Automatic tool path generation for multi-axis machining

by

Laxmiprasad Putta

B. Tech., Mechanical Engineering (1996)

Indian Institute of Technology, Madras.

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author .... [illegible] ....
Department of Mechanical Engineering
May 8, 1998

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

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

Automatic tool path generation for multi-axis machining

by

Laxmiprasad Putta

Submitted to the Department of Mechanical Engineering on May 7, 1998, in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Abstract

We present a novel approach to CAD/CAM integration for multi-axis machining. Instead of redefining the workpiece in terms of machining features, we generate tool paths directly by analyzing the accessibility of the surface of the part. This eliminates the problem of feature extraction. We envision this as the core strategy of a new direct and seamless CAD/CAM system. We perform the accessibility analysis in two stages. First, we triangulate the surface of the workpiece and perform a visibility analysis from a discrete set of orientations arranged on the Gaussian Sphere. This analysis is performed in object space to ensure reliability. For each triangle, a discrete set approximation of the accessibility cone is then constructed. Next, a minimum set cover algorithm like the Quine-McCluskey Algorithm is used to select the minimum set of orientations from which the entire workpiece can be accessed. These set of orientations correspond to the setups in the machining plan, and also dictate the orientation in which the designed part will be embedded in the stock. In particular, we bias the search for setups in favor of directions from which most of the part can be accessed i.e, the parallel and perpendicular directions of the faces in the workpiece. For each setup, we select a set of tools for optimal removal of material. Our tool-path generation strategy is based on two general steps: global roughing and face-based finishing. In global roughing, we represent the workpiece and stock in a voxelized format. We perform a waterline analysis and slice the stock into material removal slabs. In each slab, we generate zig-zag tool paths for removing bulk of the material. After gross material removal in global roughing, we finish the faces of the component in face-based finishing. Here, instead of assembling faces into features, we generate tool paths directly and independently for each face. The accessibility cones are used to help ensure interference-free cuts. After the tool paths have been generated, we optimize the plan to ensure that commonalities between adjacent faces are exploited.

Thesis Supervisor: Prof. Sanjay E. Sarma

Table of Contents

List of Figures

List of Algorithms

Acknowledgments

This thesis is the outcome of two of my best years of learning and research. The whole thing started by joining as a research assistant under Prof. Sanjay E. Sarma. Sanjay has been very supportive, he helped me learn the theory behind CAD/CAM and Computational geometry in a matter of months. Sanjay's enthusiasm is contagious, I cannot remember a time I go into his office disappointed about the results, and not coming out with a spirit to solve the problem overnight. Ofcourse, I can never forget the numerous trips to Toscannini's debating about the ways to take over the world.

I could not have asked any one better than Jaehung and Mahadevan to do research with. Jaehung has been a great friend and always inspired me to perfect my programs. Mahadevan has been of incredible help in the development and the implementation of the technology. His hard working and attention to detail is impressive.

I am fortunate to interact with Elmer, Stephen, Taejung, Seppo, Stallion, Samir, Marty and Gaurav. They are some of the best people I have come across in my life. Now to the most important person in the group, our secretary Maureen. Without Maureen here, I would have missed all the home-cooked brownies and most importantly her motherly reminders about my health.

My heart felt thanks to my parents and sister. I could not have gone through this without the love and support of them. This note cannot be complete without mentioning my roommates, Chandra and Siva. We have fool-proof strategies on issues ranging from taking over Microsoft to winning Parliament elections in India.

I thank, everyone mentioned above and others whom I might have missed here, for making my stay at MIT a pleasant and productive one.

Chapter 1: Introduction

Machining is the most widely used process today for producing functional mechanical prototypes. Machined parts can be obtained in a variety of materials with good finish and accuracy. 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 machine 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. This is commonly referred to as computer aided manufacturing (CAM) and CAD/CAM integration.

Over the last decade, the CAD/CAM research community has developed the concept of *machining features* to assist in the conversion of design information into machining instructions. 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. There is little doubt that the concept of features has been a major step forward in the automation of machining, and remains an important avenue of research.

Yet, the feature based approach is not without its disadvantages. Firstly, any feature based system is limited by the extent of its vocabulary. The full extent of the manufacturing capabilities of 4 or 5-axis milling machines cannot be efficiently captured by classical features. Features are essentially 2 1/2 D entities that work well in prismatic parts. But if the workpiece has a complicated spline surface, then representing it with a set of features is a tough, in some cases impossible, task. Secondly, machining features are not directly available from CAD representations. They must be extracted by a process that is referred to as feature extraction. Although there has been some promising research in feature extraction in recent years, no commercially viable solution has yet emerged. Commercial CAM systems and featured based design systems circumvent the recognition problem by requiring the designer to *recreate* the shape in terms of the primitives defined in the system. Since this strategy places the onus of feature extraction on the manufacturing engineer, it is time-consuming, and to an extent, defeats the purpose of generating an initial CAD representation.

Therefore, despite recent strides in feature-based techniques, CAD/CAM integration remains a time-consuming and expensive step in machining. The 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 cycle time and a considerably greater fraction of the actual cost. Furthermore, there is a growing awareness that most 4+ axis machine tools today - especially the

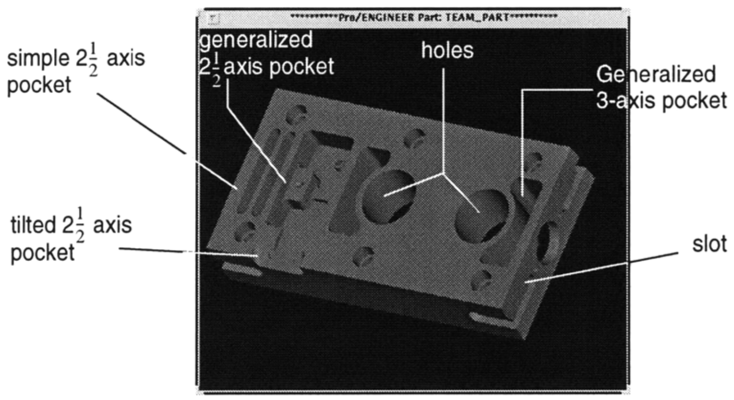

Pro/ENGINEER Part: TEAM_PART
simple $2\frac{1}{2}$ axis pocket
generalized $2\frac{1}{2}$ axis pocket
holes
Generalized 3-axis pocket
tilted $2\frac{1}{2}$ axis pocket
slot

Figure 1.1: Features in a complex component

next generation tools like hexapod - are not fully utilized to the fullest extent possible because of the difficulties associated with tool-path generation. In order to make machining technology more accessible in today's demanding industrial environment, it is necessary to explore other paradigms which may, in the future, overcome the limitations of existing approaches.

In this thesis we outline an emerging paradigm for generating multi-axis machining paths directly from the boundary representation of the geometric object. We refer to this as **Art-to-Part** Machining. The key idea in Art-to-Part Machining is simple: we will generate *free-form cutting paths* to remove all the excess material from the stock while *avoiding local and global interference* with the embedded design. Little effort is devoted to the organization of tool-paths into formal primitives like features. Instead, the goal will be to harness the dexterity of multi-axis machine tools using access arguments.

Borrowing a concept from the robotics community, tool-path generation in our strategy revolves around ensuring *cutting tool accessibility*. However, experiences in robot path planning and other fields have shown that determining exact accessibility is in general a computationally expensive process. As a practical and expedient alternative, we propose to use initial *visibility* analysis to approximate accessibility during the pre-processing stage. This accessibility is further refined during tool-path generation with the help of interference checking routines. These tool-paths are further validated and corrected during simulation and replanning stage. In this way, our approach avoids the up-front expense of accessibility analysis, only incurring it when the approximation is seen to cause interference. We show this in Figure 1.2. With this iterative strategy we hope to bring a long developing idea to practical fruition.

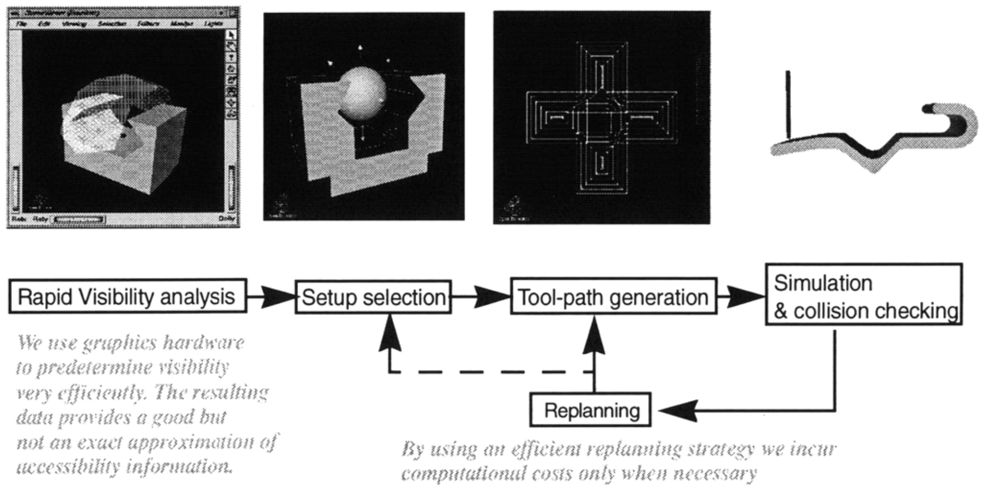

Rapid Visibility analysis
Setup selection
Tool-path generation
Simulation & collision checking
Replanning
We use graphics hardware to predetermine visibility very efficiently. The resulting data provides a good but not an exact approximation of accessibility information.
By using an efficient replanning strategy we incur computational costs only when necessary

Figure 1.2: Our approach to CAD/CAM

Outline: In Chapter 2 we present a brief outline of previous work in the area of CAD and CAM. In Chapter 3, we describe how to carry out visibility analysis, and a procedure to construct discrete visibility cones. We also explain how this visibility data is used to select a minimum number of setups from which the work piece can be fully machined. A general strategy for generating tool paths is presented in Chapter 4. In Chapter 5, we return the results obtained by applying our strategy to 3-axis and 5-axis machining. We conclude the thesis by presenting the future work in Chapter 6

Chapter 2: Background

There has been a large body of work in CAD/CAM integration. Below we summarize this previous research.

