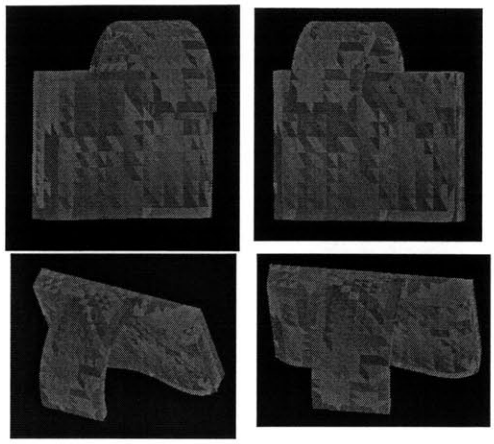
8.2.2 Teacup with handle

The 3D model of a teacup is shown in Figure 8.8 (a). We are interested in generating finishing toolpaths for both the surface and the handle. The procedure for computing the visibility matrix was illustrated in Section 8.1.4. Figure 8.8 (b) shows the visibility cones at points on the surface of the object. It is clearly seen that the handle divides the visibility cone into two regions. Figure 8.8 (c) shows the "edgelist" corresponding to α of 20^{o}. The "promising" access postures are extracted from the edgelist and corrected for collisions. Figure 8.8 (d) shows collision free tool postures to access points on the surface.

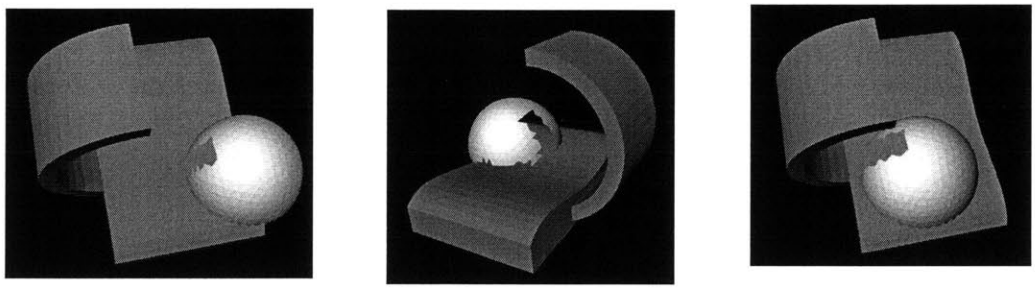
Figure 8.9 (a) shows the simulation of material removal at the surface of the teacup. Figure 8.9 (b) shows different views of the teacup surface after the simulation. The visibility information determined in Section 8.1.4 can be used to generate toolpaths to machine the handle. The "edgelist" corresponding to α of 50^{o} was formed from the surface visibility cone. Snapshots of the tool finish machining the handle are shown in Figure 8.9 (c). Figure 8.9 (d) shows snapshots of the part after the simulation.