Feature based machining: The concept of machining features has been an important step in the understanding and development of manufacturing planning. Machining features have the following advantages: 1) features are a convenient decomposition of a cad model into handleable units for high level planning; 2) tool-path 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 then *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 three-axis machining.

Surface machining: The field of surface machining has been a similarly intense area of research in the last few years. Since Faux' widely used book [Faux 81] a number of systems have been developed over the years for surface machining with special emphasis on die-mold applications: [Oetjens 87, Loney 87, Kuragano 88, Chou 89]. Most early systems, however, were either 3-axis based, or were relatively limited in their applicability because of problems of gouging and surface finish. Recognizing this problem, a few researchers in recent years have looked into the simulation of multi-axis cutting: [Oliver 86, Jerrard 89, Jerrard 91]. Jerrard's work was based on the Z-buffer approach. In the current work, however, we will use a voxelized model for simulation because it provides a more complete picture of the work place. We will use the Z-buffer instead for hidden surface removal. The issue of global interference is discussed in [Choi 89, Tseng 91 and Elber 94], and most recently by Lee *et al* [Lee 92, 95, 96].

Access based approaches: The problem of tool access has been approached from both, a solids, and a surface perspective. Seminal work in the area of visibility and visibility maps was performed Chen & Woo [Chen 92, 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 Spherical 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 Spherical maps has been adopted 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 tool-path generation is carried out in a 3D world. The paths are then inverse-mapped back to the original space to obtain 5-axis tool paths. [Spyridi 90, Henderson 96 & Tangelder 96] explore the concept of accessibility cones as pertinent to 5 axis machining problem. The work presented in this paper focuses on practical and tractable methods to determine and manipulate visibility cones in a manner appropriate for NC tool path generation.

Commercial CAD/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 problem with 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 machining is still very much an acquired skill today.

Recent awareness of these problems has lead to 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.

Robot path planning: The research presented 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. The concept of a configuration space evolved through a series of papers in the early 80's [Udupa 77, Lozano-Perez 81, 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 1991]. 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.

Chapter 3: Accessibility Analysis

Machining process starts by fixing the stock on a machine tool and moving the cutting tool in a predefined path. The tool removes the excess material (delta-volume) from the stock and produces the desired workpiece. The predefined path includes the curve along which the tool has to move and the orientation of the tool along the curve. In case of 3-axis machining the orientation of the tool is fixed. But in 5-axis machining the tool orientation becomes critical, as the tool has two additional angular degrees of freedom.

There are two stages in machining: 1) Roughing and 2) Finishing. During roughing the tool spans through the delta-volume removing most of the unwanted material, and during finishing the tool spans over the surface of the workpiece giving it the required finish and tolerance. To sucessfully carry out these two stages of machining, the tool orientation at various points in the delta-volume and on the surface of the workpiece has to be determined.

3.1 Accessibility

An object is accessible if it can be reached. A point in free space can be accessed from infinitely many directions in R^3 space. These directions are represented as points on the sphere S^2. This representation can be generated by mapping the directional vector to a point on the sphere centered at the origin, where the unit vector joining the origin to the point on the sphere represents the directional vector. The set generated by mapping all the access directions of a point onto the surface of the sphere is called an *accessibility map*. In the present case of a point in free space, the accessibility map is the entire surface of the sphere as shown in Figure 3.1. As more obstructions are introduced around the point, the accessibility map reduces from the entire surface of the sphere to a cluster of small patches on the surface of the sphere. The cones constructed with these patches as the base and the origin as the apex are called the *accessibility cones*. The earlier definition of accessibility is not concrete, as it does not quantify the entity trying to reach the object. For the purpose of machining we define accessibility as,

Accessibility: Point *P* on the surface of the workpiece is said to be accessible by a tool *T* aligned along an orientation *O*, if *P* can be reached by *T* along *O* without violating the following conditions,

1. Only the cutting portion of the tool is in contact with the stock material, and
2. Tool is not interfering with the embedded design

P is called the access point and *O* is called the access direction.

While generating tool-paths, the tool should be given an orientation along which it should be aligned at every point. In order to generate interference free tool-paths, this orientation should be one of the accessible directions for that point. So, to automate the pro-

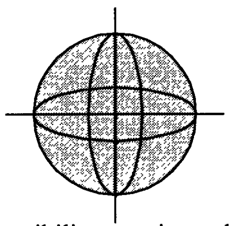
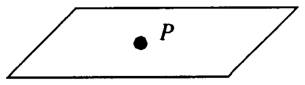
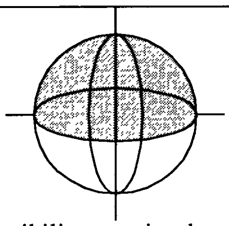
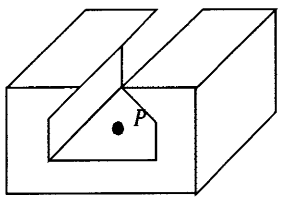
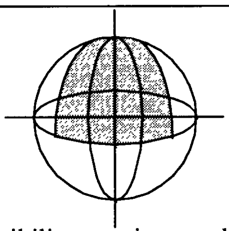
point in free space	 Accessibility map is a sphere
 point on the surface of a infinite plane	 Accessibility map is a hemi-sphere
 point on the surface of the workpiece	 Accessibility map is a patch on the surface of the sphere

Figure 3.1: Accessibility maps

cess of generating tool-paths, atleast one access direction for every point on the workpiece has to be determined. Determining accessibility cone is of real importance in machining. However, the accessibility cone is difficult to determine.

One way to determine the access direction is by trial and error. Choose a random tool and check if it is able to reach the intended point with out any interference. Repeat the above process by changing the tool and the direction of approach till it suceeds. This approach is time consuming and unreliable. [Tangelder 96 and 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. We simplify the process by assuming the tool to be straight line. Under this assumption accessibility is analogous to visibility. We formally define visibility as,

Visibility: Point P is said to be visible along a direction O, if an ray of light from P travelling along O reaches the outer space without interfering with the embedded design.

Visibility maps are generated by mapping all the direction along which a point is visible on to the surface of the sphere. These visibility maps are often referred to as *visibility cones*. The visibility cones are processed further to determine the approximate accessibility direction. Below we discuss how visibility maps are generated for surfaces and volumes.

3.2 Visibility analysis

The shape of the embedded design imposes constraints on the accessibility (in our case, visibility) of various regions of the delta volume. Machining of any point in the delta volume is guided by the accessibility of that point. These constraints determine the permissible tool size and the orientation of the tool. For error-free path planning all the constraints imposed by the embedded design have to be determined. These constraints can be determined by accessibility analysis of the workpiece. Accessibility of a point is defined for a specific tool size and orientation. A point on the workpiece is considered to be accessible if the delta volume around that point is machinable. It is computationally expensive to perform accessibility analysis for all possible tool sizes. Instead, we perform the analysis assuming the tool to be a straight line. We refer to this approximate analysis as visibility analysis.

3.2.1 Using graphics hardware

Visibility algorithms have received considerable attention in the fields of computer graphics and computational geometry. A number of algorithms have been proposed in the literature, and are summarized in [Foley 95]. However, hardware techniques, like depth buffer approach, have recently proved to be very effective. With the ability to scan convert a million polygons per second, visibility analysis can be performed very efficiently within a fraction of a second. 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.

The graphics engine scan converts a model into a scene. In our case, the model is the embedded design. Graphics boards are optimized to render convex primitives. In our visibility analysis, we reverse this process to extract the visible part of the model from the scene. One way to do this is to do a inverse screen transformation of all the points in the

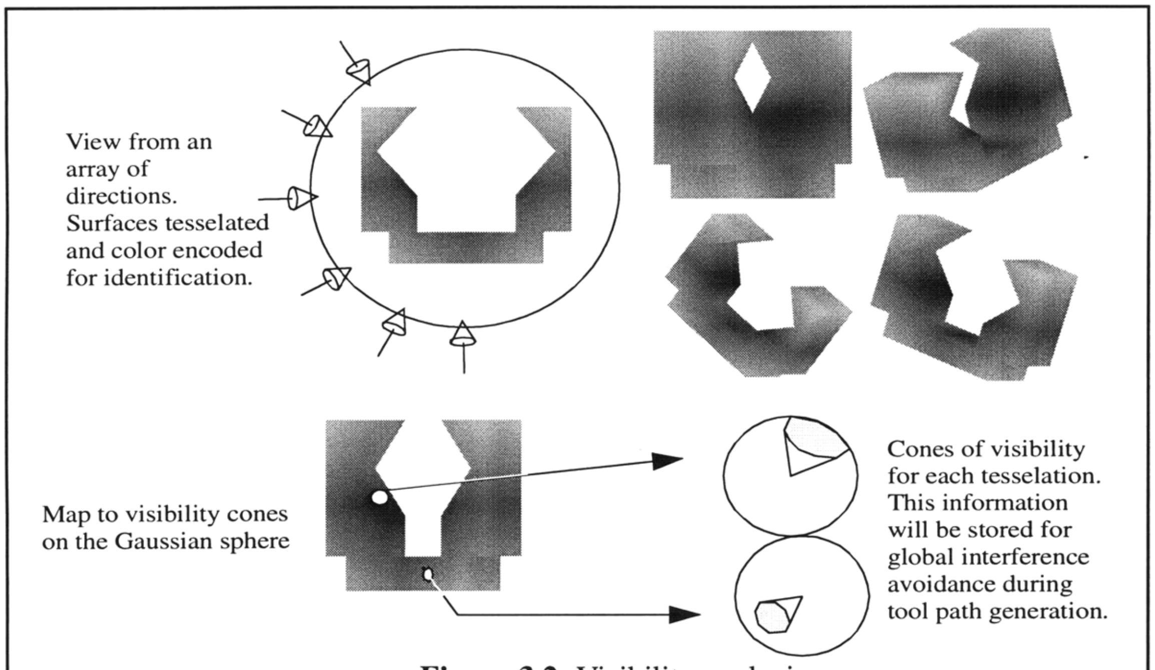

View from an array of directions. Surfaces tesselated and color encoded for identification.
Map to visibility cones on the Gaussian sphere
Cones of visibility for each tesselation. This information will be stored for global interference avoidance during tool path generation.

Figure 3.2: Visibility analysis

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.

In standard boundary representation, faces of a solid are modeled monolithically as single entities. While convenient for solid modeling, this is not ideally suited for visibility analysis. For example, if we were to encode an entire face with one color, the occurrence of that color would not imply visibility of the entire face; 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.

To perform a satisfactory visibility analysis, each face has to be subdivided into smaller entities. If all these sub-entities are visible, it may be possible to assume that the entire face is visible. Obviously, the validity of this assumption is dependent on the size of the sub-entities. In our research, instead of color encoding the faces of a BRep, we first generate a tessellated representation and then color encode each triangle separately as shown in Figure 3.2. Now, if the color assigned to a given color is visible in a certain orientation, we say that the entire triangle is visible from that orientation. Any errors resulting from this assumption will be corrected during later simulation and interference checking. Since the size of the triangles is kept small, the errors resulting from this assumption can be easily compensated.

In 24 bit *(R, G, B)* graphics board each color is represented as mix of *R,G* and *B*. Each color occupies 24 bits, with its components *(R, G, B)* occupying 8 bits each. Internally each component take a value between 0 and 255, both values included. But when the graphics buffer is queried, it returns values between 0 and 1 for each component (in the intervals of 1/255). In order to make the color encoding compatible with the internal representation of *(R, G, B)*, the *(R, G, B)* values for encoding are assigned as follows,

$R/G/B = i/255$ where $0<=i<=255$

If the *(R, G, B)* values assigned do not comply with the above rule, then the fraction is rounded off to the nearest multiple of 1/255. This will lead to errors in identifying the visible triangles, hence the visible part of the model. Note that a typical color graphics card will permit 2^{24} triangles to be encoded in this way.

3.2.2 Sampling directions

Visibility analysis is performed along a set of pre-defined orientations. Two issues have to be considered in determining the set of orientation. Firstly, it is necessary that we consider a fairly even sampling of the Gaussian Sphere. To achieve such a sampling, we start with a tetrahedron and subdivide it according to Algorithm 1, till the desired sampling rate is achieved. A tetrahedron is supplied as the initial input to this algorithm. At the required resolution, the centers of the triangular cells generated in this manner represent a fairly homogenous sampling of the sphere. The resultant triangles are called *Gaussian triangles* to distinguish it from the triangles of the model. The visibility analysis is performed along the directional vector from the center of the sphere to the centroid of the gaussian triangles generated above.

Algorithm 1: Sampling of the Gaussian Sphere

Terminology:

CreateTriangle(v1, v2, v3): constructs a triangle with the given three vertices
Vertex(T, i): returns the i^{th} vertex of triangle T
Edge(V, i): returns the i^{th} edge connected to Vertex V of a triangle
(two edges start from every vertex in a triangle)

Input:

Level of sub-division L
Set of triangles to be sub-divided $T_i()$

Output:

Set of triangles approximating a sphere $T_o()$

Algorithm:

```
For $i = 0$ To $L$ Do
    For Each $T$ belonging to $T_i()$ Do
        For $j = 0$ To 3 Do
            $v_1 \Leftarrow Vertex(T, j)$
            $v_2 \Leftarrow MidPoint(Edge(v_1, 0))$
            $v_3 \Leftarrow MidPoint(Edge(v_1, 1))$
            $T_0$.ADD($CreateTriangle(v_1, v_2, v_3)$)
        End
        $v_1 \Leftarrow MidPoint(Edge(Vertex(T, 0), 0))$
        $v_2 \Leftarrow MidPoint(Edge(Vertex(T, 1), 0))$
        $v_3 \Leftarrow MidPoint(Edge(Vertex(T, 2), 0))$
```

```
                $T_0$.ADD(*CreateTriangle($v_1$, $v_2$, $v_3$)*)
                **For Each** T belonging to $T_t()$ Do
                    *project all the three vertices of the triangle on to the surface of the sphere*
                END
            Copy $T_o$ to $T_i$
            Initialize $T_o$
            END
        END
        return $T_o$
```

Special directions: In addition to homogenous sampling, a second consideration relates to the directions of innate importance to the workpiece. The adhoc sampling we have prescribed above may miss such directions. For example, consider a large plane face oriented at some odd tilt in the work piece. It is not unlikely that this face will be machined with a flat bottomed end-mill, in which the access direction will be perpendicular to the plane. This perpendicular direction may not contained in the homogenous sampling of the Gaussian Sphere. We therefore need to incorporate such special directions into the sample set as well. Some simple heuristics can be used to select and prune the special directions. For example, the perpendiculars to all flat faces in the original BRep representation must be incorporated into the sample set. It may also be necessary to incorporate linear *edges* in the BRep model into the sample set. It is important to exercise reason in creating the sample set because too large a sample set will create computational problems later.

Neighborhood: Figure 3.3 shows how sampling directions correspond to the center of the triangular cells created by the tessellation. Special directions can be handled as follows. A special direction replaces the centroid of the triangle in which it lies as the "representative direction" for that portion of the Gaussian surface. It is possible now to define the neighbors to a particular direction as those whose cells (triangles on the gaussian sphere) contact the ones in question. There are six such neighbors, of which three cells make edge contact, and the remaining three make vertex contact.

3.2.3 Generating discrete visibility cones

To generate discrete visibility cones, visibility analysis is performed from a number of orientation arranged on the Gaussian sphere as shown Figure 3.2:. Each orientation is associated to a Gaussian triangle. 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. If a triangle is visible in that orientation, then its assumed to be visible along all the directions represented by the corresponding Gaussian triangle.

Once the visibility analysis is completed, a set of orientations from which a triangle is visible are obtained. The set represents a region in (θ, φ) space along which the triangle is visible. The cone constructed with this region as the base and the triangle as the apex, as shown in Figure 3.4, is referred to as discrete visibility cone. This information is used to determine the setup directions and tool path planning.

3.2.4 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. For example, a typical end-milling operation will

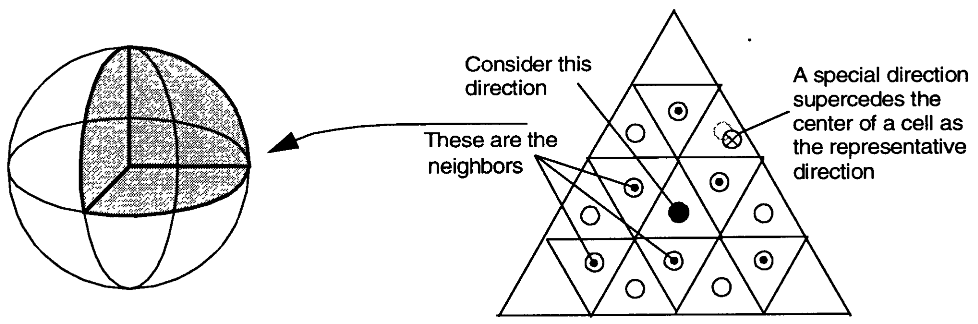

Consider this direction
These are the neighbors
A special direction supercedes the center of a cell as the representative direction

Figure 3.3: Sampling the Gaussian Sphere

create a side face that is parallel to the direction of machining. To account for such situations, we correct the visibility data generated with the *side correction algorithm*:

Algorithm 2: side correction

Terminology:

Let $v(o)$ be the set of triangles visible from a direction o_i.
Let $n(o)$ be the set of neighboring orientations to o.

Input:

o, $n(o)$ and $v(o)$, the orientations and visibility data generated by direct visibility analysis.

Output:

$v^{\perp}$, the visibility data enhanced with side visibility information.

Algorithm:

```
$v^{\perp} \Leftarrow v$
For Each $o' \in n(o)$ Do
    $d(o', o) \Leftarrow v(o') - v(o)$        Determine which triangles that have become invisible
                                            by rotating to this orientation orientations
    For Each $t \in d(o', o)$ Do
        If $t \perp o$,                     If such a triangle is perpendicular to the given orientation
        Then $v^{\perp} \Leftarrow v^{\perp} \cup t$   Then it must be visible along the side.
```

3.2.5 Volume visibility

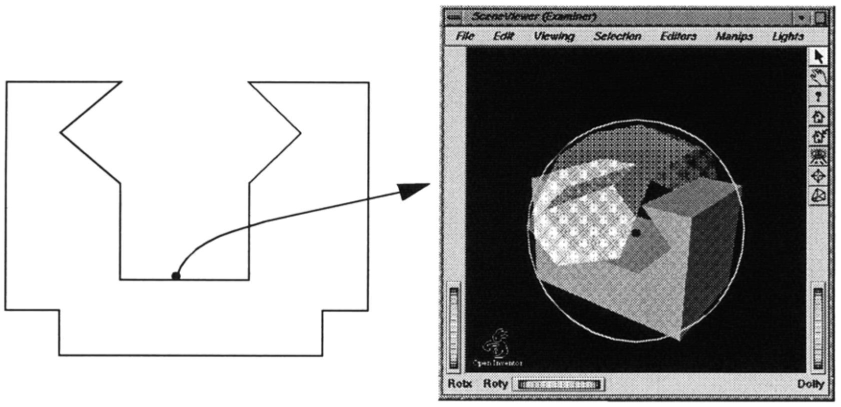


Figure 3.4: Discrete accessibility cone

Thus far we have only considered the accessibility of points on the surface of the workpiece. To generate tool paths within the delta-volume, accessibility information of the interior is necessary. To illustrate this, consider the situation in Figure 3.5. From the visibility analysis we know that point p is visible along $d1$, and point q is visible along $d2$. But, there is no information available about the admissible tool orientations along $\overrightarrow{pq}$. One way to generate this information is to interpolate between $d1$ and $d2$. This might be satisfactory for most of the cases, but there exist some cases where this might lead to tool-workpiece interference as in Figure 3.5. To prevent in the interior interference, accessibility information for the entire delta-volume has to be generated. We call this *volume visibility.*

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. We convert B-Rep to voxels by a simple scan-conversion algorithm [Foley 90, Samet 91].