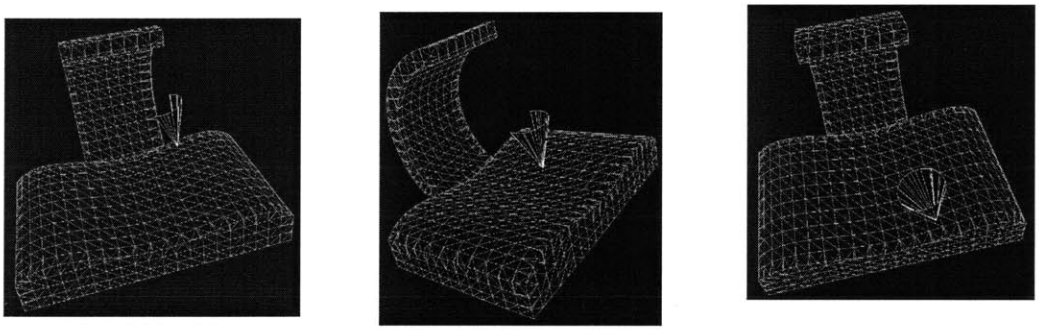
a) 3D model



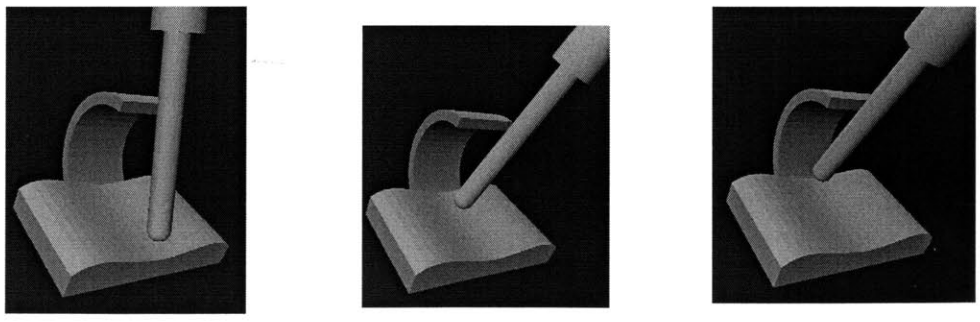
b) Visibility Analysis



c) Surface Visibility cones

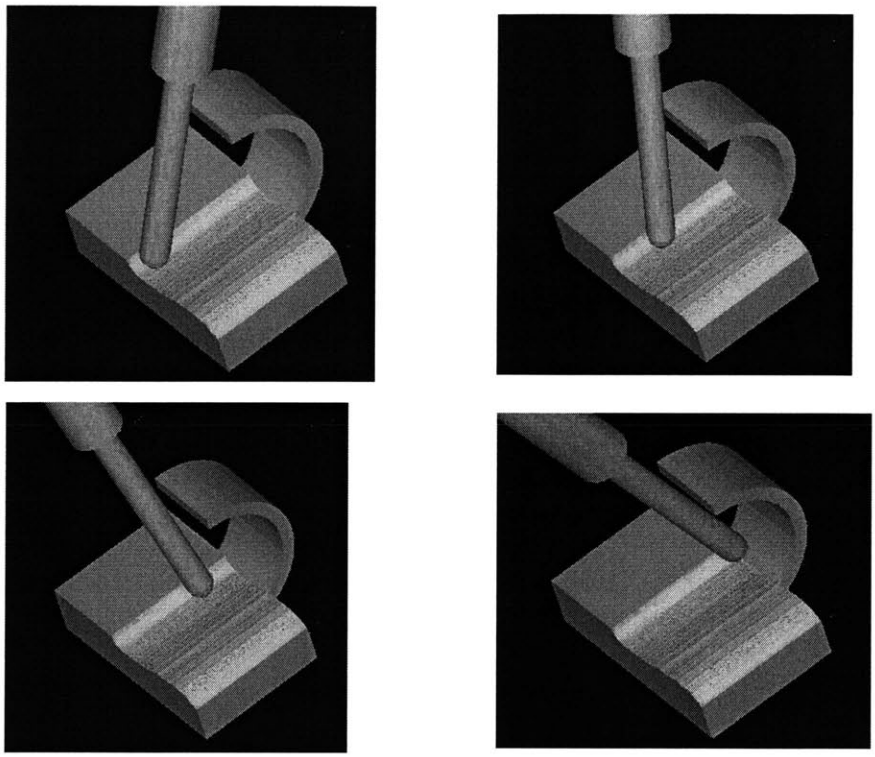


d) "Edgelist" corresponding to $\alpha = 20^{o}$

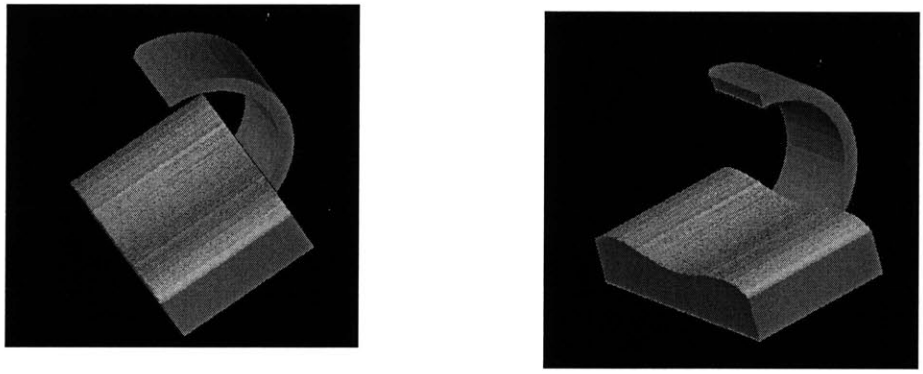


e) Collision free tool access postures

Figure 8.6: Stages in generating finishing toolpaths for the "diving board" part

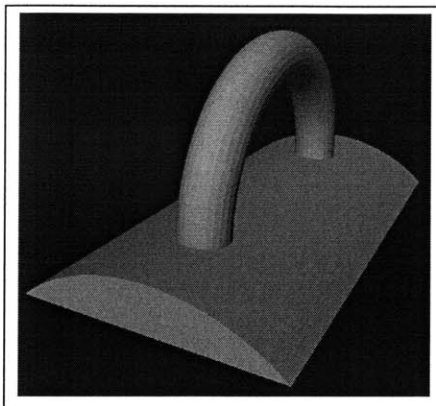


a) Simulation of material removal using NCVerify™

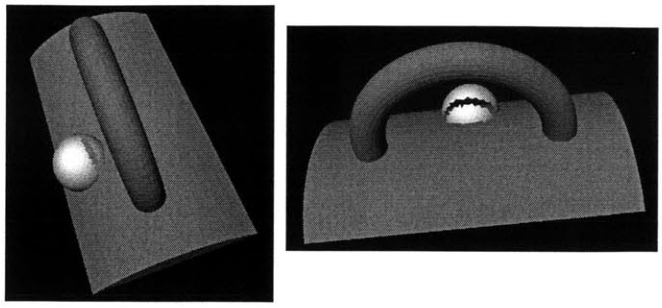


b) Different views of object after simulation

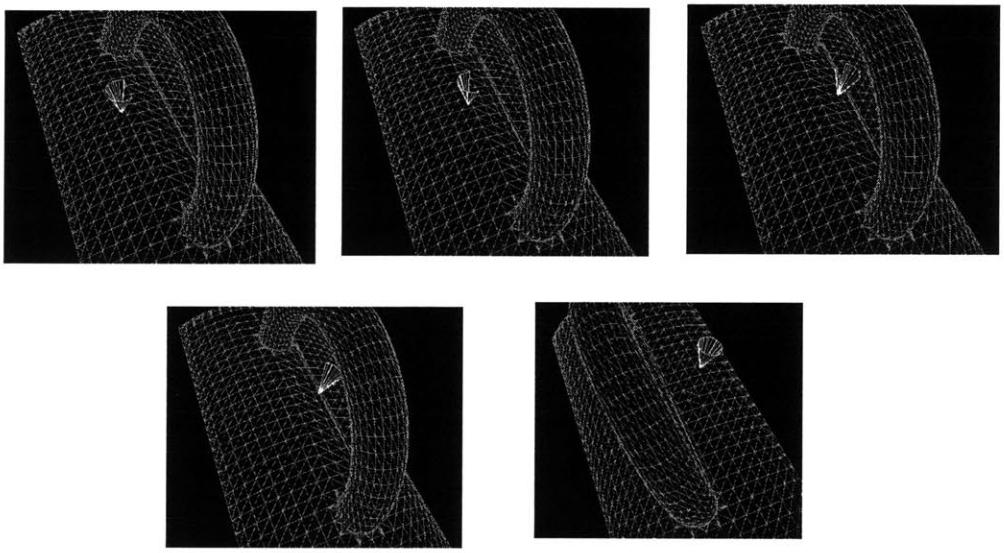
Figure 8.7: NC Simulation of the finishing toolpaths the "diving board" part



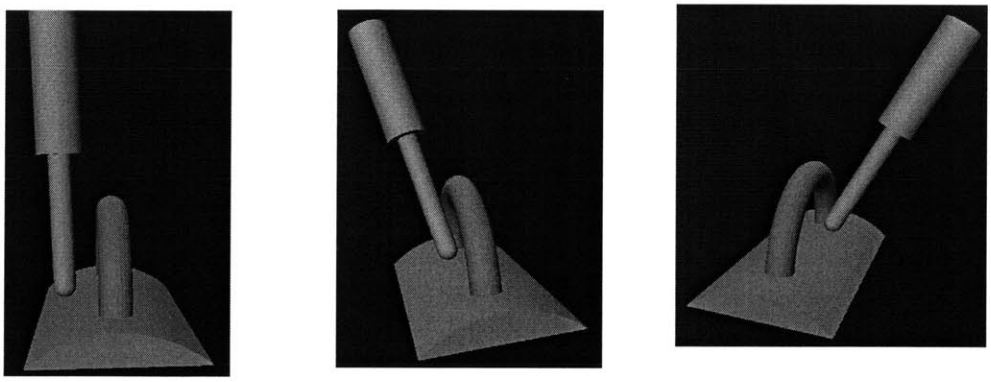
a) 3D model



b) Surface VIsibility cones

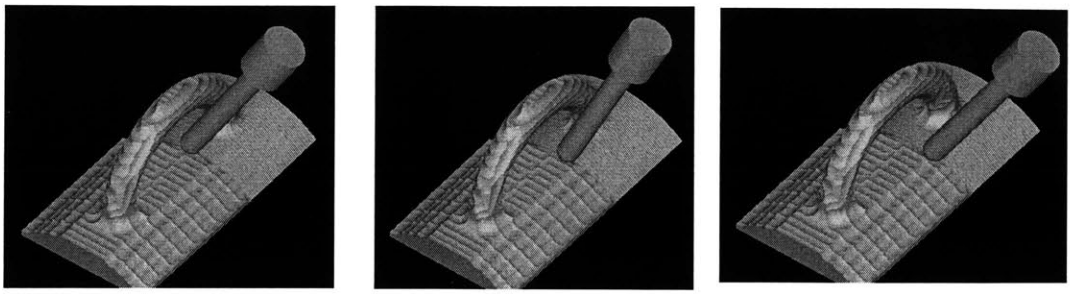


c) Edgelist corresponding to $\alpha = 20^{o}$

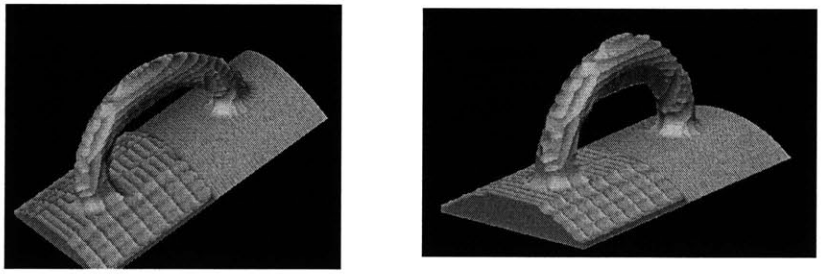


d) Collision free tool access postures

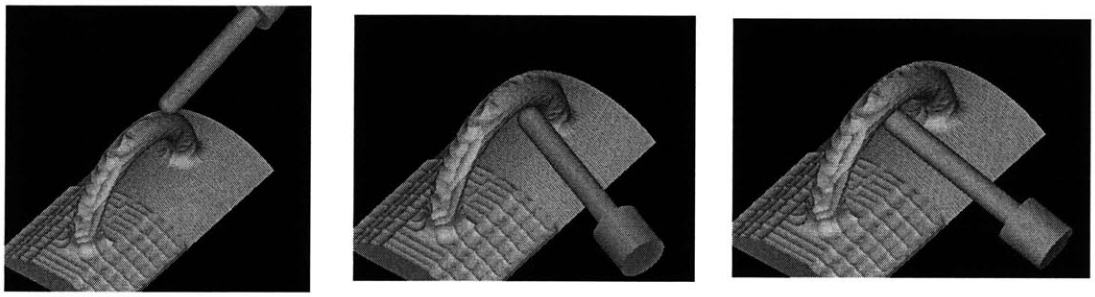
Figure 8.8: Stages in generating toolpaths for a teacup surface



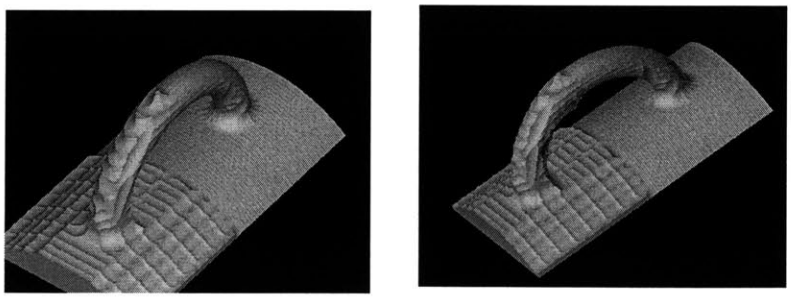
a) NC simulation of material removal at teacup surface



b) After finish machining the surface



c) NC simulation of material removal at the handle



d) After finish machining the handle

Figure 8.9: NC simulation of finish machining teacup surface and handle

Chapter 9: Conclusions and future work

9.1 Conclusions

The work presented in this thesis is a part of a larger effort to make machining a rapid prototyping process. In summary, we present a new technology for generating toolpaths directly from the CAD model of the part. The key idea here is the use of accessibility considerations as the driving constraint in path generation. This is fundamentally different from the feature–based approach, and especially targets the realm of 5–axis machining, for example, of aerospace parts. This approach is an extension of concepts developed by numerous other researchers in the areas of CAM, surface machining and robotics.