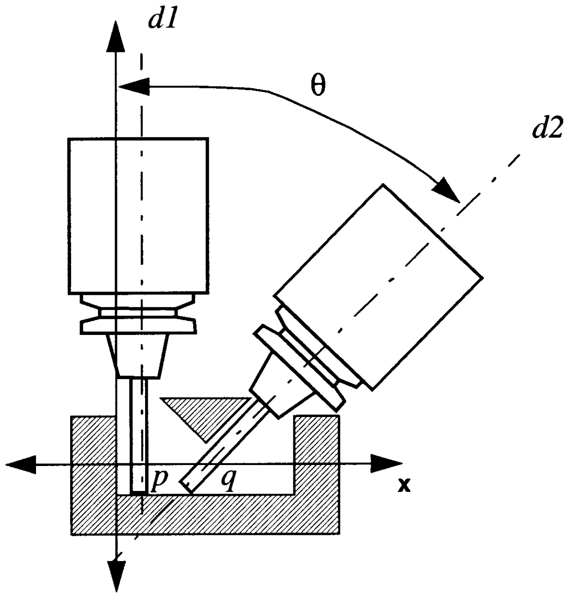


Figure 3.5: Local discontinuity in configuration space

We compute volume visibility by extending the surface visibility information obtained earlier. From the visibility analysis, each point on the surface of the workpiece has a set of orientations along which it is visible. Now, we cast rays from the point along each of these orientations, and tag all the voxels touched by these rays. By taking the advantage of coherence along a line, volume visibility can be computed in this way with minimal computational effort. Algorithm 3 summarizes the approach. 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. However, we are developing hierarchical data-structures in on-going research. The volume visibility cones look similar to the discrete visibility cones shown in Figure 3.4.

Algorithm 3: Volume visibility

```
Input:
    Orientation $O(\theta, \varphi)$
    Triangular mesh $T$
    Visible triangles (from visibility analysis) $VT$
    number of triangles visible *triangleNo*
    Voxel array data structure $VOX[x, y, z, \theta, \phi]$

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

Terminology:
    *tag_voxel()*: appends the visible orientation to the voxel

Algorithm:
    $C_x \leftarrow \cos(\theta)\sin(\varphi)$
    $C_y \leftarrow \sin(\theta)\sin(\varphi)$
    $C_z \leftarrow \cos(\varphi)$
    vertexNo $\leftarrow$ 3
    **For** $i \leftarrow 1$ **To** *triangleNo* **Do**
        **For** $j \leftarrow 1$ **To** *vertexNo* **Do**
        $X \leftarrow$ get_xcoord_vertex($T, i, j$)
        $Y \leftarrow$ get_ycoord_vertex($T, i, j$)
        $Z \leftarrow$ get_zcoord_vertex($T, i, j$)
        **While** $X_{min} \le X \le X_{max}$ and $Y_{min} \le Y \le Y_{max}$ and $Z_{min} \le Z \le Z_{max}$ **Do**
        *tag_voxel*( $VOX[x, y, z, \theta, \phi]$ )
        $X \leftarrow X + C_x$
        $Y \leftarrow Y + C_y$
        $Z \leftarrow Z + C_z$
return *VOX*
```

3.2.6 On the resolution of the graphics approach

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 tessellations and pixels. Fortunately, experiments show that inaccuracies in our approach are insignificant, and the graphics approach is indeed viable as we elaborate below.

Typically, each triangle we create is about 50 pixels in area. We scale the part to ensure that this is the case. Given that there are a limited number of pixels on the screen, this does constrain the resolution of the tessellations we can handle. However, for faces in the size range of 50 *mm*, this means that triangles can be as small as 1 *mm* on the side. At a 5^{o} 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^{o} to the viewing direction is at least half a pixel. Therefore, at a 5^{o} 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 earlier. The surface error caused by a triangulation of 1*mm* is within 10^{-6} *mm* and 10^{-3} *mm* respectively in the typical prismatic and curved components we have sampled.

Another potential source of granularity is the fact that we use a discrete set viewing directions. For example, a cylindrical hole is completely visible only along the axis of the cylinder. A problem might occur if this axis does not coincide with the viewing directions selected as a sample set; the resulting tool paths will obviously be very inefficient.

We combat this problem by including special access directions in addition to the sample set generated around the Gaussian Sphere. These special access directions include, for example, the perpendiculars to large flat faces and the axes of cylindrical holes. This ensures that the sample set does not exclude an obvious access direction, and thus incur huge inefficiencies.

3.3 Setups

The tool path is a 3D curve that defines the motion of the cutting tool. Before starting any machining operation, it is necessary to immobilize the workpiece in a certain orientation. These orientations are called the setups. In the interests of efficiency, it is necessary to minimize the setups during work holding. Note that we will not discuss fixture planning in this paper. We will assume that the workpiece can be fixtured in any setup [Sarma 96].

The issues involved in determining the setup directions in 3-axis machining are quite different from those in 5-axis machining. One definition of the setup minimization problem is as follows:

Determine the minimum number of orientations o_i such that

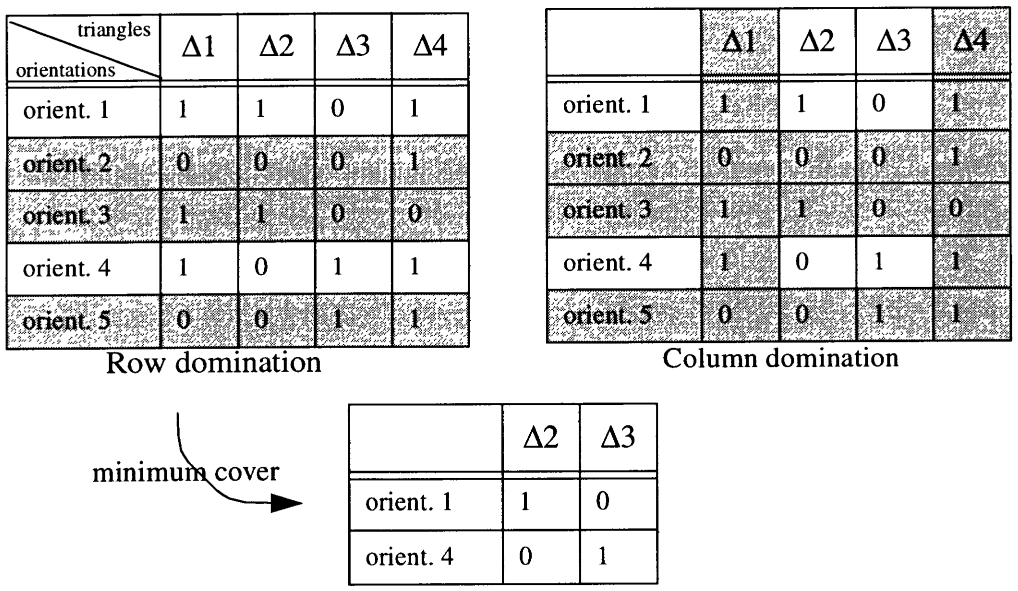
the set of triangles $\bigcup a(o_i)$
where $a(o_i)$ represents the set of triangles visible along o_i

is the set of all triangles on the surface of the workpiece. In other words, find the minimum set of orientations from which the component is completely accessible. In our analysis, we approximate accessibility with visibility.

The visibility data is available to us from the visibility analysis conducted over a large number of sampling directions. Finding the minimum number of setups reduces to the *minimum cover problem*. The general version of this problem is NP-Complete [Garey 79]. One approach to the set cover problem is the well known Quine-McCluskey Algorithm, which has been used extensively in the field of logic synthesis.

3.3.1 The Quine-McCluskey Algorithm

The Quine McCluskey algorithm can be applied directly to our application as follows. If m is the number of sampling orientations and n the number of triangles, we create an $m \times n$ matrix in which the rows correspond to orientations and the columns to triangles. If a triangle is visible from a certain orientation, we mark that element as 1. If not the element is 0. We refer to this as the visibility matrix, and show it in Figure 3.6:.



triangles / orientations	Δ1	Δ2	Δ3	Δ4
orient. 1	1	1	0	1
orient. 2	0	0	0	1
orient. 3	1	1	0	0
orient. 4	1	0	1	1
orient. 5	0	0	1	1

Row domination

	Δ1	Δ2	Δ3	Δ4
orient. 1	1	1	0	1
orient. 2	0	0	0	1
orient. 3	1	1	0	0
orient. 4	1	0	1	1
orient. 5	0	0	1	1

Column domination

minimum cover

	Δ2	Δ3
orient. 1	1	0
orient. 4	0	1

Figure 3.6: Computing the minimum cover

The Quine-McCluskey Algorithm proceeds by a series of alternating row and column dominations.

> **Row dominations:** A row j dominates a row i if every triangle visible from orientation i is also visible from orientation j. In this case, we delete row i. Row dominations correspond to the elimination of unfavorable orientations.
>
> **Column dominations:** A column p dominates a column q if triangle q is visible from every direction in which triangle p is visible. When column p column-dominates column q, we eliminate column q. Column dominations correspond to the identification of hard-to-see triangles, which are more critical in defining the final setups.

The Quine-McCluskey Algorithm proceeds by reducing the size of the visibility matrix with alternating searches for row and column dominations. After at most $o(n^3)$ steps, the algorithm may stall, as no row or column dominations may be available. The visibility matrix at this point is referred to as the cyclic core. In this situation, it becomes necessary to start checking if combinations of rows or columns dominate other rows or columns. This is essentially a brute-force search, as would be expected at some point in an NP-Complete problem. However, the initial application of the Quine-McCluskey algorithm usually reduces the search space enough to make the brute-force technique viable.

Note that if all the elements in a column are 0, then the part is unmachinable, because that triangle can not be accessed. On the other hand, if all the elements of a certain row are zero, then that orientation can be deleted without further consideration, as it is ineffectual. The result of the Quine-McCluskey Algorithm applied in this manner is a minimum set of directions from which the entire surface of the workpiece is visible.

3.3.2 Biasing for parallel and perpendicular directions

Unfortunately, the orientations obtained by this technique may be optimal interms of the number of setups, but not necessarily optimal for machining. This is because in machining it is preferable, as far as possible, to orient the tool perpendicular or parallel to the surface. This is especially important in 3-axis machining where the setup directions determine the orientation of the tool, and the resulting digitizing effect. The algorithm presented in the review section therefore needs to be biased towards orthogonal setups. We state this as follows:

> Observation 2: It is preferable to access a face from a perpendicular or parallel direction [Chen 92].