We introduced the concept of accessibility cones for points on the object and their use in tool path generation in Chapter 3. The direct evaluation of the accessibility cones for an object involves the computationally expensive minkowski operator. In order to make the problem computationally tractable we consider a class of non–overhung tools for which accessibility can be approximated with visibility. The reasoning behind this approximation is that visibility cones can be computed rapidly and it gives a global idea about the presence of obstructions and any gouge resulting from this approximation can be corrected by using fast collision avoidance routines that function in the toolpath generation loop. We presented a technique to compute visibility cones rapidly by using the graphics hardware as a geometric modelling engine. Though at the outset the memory requirements of the visibility analysis looks prohibitive for real world parts, the fact that we can generate visibility information for a portion of a large object and adaptively mesh the remaining object makes this a tractable problem.

The point visibility cones provides us with a "promising" access direction that is guaranteed to be collision free for a ray of light. However, this need not be a valid access direction for a "real" tool that can be general to include the cutting and non–cutting portions (tool–holder and spindle). Similarly, the workpiece model can include the part, fixtures and any other obstructions in the machine workspace. We proposed the concept of access profiling in Chapter 4 to determine a collision free tool posture by performing a pseudo–gradient local search around the neighborhood of the most promising access direction. The concepts were then applied to generate roughing and finishing toolpaths.

Roughing is a coarse process with a sole aim of hogging out material rapidly from the stock. The part is stratified into several slabs perpendicular to the setup direction and toolpaths are generated slab wise. The concept of "roughing" visibility for the volume and how it can be generated rapidly from surface visibility was explained in Chapter 5. In the absence of any prior information, the skeletal point in the point visibility cone is taken as the "promising" access direction. A "real" tool is profiled for access around the promising access direction. The access profiling included additional constraints to ensure that the tool does not penetrate the slab that is being machined. The valid tool postures are then interpolated to obtain a toolpath that is C_1 continuous in position and orientation.

On the other hand, finishing is a delicate process with minimal material removal. A concept of machining strategy was introduced in Chapter 6 to capture the intent of the machinist. We also pre-

sented a heuristic to select the most "promising" access direction from the visibility cone satisfying the machining strategy. A "real" tool is profiled for access around the promising access direction. The valid tool postures are then interpolated to obtain a toolpath that is C_1 continuous in position and orientation.

The algorithms and heuristics proposed in this work were programmed in C++ and executed on a SGI octane (dual 250Mhz, 128 MB). The methodology of access based toolpath generation was applied to a class of parts for which blind folded application of current "state of the art" techniques would result in invalid toolpaths. The tool paths generated were virtually simulated using a simulation package and then machined on a five axis machine. The screen dumps of the various stages in our approach as applied to several example parts have been shown in Chapter 8.

The technique presented here is, in a sense, a brute force approach to the CAM problem. For example, the visibility computations discussed are a potentially intensive means for approximating accessibility. Improvements in computer processor power, availability of cheap memory and development of new graphics hardware makes this a feasible alternative. Similarly, with the voxelized representation of the workpiece, we are once again taking advantage of the greater memory and performance characteristics of current computer technology.

Although the feature–based approach remains intellectually appealing, we see our approach as the one that is potentially more comprehensive, and one which can overcome some of the limitations of feature–based machining. The construction of a comprehensive access based toolpath generation system is underway.

9.2 Future work

This thesis is a seed work in developing a fully automated access based tool path generation system. As a proof of concept, we have shown the functioning of our system and demonstrated the capabilities of it to generate toolpaths for real world parts. We have implemented the modules to generate the toolpaths from the CAD model of a part, However to develop a system useful to the industry, couple of important modules should be researched upon and implemented.

9.2.1 Effect of shape on admissible orientations in finishing

In machining, the shape of the machined workpiece is defined by

1. shape of the tool and
2. path through which the tool is moved.

Over the years, specialized tools of various cutting profiles have evolved for finishing most common surface shapes. Specialized cutting tools must be oriented in particular ways to achieve the desired shape. This impacts path planning because the orientations of the tool in such cases are constrained not only by accessibility, but also by the shape of the surface and the particular way in which tool must be used. We refer to the orientations dictated by the cut, as opposed to those dictated by access, as *admissible* directions.

Surfaces and tools: The question we ask in this section is how the shape of the surface and the choice of tool together affect the set of orientations from which the cut can be performed. We

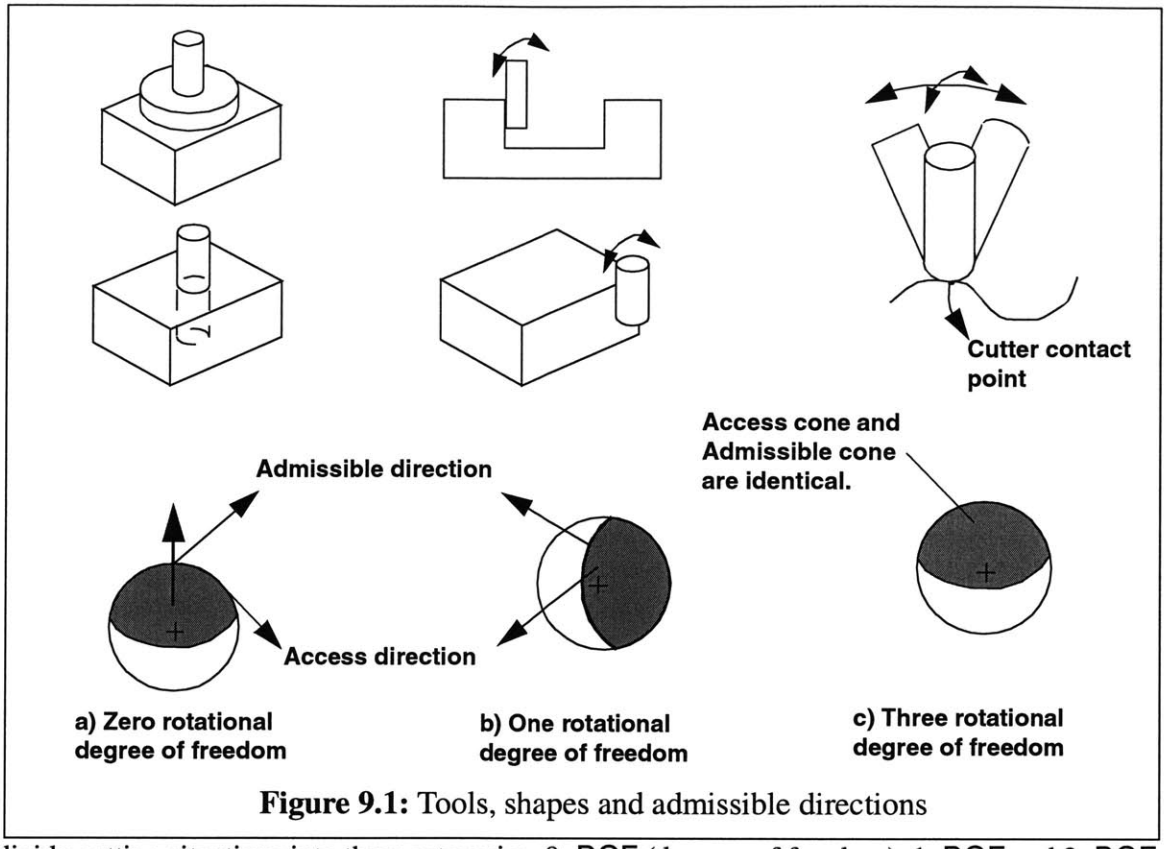


Figure 9.1: Tools, shapes and admissible directions

divide cutting situations into three categories: 0–DOF (degrees of freedom), 1–DOF and 2–DOF, where the degrees of freedom referred to here only pertain to the orientation of the tool, and not to motion in the x, y and z directions.

0–DOF situations occur in processes such as face–milling, drilling, counter–sinking and chamfering, the first two of which are shown in Figure 9.1 (a). Face milling of plane faces, for example, can be performed by either a specialized face–mill or fly–cutter, or by the bottom face of an end–mill. In fact face–milling with the bottom of an end–mill is one of the most common methods for finishing plane faces. In 0–DOF situations, the tool must be oriented the one unique direction dictated by the cut regardless of the access cone. If the admissible orientation does not lie within the access cone, then the 0–DOF cutting strategy under consideration is not permissible. 1–DOF situations occur during side–milling as shown in Figure 9.1 (b), where an end–mill is being used to machine a plane face on its side. The tool can be oriented along any line parallel to the plane, and the admissible tool orientations form a great circle on the Gauss Map. Once again, for side–milling to be a legal option, the great circle of admissible directions must have a non–zero intersection with the access cone. If this intersection is an empty set, then side milling is not permissible. Finally, the least constraining cutting situation is 2–DOF machining, which occurs in end–milling of surfaces with ball end mills, bull–nosed end–mills and occasionally even flat–bottomed end–mills. The admissible directions in 2–DOF situations are two dimensional patches on the Gauss Map, very similar to access cones. 2–DOF is typically used in surface machining, as shown in Figure 9.1 (c).

Edge and corner conditions: Just as the surface and tool affect the admissible directions, so do the conditions of the edges and corners between surfaces. For example, a concave right–angle edge between to surfaces can only be machined with a flat–bottomed end mill oriented parallel to

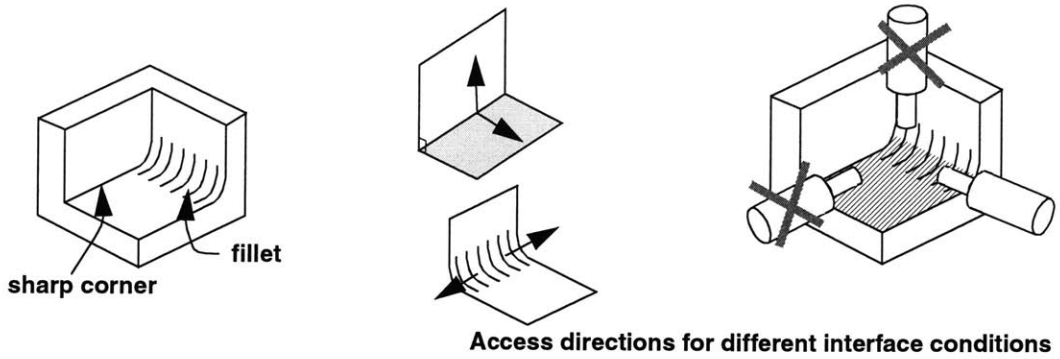


Figure 9.2: Edge conditions

one of the surfaces as shown in Figure 9.2. A tool direction is admissible if it is admissible for the surface, as described above, and if it is admissible for all the edges and corners. Some typical edge and corner conditions, and their implications on the admissible directions are shown in Figure 9.2.

9.2.2 Partitioning and smoothing orientation maps

Orientation maps are an assignment of 3D unit vectors to points on a 2D manifold. In 5–axis CNC tool path generation, an orientation map is used to represent the feasible postures of the cutting tool on a surface for machining the surface. The discrete accessibility map is an approximation of the continuous accessibility map. Figure 9.3 shows a discrete orientation map and how it can be mapped to tool postures.

The discrete orientation map may suffer from two problems. First, it may be noisy. This noise can have a significant impact on the performance of the motion plan, and may result in unnecessarily jerky motion. Second, the discrete accessibility map may lose topological detail of the continuous map. In other words, interpolated postures from the discrete map may unintentionally wander outside the C–Space, resulting in collisions. We have to develop techniques to smooth orientation fields and to detect "edges" *i.e.*, places where there is a large change in orientation, representing a discontinuity in the underlying continuous C–Space.

9.2.2.1 Connection to image processing

Grey–scale images are maps of the form I: $Z^2 \rightarrow R$. Image processing is a highly developed field where several of the problems we have listed have been addressed. It can be seen that the discrete orientation map α: $Z^2 \rightarrow S^2$, described in the previous section, has the same form. The only

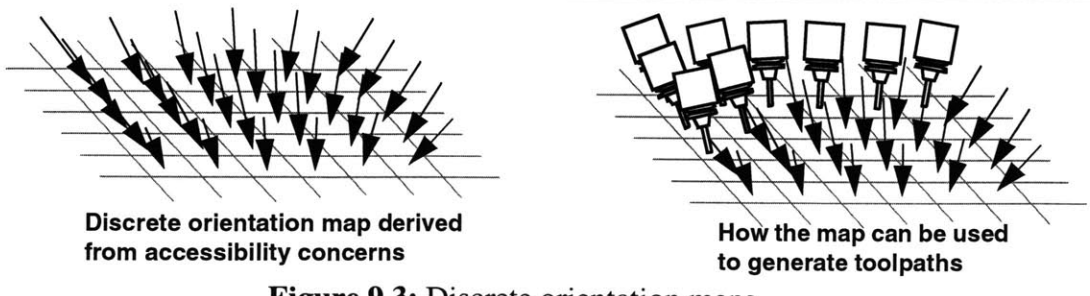


Figure 9.3: Discrete orientation maps

difference is that the familiar space R is replaced with the less familiar space of rotations S^2. The methods of image processing can be applied to orientation maps because rotations have a group structure and algebra.

9.2.3 Efficient data structures

The data generated by "roughing" is a five dimensional array information with a five–tuple index $(x, y, z, \theta, \varphi)$. To store this data in a ordinary data structure is expensive. Currently, memory limitations demand that we use only a coarse voxel grid for generating the "roughing" visibility. A hierarchical data structure such as a 5–d tree that is very similar to an octree spatially can be used to reduce the memory requirements tremendously.

References

Anderson 78

Anderson, R. O., "Detecting and elimindating collisions in NC machining", *Computer Aided Design*, 10(4), 231-237.

Anderson 90

Anderson, D. C. and Chang, T. C. "Geometric Reasoning in feature-based design and manufacturing," *Computers and Graphics*, 14(2) 225-235. 1990.

Anderson 90a

Anderson D.C., Chang T.C., Automated process planning using object-oriented feature based design, in Advanced Geometric Modelling for Engineering Applications, eds. Krause F.-L., Jansen H., Elsevier, IFIP/GI, 1990.

Angleton 89

Angleton, J. M., "Automatic Generation and Correction of Tool Paths for Sculptured Surface Machining", MSc Thesis, 1989, Computer Science Department, University of New Hampshire.

Arbab 82

Arbab, F. "Requirements and Architecture of CAM-Oriented CAD Systems for Design and Manufacture of Mechanical Parts," Ph.D. Thesis, University of California, Los Angeles. 1982.

Banchoff 96

Thomas, F. B., "Beyond the Third Dimensions : Geometry, Computer Graphics, and Higher Dimensions", *Scientific American Library Series*, Paperback (March 1996) W H Freeman & Co, ISBN: 0716760150.

Bronsvoort 89

Bronsvoort, W. F. and Garnaat, H., "Incremental display of CSG models using local updating", *Computer Aided Design*, 21(4), May 221-231.

Chamberlain 93

Chamberlain M.A., Joneja A., Chang T-C., "Protrusion-features handling in design and manufacturing planning", *Computer Aided Design*, vol. 25, no. 1, 1993, 19 - 28.

Chen 92

Chen. L-L and Woo, T. C. "Computational geometry on the sphere with application to automated machining" *Transactions of ASME*, Vol. 114: 288-295.

Choi 93

Choi, B. K., Park, J. W. and Chung, Y. C., "Variable radius blending by ball position sampling", *Proc. of 1st Pacific Conference on Computer Graphics and Applications*, World Scientific Pub. Co., 221-234.

Choi 94

Choi, B. K., Chung, Y. C., Park, J. W. and Kim, D. H., "Unified CAM-system architecture for die and mold manufacturing", *Computer Aided Design*, 1994, 26(3), 235-243.

Choi 98

Choi, B.K. and Jerard, R.B. "Sculptured surface machining", Kluwer Academic Publishers, 1998, ISBN 0-412-78020-8

Cohen 95

J. D. Cohen, M. C. Lin, D. Manocha, and M. K. Ponamgi. "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments," *Proceedings of ACM Int. 3D Graphics Conference*, 1995.