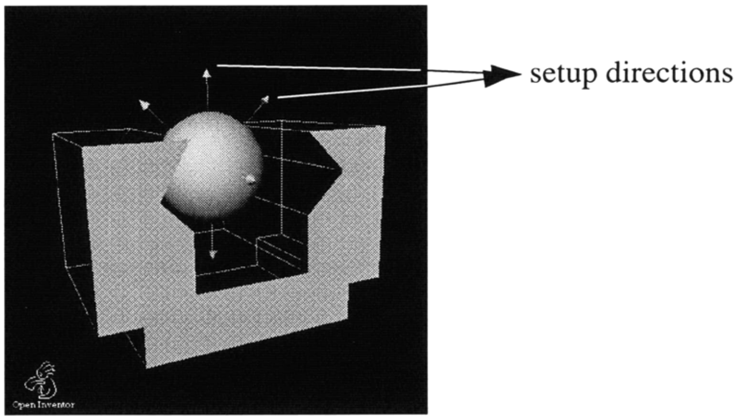
We refer to this as the parallel-perpendicular (PP) heuristic. To incorporate PP heuristic we use a multi-valued version of the Quine-McCluskey Algorithm. The elements of the visibility matrix will be assigned a number as follows:.

element [i,j] = 2 when:
the angle between orientations *i* and triangle *j* is greater than 80^o
or less than 10^o,
and triangle *j* is part of a flat face *f*
and every other triangle in *f* is visible from orientation *j* [1]

= 1 when the angle is between $10^{\,o}$ and $80^{\,o}$ and triangle *j* is visible from direction *i*
= 0 when triangle *j* is not visible from direction *i*

In other words, we assign a higher weight to flat faces that are entirely visible from a certain direction. We then use the following ordering to determine dominance: $0 < 1 < 2$. A row *i* dominates a row *j* if every element of *i* dominates the corresponding element of *j*. Column domination can be defined similarly. This approach biases the Quine-McCluskey Algorithm towards "orthogonal setups" by blocking the domination of important PP orientations. An example of the output of this analysis is shown in Figure 3.7:.

1. Notice that edge conditions like fillets and sharp corners can also be considered in the analysis at this point. For example, there is no point approaching a face from a parallel direction if the edge-conditions of the face are all sharp corners. Parallel access leaves fillets.



Setup direction obtained by performing the visibility analysis from different directions arranged on the Gaussian sphere, and biassing the Quine-McCluskey algorithm towards the PP directions.

Figure 3.7: Setup directions

The performance of these algorithms thus far has been very promising. Typically, the size of the cyclic core is in the order of 10-20 orientations. At this point, enumerative analysis becomes viable and inexpensive. Typical situations involving 5,000-10,000 triangles, and 300-1,000 sampling orientations, can be handled within a few minutes of user time on a standard workstation.

3.3.3 Conclusion

Generating accessibility information is very important in machining. Unfortunately there is no easy and straight forward way of dealing with it. Most of the known methods, like Minkowski operation, are computationally expensive.

We simplify the process by assuming the tool to be a straight line. Under this assumption accessibility is analogous to visibility. We perform visibility analysis from a set of orientations arranged on the Gaussian sphere. Discrete visibility cones are constructed for each triangle in the model by keeping track of the orientations along which that triangle is visible. These visibility cones are used in the setup selection and in determining the approximate accessibility direction. Determining the approximate accessibility direction is discussed in the next chapter.

Chapter 4: Tool path generation strategy

Our aim is to machine the delta-volume and produce a work piece of specified finish and tolerance. This is achieved by generating tool paths to machine the delta-volume. Tool path has two components: the path along which the tool must move and the orientation of the tool along the path.

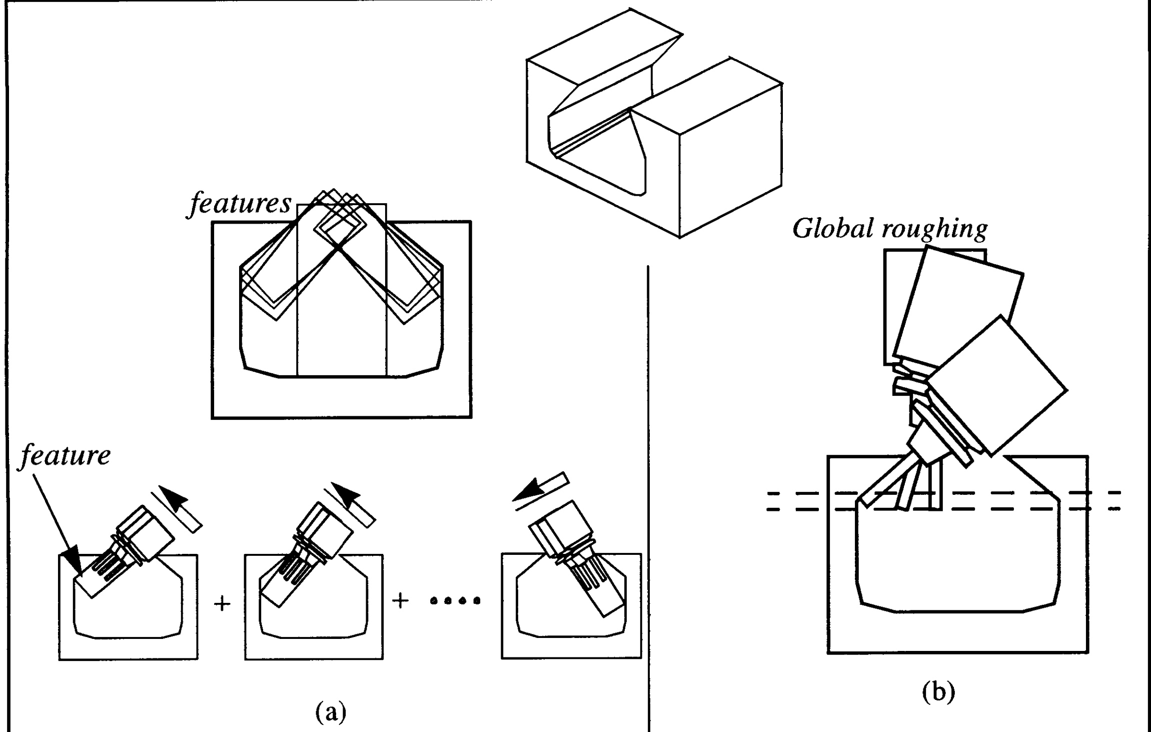
In the previous chapter we discussed the process of generating discrete visibility cones for a triangle of the model. These cones represent the region through which a ray of light can reach the triangle with out interfering with the embedded design. But, since the tool is of a fixed diameter, all these orientations cannot be legal. In this chapter, we will illustrate the process of determining the approximate tool orientation from the discrete accessibility cones.

There are many techniques discussed in literature to generate tool paths. None of these techniques are capable of generating tool paths automatically for a complicated part. Our approach, discussed below, is more general and helps in automated tool path generation.

In our approach we remove the delta-volume in two stages. In the first stage, the bulk of the volume is removed using a heavier tool. We refer this stage as *gobal roughing*. In the second stage, faces of the workpiece are considered one at a time and then a tool is used to remove material at a finer rate to leave the workpiece surface with the required finish and tolerance. We refer to this stage as *face-based finishing*.

4.1 Global roughing

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



Feature based description is awkward. Features must be extracted and tool-paths generated individually for each feature. Resulting paths likely to be redundant. Inappropriate tools may be selected based on features (instead of largest tool).

Global roughing: Tool-path is generated directly from geometry. Maximal tool is selected, and there are no redundant cuts. Multi-axis tool path "searches" for delta-volume

Figure 4.1: Global roughing: A simple example

material before finishing operations. In the previous chapter, we described some general visibility techniques to aid global interference avoidance. In this section we show how roughing can be performed regardless of the individual features. The basic idea of G*lobal roughing* is illustrate Figure 4.1.

At this stage, we assume that the fixturing and the setup details are taken care of. Fixturing has to be taken care by the user. Setup selection can be done by the modules described in the previous chapter or can be user defined.

4.1.1 Slicing the tessellated model

The program receives the part in a pre-determined posture. The tessellated representation is sliced at a sequence of depths perpendicular to the setup direction. The slice plane intersects some of the triangles of the model. We extract the line along which the triangles intersect the slice plane. These lines are grouped together, based upon the neighborhood information of the triangles in the model, to form closed contours. This process could generate more than one closed contour. Some of the contours correspond to the embedded design and some of them to the delta volume. In addition to the above contours we also generate a contour corresponding to the stock. Figure 4.2-a shows the contours generated

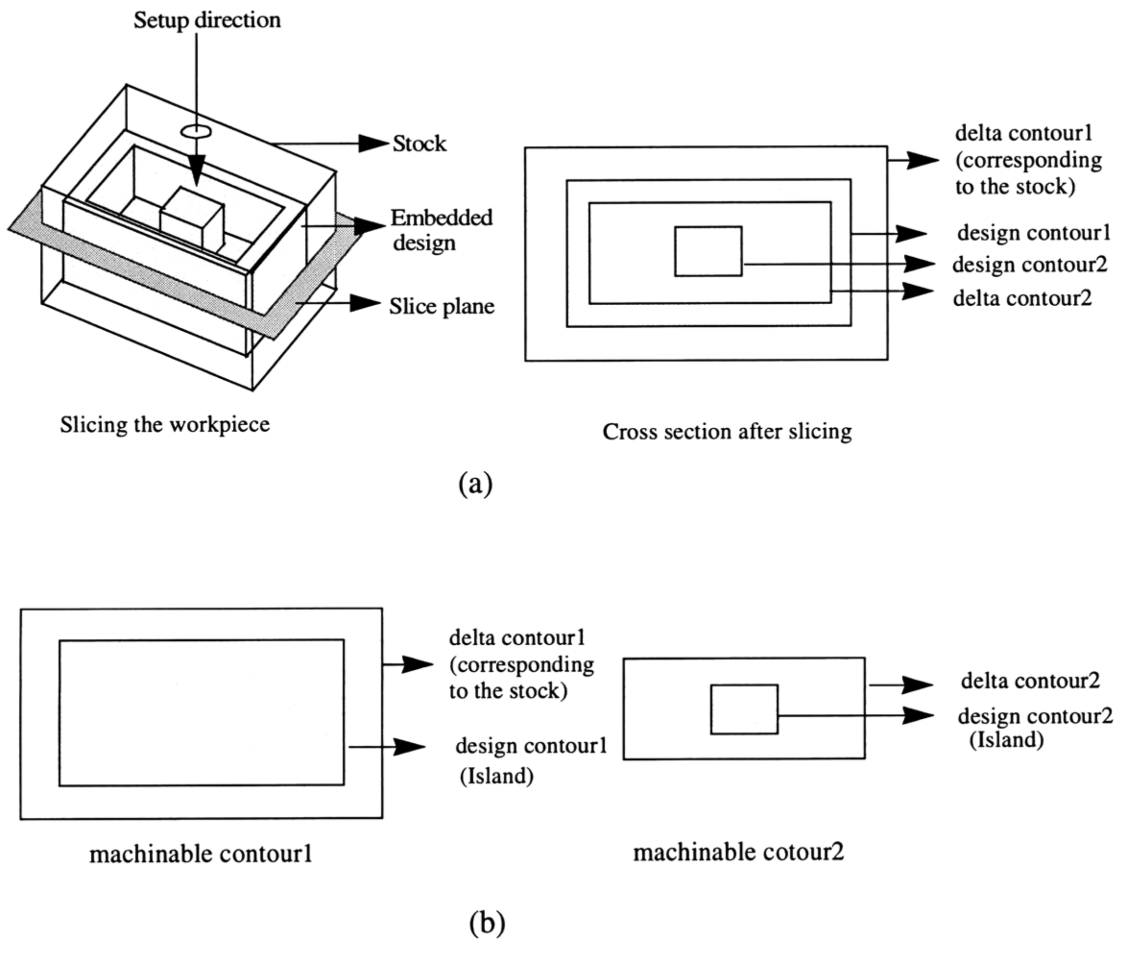


Figure 4.2: Slicing the workpiece

by slicing the embedded design. Machinable contours are generated by identifying the contours corresponding to the design that are contained within a contour corresponding to the delta volume, as shown in Figure 4.2-b.