Cutkosky 88

Cutkosky, M., Tenenbaum, M. and Miller, D. "Features in Process-based Design", *Proceedings of the ASME Computers in Engineering Conference*, San Francisco. 1988.

Cutkosky 90

Cutkoskt, M. R. and Tenenbaum, J. M. "A methodology and computational framework for concurrent product and process design", *Mech. Mach. Theory*, **25**(3) pp. 365-381. 1990.

Dong 88

Dong, X. and Wozny, M. "FRAFES: A frame based feature extraction system" *Proceedings of the Int. Conf. on Computer Integrated Manufacturing*, RPI, USA. 1988.

Drysdale 89

Drysdale, R. L., Jerard, R. B., Schaudt, B. and Hauck, K., "Discrete simulation on NC machining", *Algorithmica*

Special Issue on Computational Geometry, 1989, 4(1), 33-60.

Elber 94

Elber, G. and Cohen, E. "Toolpath generation for freeform surface models" *Computer Aided Design*, **26**(6): 490-496. 1994.

Finger 90

Finger, S. and Safier, S. "Representing and recognizing features in Mechanical Designs" *Proc. Second International Conference on Design Theory and Methodology*, DTM '90. 1990.

Foley 95

Foley, J. et al, "Computer Graphics: Principles and Practice", Addison-Wesley, 1995.

Fridshal 82

Fridshal, R., Cheng, K. P., Duncan, D. and Zucker, W., "Numerical cotrol part program verification system", *Proc. Conf. CAD/CAM Technology in Mechanical Engineering*, March, MIT Press. 236-254.

Gadh 92

Gadh, R. and Prinz, F. "Recognition of Geometric Features Using Differential Depth Filters," *Computer Aided Design*, **24**(11): 583-598. 1992.

Gaines 97

Gaines, D. and Hayes, C. "A constraint-based algorithm for reasoning about the shape producing capabilities of cutting tools in machined parts", *Proceedings of the ASME Design Engineering Technical Conference*, Paper # DETC/DFM 4322, September 14-17 1997, Sacramento, CA.

Gindy 89

Gindy, N. N. Z. "A hierarchical structure for form features" *International Journal of Production Research*, **27**(12), pp 2089-2103, 1989.

Gossard 78

Gossard, D. C. and Tsuchiya, F. S., "Application of set theory to the verification of NC tapes", *Proceedings of North American Metalworking Conference*, April, 1978.

Gottschalk 96

S. Gottschalk, M. C. Lin, and D. Manocha. "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection," *Proc. of ACM SIGGRAPH '96 Conference Proceedings*, 1996.

Gupta 95

Gupta, S. K., "Automated Manufacturability Analysis of Machined Parts", Ph. D. Thesis 95-3, Institute for Systems research, University of Maryland. 1994.

Hayes 89

Hayes, C. C. "Automating Process Planning; Using Feature Interactions to Guide Search," *Journal of Manufacturing Systems*, **8**:1-15. 1989.

Held 1991

Held, M., "On the computational geometry of pocket milling", Lecture Notes in Computer Science, Springer-Verlag, London. 1991.

Held 96

Held, M., Klosowski, J. T., Mitchell, J. S. B., Sowizral, H., Zikan, K.. "Real-Time Collision Detection for Motion Simulation within Complex Environments", *SIGGRAPH Visual Proceedings*, 1996.

Henderson 96

Henderson, M. R., Chell, A. R. and Hubele, N. F. "Feature based manufacturing evaluation using statistical process control", *Proceedings of the 1996 NSF Design And Manufacturing Grantees Conference*. 1996.

Ho 1999

Ho, S. "Real time detection of geometric intereference: Application to full-body five axis haptics", S. M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, June 1999.

Horsch 98

Horsch, T. and Juttler, B., "Cartesian Spline Interpolation for Industrial Robots", *Computer Aided Design, 1998,* Vol. 30, 217-224.

Hoschek 93

Hoschek, J., Lasser, D. and Schumaker, L., "Fundamentals of Computer Aided Geometric Design", A K Peters Ltd, ISBN: 1568810075.

Hummel 86

Hummel, K. E. and Brooks, S. L. "Symbolic Representation of Manufacturing Features for an Automated Process Planning System", *ASME WAM*, 1987.

Jerard 89a

Jerard, R. B., Hussaini, S. Z., Drysdale, R. L., Schaudt, B. and Magewick, J., "Methods for detecting errors un sculptured surface machining", *IEEE Computer Graphics and Applications*, 1989, 9(1), 26-39.

Jerrard 89b

Jerrard, R. B., Hussaini, S. Z., Drysdale, R. L. and Schaudt, B. "Approximate methods for simulation and verification of numerically controlled machining programs" *Visual Computer*, **5**(6). 1989.

Jerrard 91

Jerrard, R. B., Angleton, J. M. and Drysdale, R. L "Sculptured surface toolpath generation with global interference checking", *Pres 1991 Des. Productivity Int. Conf.*, Honolulu, Hawaii. 1991.

Joshi 88

Joshi, S. and Chang, T. C. "Graph-based heuristics for recognizing machining features from a 3D solid model", *Computer Aided Design*, **20**(2). 1988.

Kim 90

Kim, Y. S., "Convex decomposition and solid geometric modeling", Ph. D. Dissertation, Stanford University. 1990.

Kim 95

Kim, M. J., Kim, M. S. and Shin, S. Y., "A general construction scheme for unit quaternion curves with simple high order derivatives", *Proceedings of the 22nd annual ACM conference on Computer graphics*, August 6 - 11, 1995, Pages 369-376.

Kramer 88

Kramer, T. R. "Process Planning for Milling Machines from Feature Based Design" *Proceedings of Manufacturing International* '88, pp178-189, ASME. 1988.

Krypianou 80

Krypianou, N. L. K., "Shape Classification in Computer Aided Design", Ph. D. Thesis, Computer Laboratory, University of Cambridge, U. K. 1980.

Latombe 91

Latombe, J. C., "Robot motion planning", Kluwer Academic Pres. 1991.

Laxmiprasad 98

Laxmiprasad, P., "Automatic toolpath generation for multi-axis machining", M. S. in Mechanical Engineering, Massachesetts Institute of Technology, 1998.

Lee 95

Lee, Y.S., and Chang, T.C., "Two-Phase Approach to Global Tool Interference Avoidance in 5-axis Machining", *Computer Aided Design*, Vol. 27, No. 10, 1995, pp. 715-729.

Lee 96

Lee, Y.S., and Chang, T.C., "Automatic Cutter Selection For 5-axis Sculptured Surface Machining", *International Journal of Production Research*, Vol. 34, No. 4, 1996, pp. 977-998.

Lee 97

Lee, Y. S. and Ji, H., "Surface interrogation and machining strip evaluation for 5-axis CNC die and mold machining", *International Journal of Production Research*, 1997, 35(1), 225-252.

Li 94

Li, S. X. and Jerard, R. B., "5-axis machining of sculptured surfaces with a flat-end cutter", *Computer Aided Design*, 26(3), March 1994.

Lin 93

Lin, M. C., "Efficient Collision Detection for Animation and Robotics", Ph.D. Thesis, University of California, Berkeley, December 1993.

Lozano-Perez 81

Lozano-Perez, T., "Automatic planning of manipulator transfer movements", *IEEE Transactions on Systems, Manufacturing and Cybernetics*, SMC-**10**(11): 681-698. 1981.

Lozano-Perez 83

Lozano-Perez, T., "Spatial planning: A configuratin space approach", *IEEE Transactions of Computers*, C-**32**(2):

108-120. 1983.

Marciniak 01

Personal communication with Prof. Marciniak, K., Department of Precision Mechanics, Warsaw University of Technology.

Martin 98

Martin, A., "Analysis and repair of STL files", M.S. Thesis, 1998, California State University, Long Beach, Department of Mechanical Engineering.

Massie 96

Massie, T., "Initial Haptic Explorations with the Phantom: Virtual Touch Through Point Interaction", Masters Thesis in Mechanical Engineering, Massachusetts Institute of Technology, February 1996.

McCool 98

McCool, M., "Orientation and Quaternions", http://www.greenwich.ac.uk/~lp03/Lectures/Graphics/chapter2_10.html.

Michovsky

Michovsky Cutting tools, http://www.cuttingtools.cz/index.html.

Mirtich 98

B. Mirtich. "V-Clip: Fast and Robust Polyhedral Collision Detection," *ACM Transactions on Graphics*, 177-208, July 1998.

Morishige 99

Morishige, K., Takeuchi, Y. and Kase, K., "Tool Path Generation Using C-Space for 5-Axis Control Machining", *Journal of Manufacturing Science and Engineering*, February 1991, Vol. 121, 144-150.

Nau 86

Nau, D. S. and Gray, M., "SIPS: An Approach of Heirarchical Knowledge Clustering in Process Planning", *ASME WAM*, 1986.

Nau 92

Nau, D. S., Zhang, G. and Gupta, S. K. "Generation and evaluation of alternative operation sequences", In A. R. Thangaraj, A. Bagchi, A. Ajanappa and D. K. Anand, editors, *Quality Assurance through the Integration of Manufacturing Processes and Systems*, PED-vol. 56, pp 93-108. 1992.

Oliver 86

Oliver, J. H., "Graphic verification of NC milling programs for sculptured surface parts", Ph. D. Thesis, Michigan State University. 1986.

Rameau 93

Rameau, J. and Supline, R., "Penetration analysis of solids", *2nd ACM Solid Modeling* ('93), Montreal, Canada. 1993

Regli 95

Regli, W., "Geometric algorithms for the recognition of features from solid models", Ph. D. Dissertation, University of Maryland.

Regli 96

William Regli and Mike Pratt, "What are Feature Interactions?", *Proceedings of the ASME Computers in Engineering Conference [American Society of Mechanical Engineers, 1996]*, Irvine, CA, September, 1996.

Roberts 96

Roberts, Chell., Personal conversation, Kansas City, Nov. 1996.

Saito 91

Saito, T. and Takahashi, T., "NC machining with G-buffer method", *Computer Graphics*, 1991, 25(4), July, 207-216.

Sakurai 90

Sakurai, J. and Gossard, D., "Recognizing shape features in solid models", *IEEE Computer Graphics and Applications*, September, 1990.

Salmon 96

Salmon, N., Personal Communication. July, 1996.

Salomons 1995

Salomons, O., "Computer Support in design of mechanical products", PhD Thesis, 1995, Universiteit Twente, http://www.opm.wb.utwente.nl/staff/otto/thesis/contents.html

Sarma 96
Sarma, S. E., Schofield, S., Stori, J, MacFarlane, J. and Wright, P. K. "Rapid Part Realization from Detail Design," *Computer Aided-Design*, **28**(5):383-392. 1996.

Shah 88
Shah, J. J., and Rogers, M., "Functional requirement and conceptual design of the feature based modeling system" *Computer Aided Engineering Journal*, **7**(2):9-15. 1988.

Shoemake 85
Shoemake, K., "Animating rotation with quaternion curves", *Computer Graphics*, 1985, 19(3), 245-251.

Sirius
Sirius Software system, http://www.siriussys.com.

Spitz 00
Spitz, S. N. and Requicha, A. A. G., "Accessibility Analysis using computer graphics hardware", *IEEE transactions on visualization and computer graphics*, Vol. 6, No. 3, July/Sep 2000.

Spyridi 90
Spyridi, A. J. and Requicha, A. A. G. "Accessibility analysis for automatic inspection of parts", *Proc. IEEE International Conf. on Robotics and Automation*, pp 1284-1289, Cincinnati, Ohio. 1990.

Srinivasan 99
Srinivasan, V., "A geometrical product specification language based on classification of symmetry groups", *Computer Aided Design*, 1999, 31, 659-668.

Stage 97
Stage, R., Roberts, C. and Henderson, M., "A framework for represe4nting and computing tool accessibility", *Proceedings of the ASME Design Engineering Technical Conference*, Paper # DETC/DFM 4323, September 14-17 1997, Sacramento, CA.

Suh 95
Suh, S. H. and Kang, J. K., "Process planning for multi-axis NC machining of free surfaces", *International Journal of Production Research*, 1995, Vol. 33(10), 2723-2738.

Sutherland
http://www.mfg.mtu.edu/cyberman/machtool/machtool

Taejung 01
Taejung Kim, "Time-optimal CNC Tool Paths——A Mathematical Model of Machining", PhD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 2001.

Tangelder 96
Tangelder, J., Vergeest, J. and Overmars, M., "Computation of voxel maps containing tool access directions for machining free-form shapes", *Proceedings of the ASME DETC/DFM*, 1996.

Tseng 91
Tseng, Y. J. and Joshi, S., "Determining feasible approach directions for machining Bezier curves and surfaces", *Computer Aided Design*, **23**(5):367-379. 1991.

Turner 88
Turner, G. P. and Anderson, D. C., "An object oriented approach to interactive feature based design for quick turn around manufacturing", *Proceedings of the ASME Computers in Engineering Conference*, San Francisco. 1988.

Udupa 77
Udupa, S., "Collision detection and avoidance in computer controlled manipulators", Ph. D. Thesis, Department of Electrical Engineering and Computer Science, California Institute of Technology, 1977.

Van Hook 86
Van Hook, T., "Real time shaded NC milling display", *Computer Graphics*, Proceeding of SIGGRAPH, August, 20(4), 15-20.

Vandenbrande 90
Vandenbrande, J., "Automatic recognition of machinable features in solid models", Ph. D. Dissertation, University of Rochester, 1990.

Vickers 89
Vickers, G.W. and Quan, K.W., "Ball-mills versus end-mills for curved surface milling", *Journal of Engineering for Industry (Transactions of ASME)*, 111, 22-6.

Voelcker 81

Voelcker, H. B. and Hunt, W. A., "The role of solid modelling in machining process moceling and NC verification", *Transactions of ASME*, 111, 22-6.

Wang 86

Wang, W. P. and Wang, K. K., "Real-time verification of multiaxis NC programs with raster graphics", *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, 7-10 April, 166-171.

Wilhelmy

Wilhelmy, J., "Quaternions and Splines", http://user.cs.tu-berlin.de/~digisnap/splines.html.

Woo 82

Woo, T. C., "Feature Extraction by Volume Decomposition", *Proceedings of the Conf. on CAD/CAM Tech. in Mechanical Engineering*, Cambridge. 1982.

Woo 94

Woo, T. "Visibility maps and spherical algorithms" *Computer Aided Design*, **26**(1). 1994.

Woodwark 84

Woodwark, J. R. and Wallis, A. F., "Creating large solid models for NC toolpath verification", *Proc. of CAD - 84 conference*, Vrighton, UK, Butterworths, 455-460.

Wuerger 95

Wuerger, D. and Gadh, R., "Virtual prototyping of die design — algorithmic, computational and practical considerations", *Proceedings of the Computer Aided Concurrent Design Symposium*, ASME Design Engineering Technical Conferences, Boston. 1995.

Yut 95

Yut, G. and Chang, T. C., " A Heuristic Grouping Algorithm for Fixture and Tool Setups", *Engineering Design and Automation*, 1(1), 21-31. 1995.

Appendix A: Types of Tools

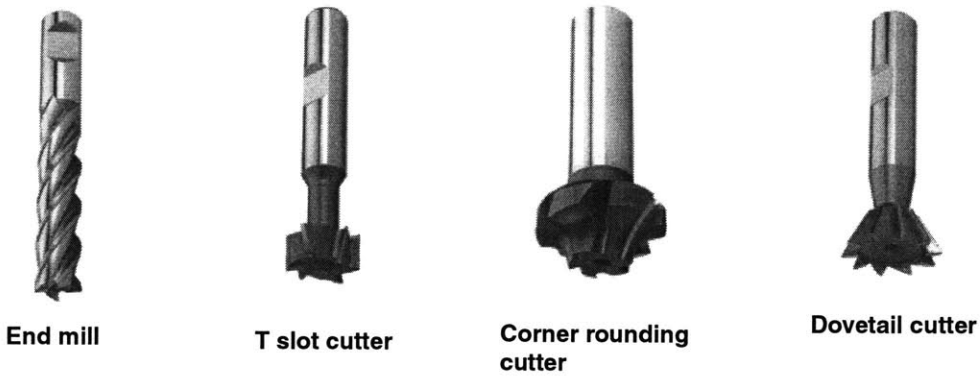


Figure A.1: Types of cutters [Michovsky]

The surface of a cutting tool in a standard, vertical, milling center can be divided into two regions: the cutting portion and the non–cutting portion. Figure A.1 shows the different types of tools that could be used in a milling machine [Michovsky]. The tools can be classified into two categories: non–overhung and overhung tools.