4.1.2 Contour Offset

We generate centerline tool paths to remove the delta volume. The center of the tool follows this path during the process of machining. These tool paths cannot proceed all the way to the boundary of the contour, as the tool would gouge into the embedded design. To prevent this, the contour has to be offseted non-linearly by distance d given by,

$d = D/2\cos(\theta)$

θ is given by the angle the tool makes with the setup direction

As mentioned earlier, the contour is constructed by grouping together the line segments obtained by slicing the tessellated model. So, each line has a corresponding triangle in the tessellated model. In the previous chapter, we constructed discrete accessibility cone for each triangle. The angle at which the tool should be oriented with respect to the setup direction, when it is in the neighborhood of a particular line, is obtained by processing the accessibility cone of the corresponding triangle. The procedure to do this is illustrated later in this chapter.

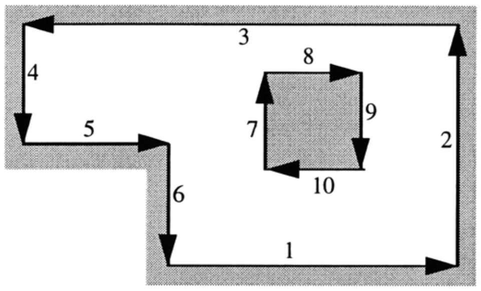
Pseudo code1: Non-linear offset

Given:

Machinable contour (One outer contour and zero/more islands)
Normal direction N of the outer contour
Tool diameter D

Pseudo code

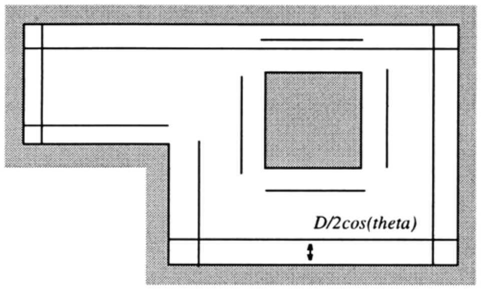
step1: Number the elements (line segments) of the outer contour in the counter clockwise direction and the elements of the islands in the clockwise direction. Arrange the vertices of the contour elements such that their directional vector points in the direction as shown in the figure below..



Step1

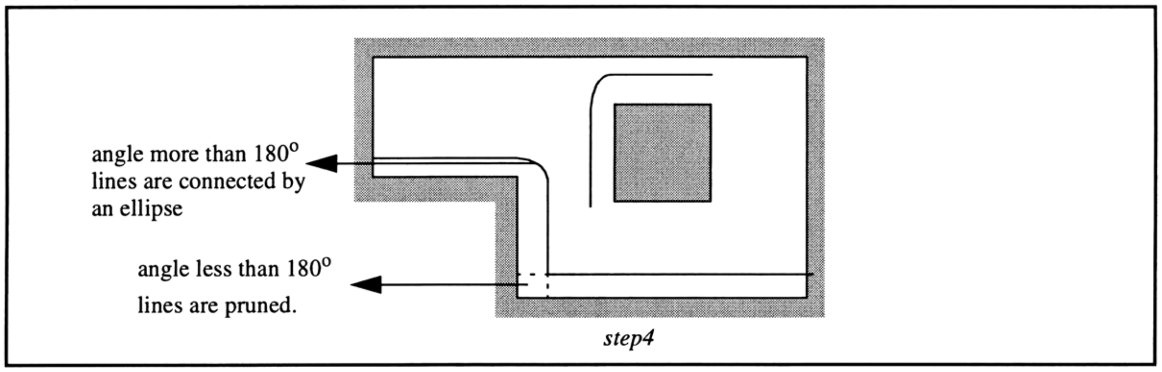
step2: For each element *i* in the contour, find the orientation(explained in the next section) the tool has to approach in order to machine the material in its neighborhood. Find Theta which is the angle between the tool orientation and the setup direction

step3: Offset the element *i* by a distance $D/2cos(\theta)$ in the direction given by $N\ X\ D_i$, where D_i is the directional vector of element *i* and *X* represents the crossproduct between two vectors..



Step3

step4: Consider two successive elements *i* and *j* in that order. If the angle between them is less than 180^o then eliminate the part of *i* that is between the point of intersection and the end of the line, and part of *j* that is between the start of the line and the point of intersection. If the angle in greater that 180^o, then there will be a gap between the two offset lines. Connect these two lines with a part of a ellipse. The equation of the ellipse is got by solving the equation of a conic section with C^0 continuity (end point of the first line and the start point of the second line) and C1 continuity (slopes of the lines) as the boundary conditions. At this stage, the offset elements of an individual contours when put together form a

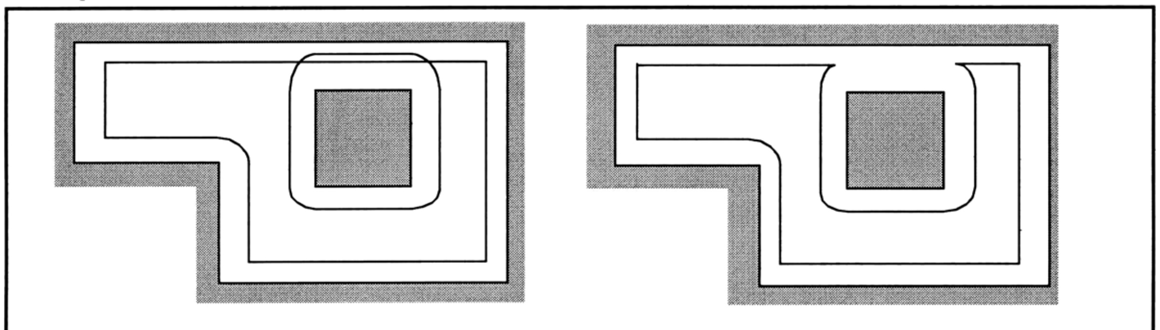


step4

closed contour called the offset contour. But if the offset distance is too large then the offset contour might have self intersecting loops. Loop elimination techniques have to be used to eliminate these loops.

step5: Once *step4* is performed for the outer contour and the islands, perform an union of all the offset contours corresponding to the islands.

step6: Subtract the resultant contour of all the islands from the offset of the outer contour.



step6

The offset contours obtained from this step are then associated with the voxels that they occupy. These voxels are used to harness the voxel visibility data derived during the visibility analysis. There are three levels in this analysis, at increasing levels of scale. We describe them in order in this section. The first stage is concerned with orienting the tool, in a single voxel, and we describe it in Section 4.1.3 below. The second stage is concerned with interpolating a tool path between two adjacent voxels, and we describe it in Section

4.1.4. Finally, at the most global scale, we are interested in stringing together a tool path that covers the entire slab. We describe this in Section 4.1.5.

4.1.3 Orienting the tool in each voxel

Our approach is to tackle the tool path problem at the most local level and to build a global path from local information. In this section we describe how to orient a tool in a particular area of a slice, namely, the region inside one voxel. We are *not* concerned with moving the tool sideways from voxel to voxel in this section. We address that problem in 4.1.4, entitled "Voxel-to-voxel transition"

4.1.3.1 Access direction from visibility cones: cone thinning in voxel roughing

In the previous section we described how it is possible to generate visibility cones rapidly from a tessellated approximation of an object. In reality, however, visibility does not imply accessibility. Whether a point on the workpiece is truly accessible depends on the shape of the tool, which we have not determined yet. The question we ask is which is the most effective direction from the point of view of access? We ignore questions of admissibility in this discussion, as it is not pertinent to roughing.

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 measures of the center. One would be to find the "center of mass" of the cones. However, this would not work because the center of mass 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. As shown in Figure 4.3, the skeleton may be obtained by thinning the tessellated representation of the visibility cone on the Gaussian Sphere. The thinning we have shown is similar to that used in computer vision applications with two important differences: firstly, we are using triangular rather than square cells, and sec-

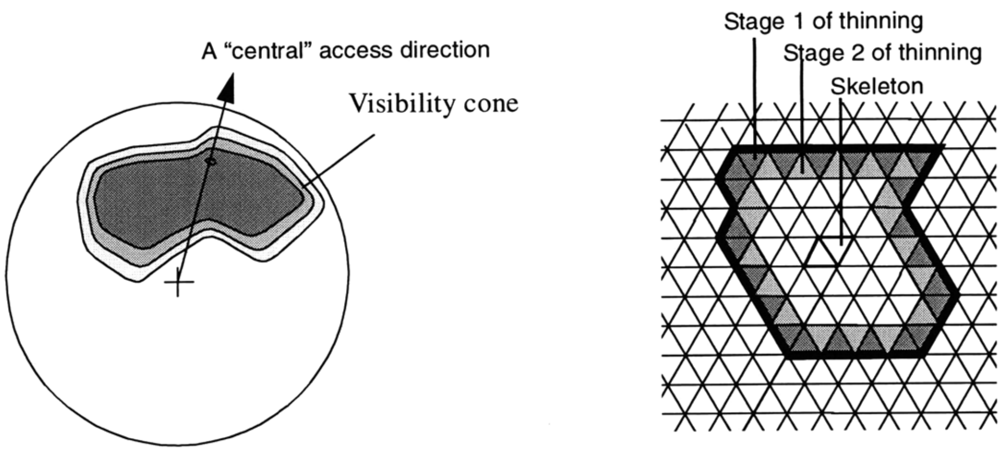


Figure 4.3: Cone thinning

ondly, the thinning is being carried out on the surface of a sphere. The algorithm consists of an iterative thinning step 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, as shown in Figure 4.3, we move the boundary inwards and repeat the process iteratively. We stop the iteration when the next step will reduce the shrinking region to an empty step. In practice, we can achieve this by continuing the iteration until the shrinking region goes to an empty set, and then backing up one step. The entity that remains at this stage is the skeleton.