We define non–overhung tools as the one in which a ray drawn from a lower portion of the tool along the tool axis does not exit the body of the tool. Figure A.2 (a) shows a non–overhung tool. This definition implies that as we go up along the axis from the tip, the radius of the tool cannot decrease. The most standard cutting tools like face–mills, end–mills and drills can be adequately approximated as non–overhung. The tool in which this condition is not satisfied is referred to as an overhung cutting tool. Figure A.2 (b) shows an overhung cutting tool. Overhung tools are those which have either an upward cutting region, like T–slot cutters, or narrower shanks in the upper part of the tool.

If a point p is accessible by a non–overhung tool from a direction O, then it is visible along direction O. In other words, for non–overhung tools, accessibility implies visibility. The converse, of course, need not be true. This is the premise of our approach for approximating accessibility

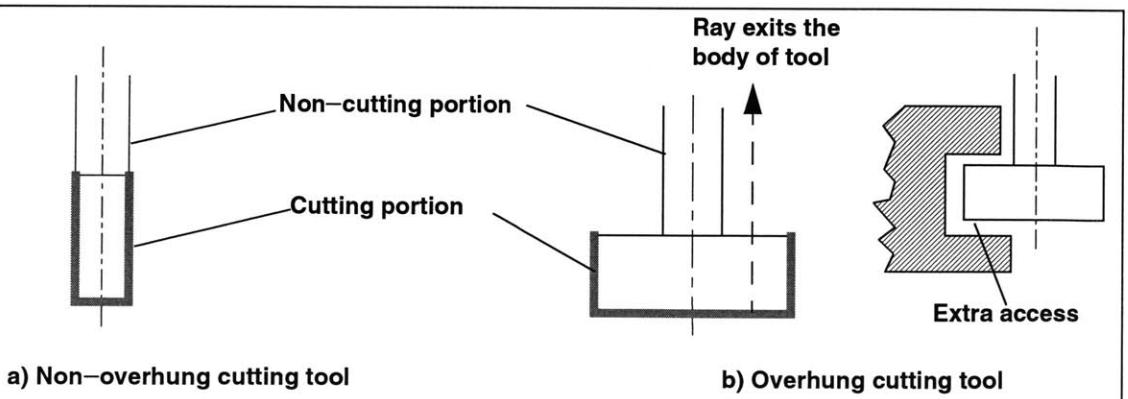


Figure A.2: Overhung and Non–overhung cutting tools

using visibility. However, this is not true for overhung tools. Figure A.2 (b) illustrates the ability of overhung tools to reach undercut features that would not be visible along the axis of the tool. The overhung tools can be approximated as a larger non–overhung tool, but this reduces our ability to reason about access in "undercut features". We have limited our discussion in this thesis to non–overhung cutting tools.

Appendix B: Internal representation of STL meshes

The term STL, for Stereo–lithography tessellation language, refers to the representation of 3D forms as boundary representation constructed entirely of triangular facets. The true beauty of the STL file lies not in its complexity, however, but quite the opposite, in its simplicity. The most basic of planar entities, the triangle, is used to describe everything. A sample STL representation of a triangle is shown below:

```
facet normal 0.000000e+000 0.000000e+000 1.000000e+000
outer loop
vertex   4.330127e-001 -2.500000e-001 2.500000e-001
vertex   4.330127e-001 0.000000e+000 2.500000e-001
vertex   1.801342e-016 -3.685678e-016 2.500000e-001
endloop
endfacet
```

Each triangle is represented by 9 floating point numbers corresponding to the x, y and z coordinates of its three vertices and its outward normal direction. This representation has an advantage of triangles being independent of each other. However, the inherent flaw is that even a vertex shared by n adjacent triangles is represented n times in floating point values.

Floating point operations and the instabilities associated with them have been dealt with by a number of researchers over the past decade. Figure B.1 shows the schema of this representation. The problem with floating point computations leads to instability, since the algorithms are valid and stable for perfect geometric objects only. The error accumulation due to successive floating point operation introduces discontinuities in the mesh. This leads to geometrical flaws, such as tears in the surface of the mesh, that will lead to the tripping of the slice generation algorithm due to inaccuracies in floating point computation.

B.1 Indexed mesh representation

This problem was avoided by converting the STL mesh into an Indexed mesh format where the vertices of the mesh are represented once and the edges of the triangles have pointers to its vertices. Thus this representation has minimal storage of floating point numbers and the geometric information like equality of edges, equality of the intersecting point and adjacency can be accurately determined from pointers or indices.

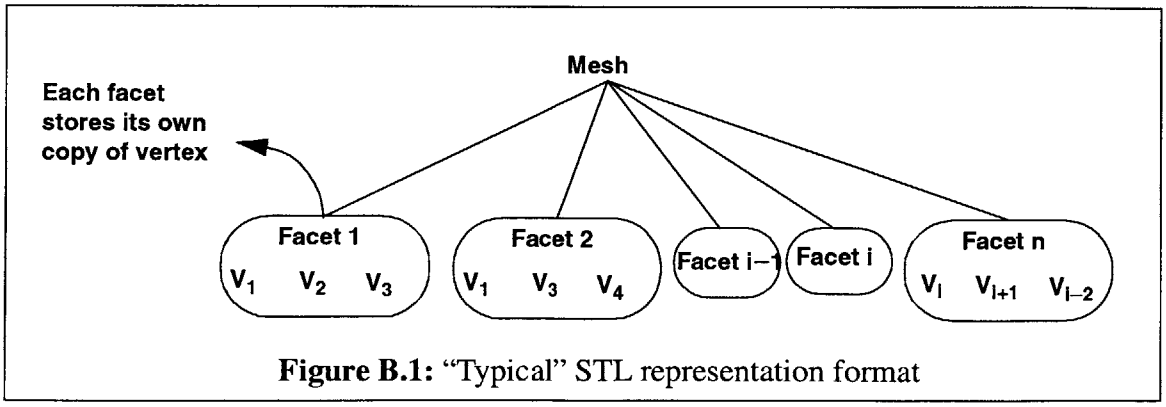


Figure B.1: "Typical" STL representation format

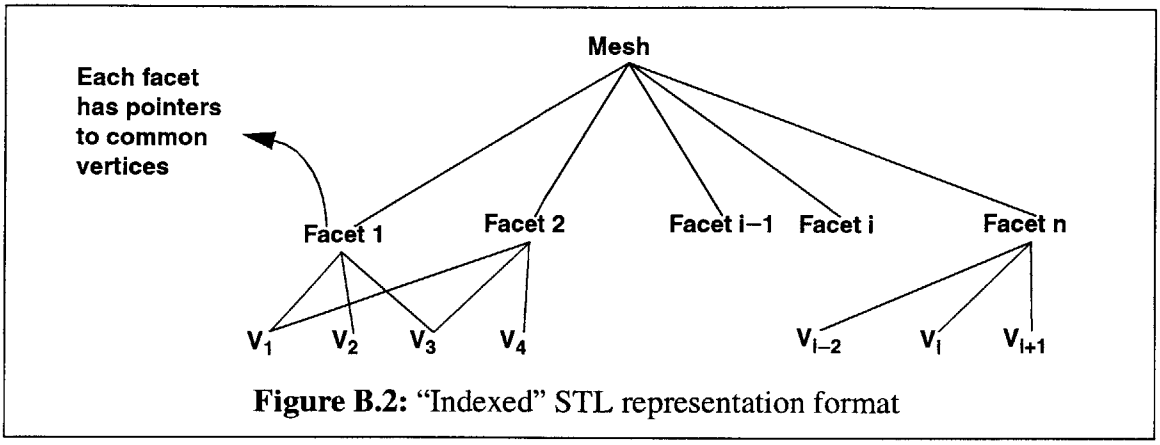


Figure B.2: "Indexed" STL representation format

Figure B.2 shows the schema of representing the mesh in an Indexed format. An Indexed representation of a two triangles is shown below:

Vertex coordinate file

```
<Vertex Number> <x-cood> <y-cood> <z-cood>
1 4.330127e-001 -2.500000e-001 2.500000e-001
2 4.330127e-001 0.000000e+000 2.500000e-001
3 1.801342e-016 -3.685678e-016 2.500000e-001
4 1.801342e-016 -2.500000e-001 2.500000e-001
```

Triangle file

```
<Triangle Number> <Vertex 1> <Vertex 2> <Vertex 3>
1 1 2 3
2 3 4 2
```

If the tessellated representation is generated from the original BRep file, each triangle can be given additional attributes: a description of the surface from which the triangle was derived, its actual normal direction and its actual radius of curvature. These attributes provide a level of geometric and topological detail that most triangulated formats miss. We have used ACIS^{TM} geometric modeler to generate the 3D models and the tessellated mesh was generated with the necessary parameters. The parameters that could be defined by the user are the surface deviation and the aspect ratio of the triangle.

Appendix C: Tool retraction and repositioning

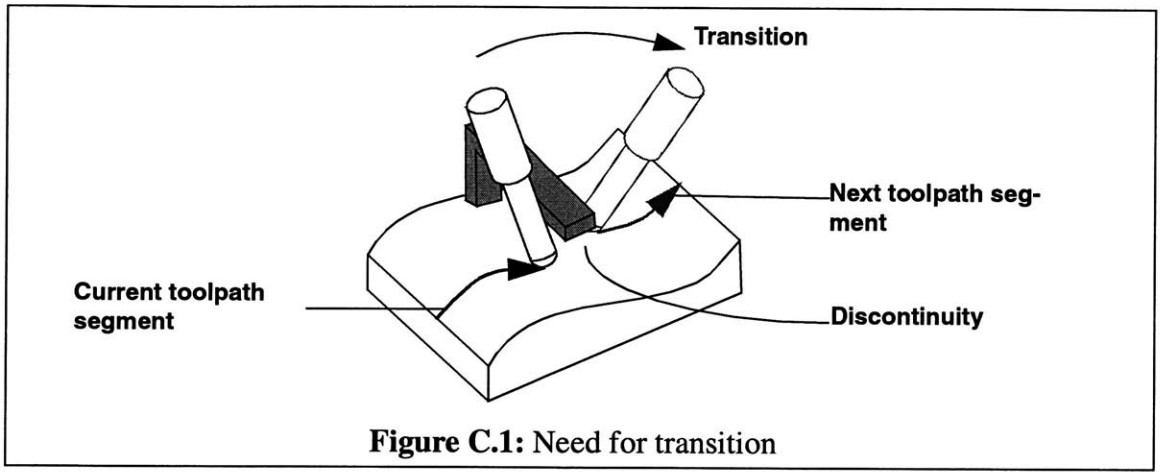


Figure C.1: Need for transition

The toolpath can have several discontinuities when it is machining a 2.5D slice or a surface. A sequence of consecutive tool postures will be referred to as a toolpath segment. In Section 5.3, we described the several possible sources of discontinuities in the toolpath. The discontinuities will force the tool to end the current toolpath segment and begin a new toolpath segment. It is of utmost importance that the tool should not collide with the part while transitioning between two toolpath segments. Figure C.1 shows an example where the tool must be retracted at the end of current toolpath segment and reinserted at the beginning of the next toolpath segment. The problem now boils down to transitioning from a "retraction point" $(x_1, y_1, z_1, \theta_1, \phi_1)$ to "plunge point" $(x_2, y_2, z_2, \theta_2, \phi_2)$ through a set of new points and orientations without colliding with the part.

The most popular solution to this problem involves specification of a relief plane where the transitioning can take place. The relief plane that is well above all the obstructions in the part can be determined based on the bounding box of the part. For non–overhung tools we can use the fact that retraction and insertion of the tool along the tool axis will not result in collisions. The three main stages are given below:

1. The tool is retracted along its axis to the projection point corresponding to the retraction point.
2. The tool is interpolated in the relief plane till it reaches the projection corresponding to the plunge point.
3. The tool plunges into the workpiece along its axis till it reaches the plunge point.

Figure C.2 shows the part along with its relief plane. The tool postures at a (retraction point) and b (plunge point) have been projected on the relief plane along their orientations to obtain a' and b'. The point a' and b' can be determined from geometry using equations C.1 and C.2.

$$\vec{r}_{a'} = \vec{r}_a + \lambda(\vec{o}_a) \tag{C.1}$$

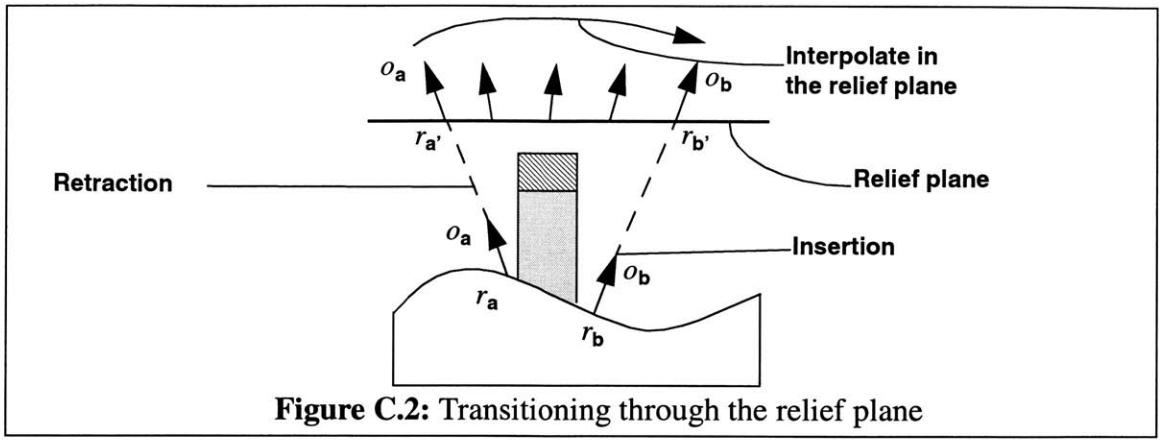


Figure C.2: Transitioning through the relief plane

$$r_{a'} \cdot \overrightarrow{n_{\text{plane}}} + D_{\text{plane}} = 0 \tag{C.2}$$

We can use the interpolation procedure described in Chapter 7 to interpolate between ($r_{a'}$, o_a) and ($r_{b'}$, o_b).

Appendix D: Feature extraction

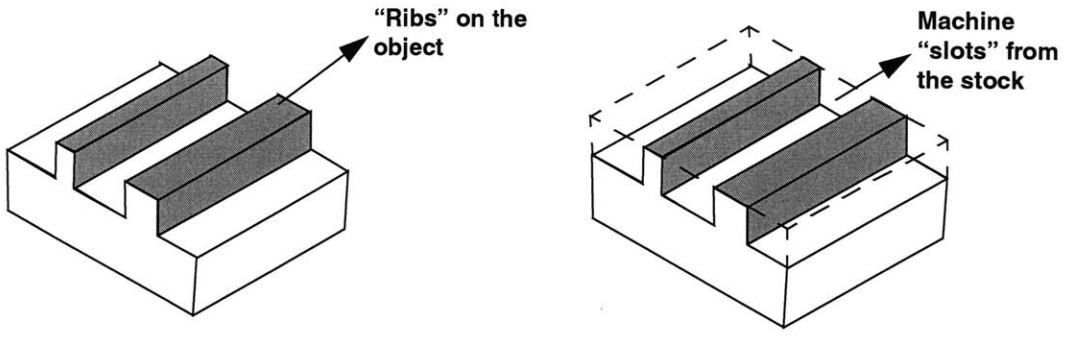


Figure D.1: Design and Manufacturing Features [Salomons 1995]

Features can be viewed upon as information sets that refer to aspects of form or other attributes of a part, in such a way that these sets can be used in reasoning about design, performance and manufacture of the part or the assemblies they constitute. Designing with "design" features overcomes the problem of lack of modelling freedom in the design with manufacturing features approach. Several commercial CAD systems employ this approach. The problem now shifts focus to the procedure for extracting "machinable" features from the "design" feature. This is referred to by researchers as Feature Extraction We present several issues associated with the feature extraction that make it a very tough problem:

1. The design features generally are not application dependent and can differ significantly from those used in manufacturing. Figure D.1 shows a part with design features and the ambiguity present while extracting manufacturing feature for a simple part. A designer could think of ribs as a protrusion feature from a body, whereas a manufacturing planner would think of the design as a slot removed from a stock of material.
2. The mapping between design and manufacturing features is not one to one. For example, One design feature may require several manufacturing features to ultimately yield the desired configuration. Conversely, several design features may contribute to a single manufacturing feature. Therefore, we need feature refinement maps to map design features into a sequence of manufacturing features through geometric reasoning.
3. Interaction between features and their effect on automated feature recognition and manufacturing planning has been studied by Regli [Regli 96].
4. The design and manufacturing engineers prefer different feature representations. For example, design engineers prefer a volume based feature representation whereas manufacturing engineers prefer a surface based representation.

Although, there has been a lot of promising research in feature extraction in recent years, no commercially viable solution has yet emerged.