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 trunion tables have a 110° limit on *a* axis rotation. Similar, most rotating heads have a limit of about 70° of *b* rotation. These limits must be considered during machining. We do so in the cone thinning stage as shown in Figure 4.4 using a process known as *restricted thinning*. Restricted thinning is performed on a primary con-

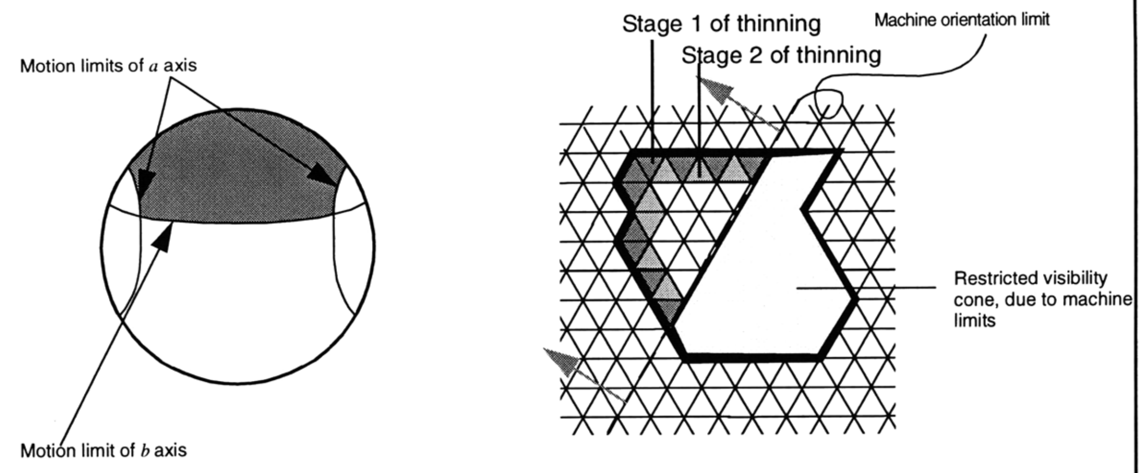

Motion limits of *a* axis
Motion limit of *b* axis
Stage 1 of thinning
Stage 2 of thinning
Machine orientation limit
Restricted visibility cone, due to machine limits

Figure 4.4: Accounting for machine limits

tour, but restricted to a secondary contour. The steps in restricted thinning are identical to those 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.

4.1.3.2 Profiling: Selecting tools and tweaking orientations

Once a reasonable access direction has been determined as described above, our next step is to pick an appropriate size of tool and to "tweak" the access direction to ensure that no collision takes place. We do this with a geometric test that we refer to as *profiling*. The purpose of profiling is to ascertain what the possible collisions are if a tool is placed in the orientation suggested by access analysis as discussed in the previous section. We compute the local profile with a test cylinder consisting of a tessellated surface with embedded tessellated discs as shown in Figure4.5 (a). Since we are interested primarily in local interaction, the profile cylinder should be of a diameter only slightly larger than the largest tool that we are likely to use in roughing. In this sense, profiling can be thought of as a way to determine the shape of the part in the local proximity of the particular tool posture.

When the test cylinder is intersected with the workpiece in the given posture, we obtain a collision profile as shown in Figure 4.5 (b). Typically there should be no collisions because the access direction has already been picked with this consideration in mind. In such cases we can use the largest tool available for that point of the delta volume. However, at the extremities of the delta volume, collisions are not unlikely. When they do occur, there are two ways to address to the problem, listed in order of priority are as follows. The first option is to tilt the tool to avoid collisions. After this, the only option may be to assign a smaller tool. It is also possible that a voxel cannot be accessed in this setup by one of the tools available. Algorithms for evaluating these options are given below.

Pseudo code2: Tool-Profiling

```
Given:
    A triangulated tool model(TM) divided into even number of zones in the circumferential direction
    A triangulated object model(OM)
    preliminary estimate of the tool orientation and machining position

Pseudo code

step 1: Position the tool at the machining position in the given orientation.
```

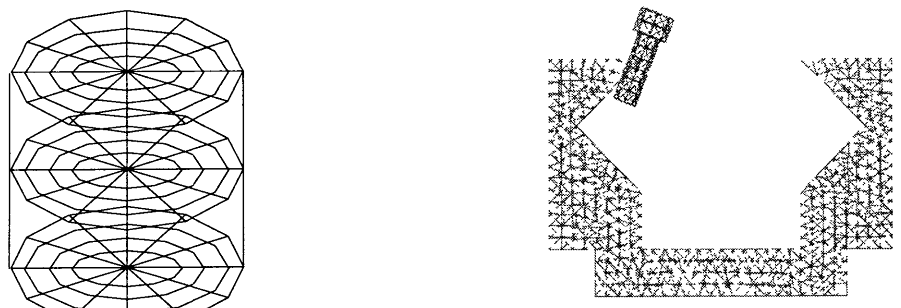

a) Profiling cylinder b) Collision profile

Figure 4.5: Geometric profiling

step 2: Check if the tool intersects with the model and mark all the triangles of the tool that intersect with the model. Find the zones to which these triangles belong and determine the depth of penetration in each zone, given by the radial distance of the centroid of the innermost intersecting triangle from the tool surface. The correction angle is a function of the penetration depth and the distance of the innermost intersecting triangle from the bottom surface of the tool.

step 3: If diametrically opposite zones have intersecting tool triangles then it is most likely that there is no escape direction for that tool. So we can reduce the tool diameter and start the iteration from step 1. A good estimate of the new reduced diameter can be derived from the depth of penetration of the tool that fails.

step 4: For every zone, determine the angle of correction and the axis of rotation such that the collision is avoided.

step 5: Evaluate the composite rotation of the tool as a result of contribution from different zones and reorient the tool. In order to maintain the machining point at the same plane, the tool is either pushed/ pulled along its axis.

step 6: An interference check is made with the tool at its new position.*If there is any interference then goto step 2 and continue the iteration, else return the orientation and the tool geometry.*

Tool selection: The output of profiling is a maximum sized tool assignment and a finely tuned orientation for every voxel in the roughing slice.

Interference checking: A particular motivation for the local profiling method described above is that, while large scale intersection checking is very expensive computationally, local intersection checking is very much a tractable problem. In recent years several algorithms for bounding box based collision checking have emerged. Two of these, the Oriented Bounding Box (OBB) Method and the Discrete Orientation Bounding Boxes (DOBB) Method are extremely efficient. Empirically, the authors of this research have observed that the performance of such methods is highly nonlinear with the number

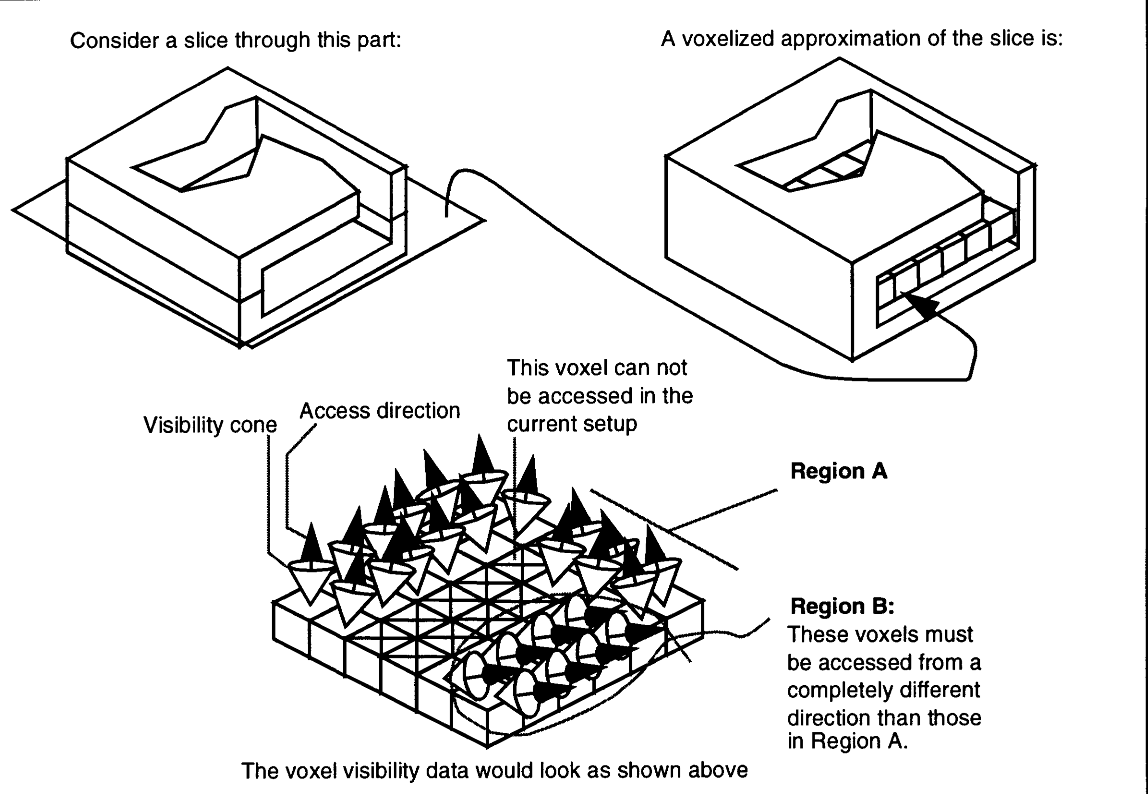

Consider a slice through this part:
A voxelized approximation of the slice is:
This voxel can not be accessed in the current setup
Visibility cone
Access direction
Region A
Region B:
These voxels must be accessed from a completely different direction than those in Region A.
The voxel visibility data would look as shown above

Figure 4.6: A slice with access information

of collisions. In our approach, because the localization of the profiling check ensures few — if any — collisions, we use these algorithms to their greatest advantage.

4.1.4 Voxel-to-voxel transition

At this stage, we have an access direction field for a slab of voxels as shown in Figure 4.6. Of course, not all the voxels may be accessible in the setup. We refer to inaccessible voxels as dead voxels. The task now is to generate tool paths that traverse all the live voxels. The questions we ask in this section are the following:

- Given two adjacent voxels, is it possible to interpolate a tool path between them?
- If so, what is a legal path?

These issues arise because of the possibility of discontinuities in the configuration space, as discussed in Section 3.3.2. For example, in Figure 4.6, it would be disastrous to attempt to interpolate a tool path directly from a voxel in Region A to a voxel in Region B,

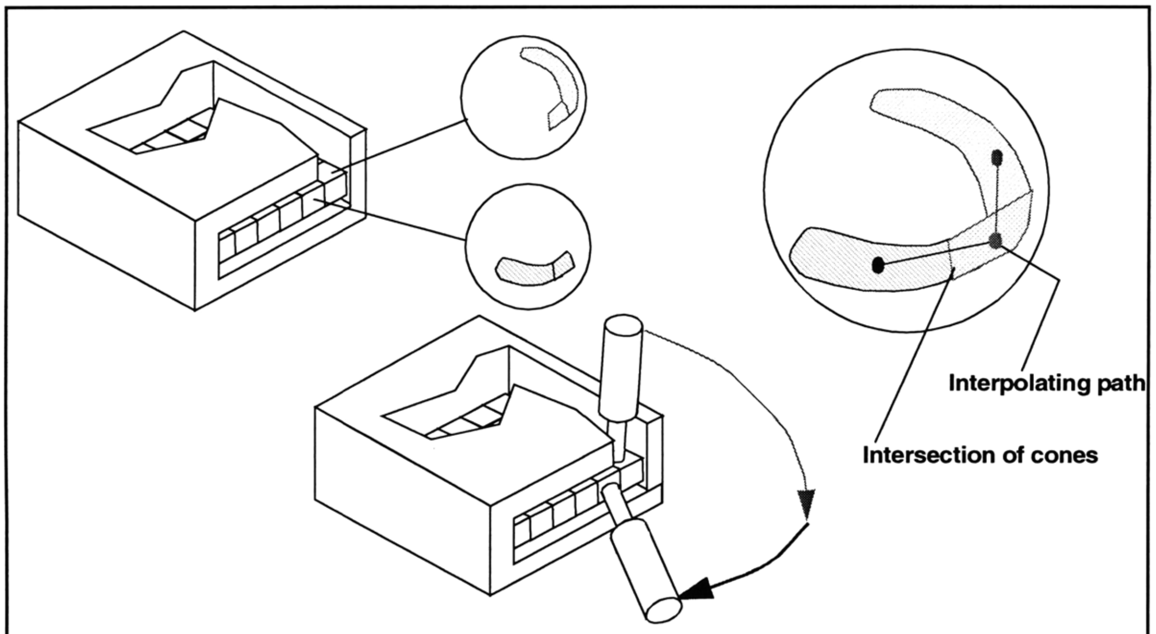


Figure 4.7: Voxel-to-voxel transi-

as it would probably collide with the overhang above Region B. Unfortunately, in our discrete approximation of the configuration space, a precise judgement of discontinuities in the configuration space is impossible. In fact this problem occurs in other areas where discrete sampling is required. Therefore, we must rely on an approximate or heuristic check. We describe such a heuristic below.

Continuity heuristic: We will say that configuration space is continuous between two voxels if there is a intersection between the visibility cones of the voxels is non-zero, and exceeds a certain threshold. Formally, we state that tool paths between voxels v_1 and v_2 can be interpolated if their Visibility cones $V(v_1)$ and $V(v_2)$ satisfy if the solid angle of the cone $(V(v_1) \cap V(v_2))$ is greater than some threshold θ. Furthermore, we will state that the interpolated path must pass through the skeleton of the cone $(V(v_1) \cap V(v_2))$. This is shown in Figure 4.7. If the intersection of the visibility cones of two voxels is an empty set, then it follows that a tool path cannot be interpolated between the voxels. In other words, the tool must be retracted if we want to transition between these two voxels.

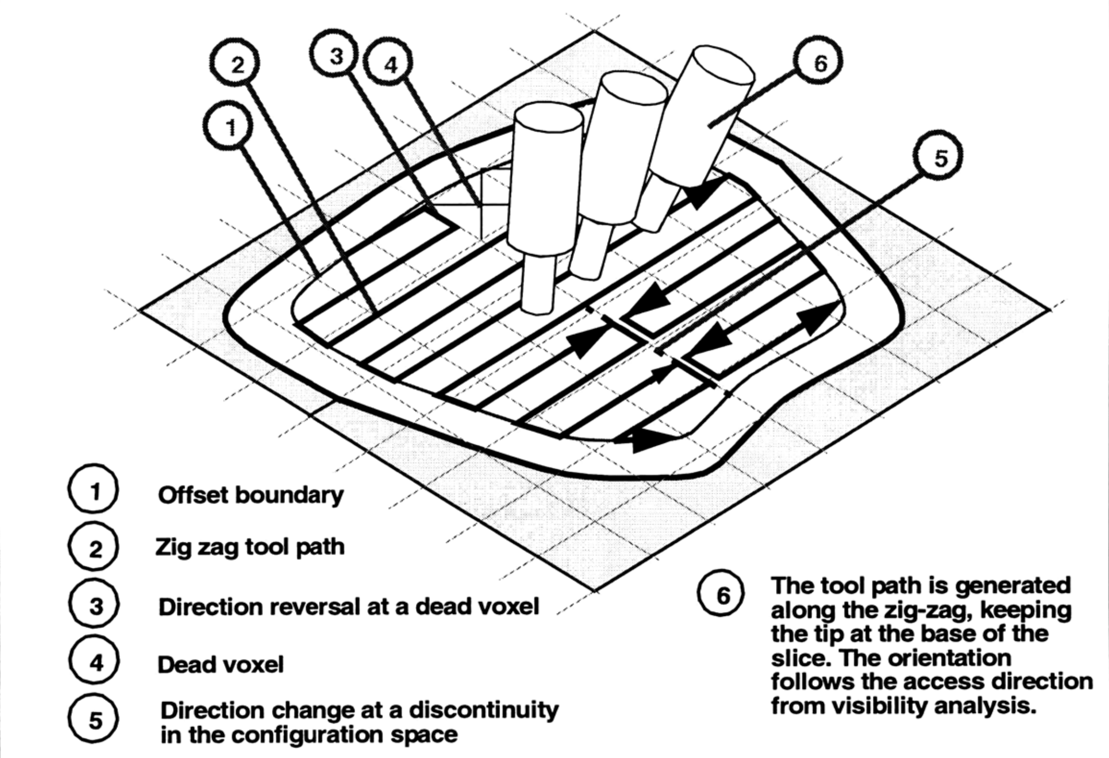

1
2
3
4
5
6
1 Offset boundary
2 Zig zag tool path
3 Direction reversal at a dead voxel
4 Dead voxel
5 Direction change at a discontinuity in the configuration space
6 The tool path is generated along the zig-zag, keeping the tip at the base of the slice. The orientation follows the access direction from visibility analysis.

Figure 4.8: Generating tool paths

4.1.5 Putting together tool paths

We have now computed how to orient the tool in each voxel, and how (and if) the tool can move from voxel to voxel. Our final task now is to string together this information to generate a large scale tool path that sweeps the entire delta volume. Our approach will be to generate a zig-zag tool path. We list the steps below:

1. The first step in our is to create an offset curve to the boundary of the slab as shown in Figure 4.8 ①. The offset depends on the angle of the side-boundary surface of the slab. Unlike in 3-axis machining, in which the tool is always vertical, and the cross-section a circle, in 5-axis machining the cross section must be abstracted as an ellipse. In our initial analysis, we will tilt the tool to be parallel to the side face of the slab. In fact, because of the profiling test we described earlier, the accuracy of the offset is not critical.

2. Within the offset region, we now which voxels are dead. These voxels are not accessible, and should be removed from the slab as shown in Figure 4.8 ③. Similarly, we know which pairs of voxels have no direct transition. As shown in Figure 4.8 ⑤, these discontinuities can marked as boundaries within the slab.

3. Now within the offset region, our approach is generate zig-zag tool paths. The first step

here is to select a zig-zagging direction, and to determine a tool step-over distance based on cutting considerations. The task at hand is to generate a zig zag tool path within the offset region. The zig-zag pattern must avoid all the dead voxels, as well as not cross any of the discontinuities. This is a fairly straight forward task with parallels in scan conversion, and we do not enter into detail here.

In summary, we have now generated complete tool paths. Starting from visibility data, this section traced how to generate access direction and tooling guidelines, interpolate voxels and generate tool paths for each slab. The output of these algorithms is center-line data for roughing an arbitrary shape.

4.2 Finishing: shapes, tools and admissible tool orientations

In machining, the shape of the machined workpiece is defined by (1) the shape of the tool and (2) the path and orientations 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. For example, if we consider a large plane surface to be milled by a face-mill as shown in Figure 4.9(a), the only possible tool orientation is normal to the surface. This *overrides* the actual accessibility cone, which for a plane face is an entire hemisphere. In other words, collision avoidance is not the only consideration in determining a cutting direction — the intricacies of generating a surface can not be ignored in our general approach. We refer to the orientations dictated by the cut, as opposed to those dictated by access, as *admissible* directions. We refer to directions that are both accessible and admissible as *permissible*, and these are the ones that finishing should ideally be performed from. Like access directions, admissible directions and permissible directions can also be represented as cones in spherical Maps. Below we explore admissibility in greater detail. It should be noted that admissible considerations are only really pertinent at or near the surface of the embedded design, and not in the interior of the delta-volume.

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 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 4.9(a). Face milling of plane faces, for example, can be performed by either a specialized face-mill or fly-cutter, or by

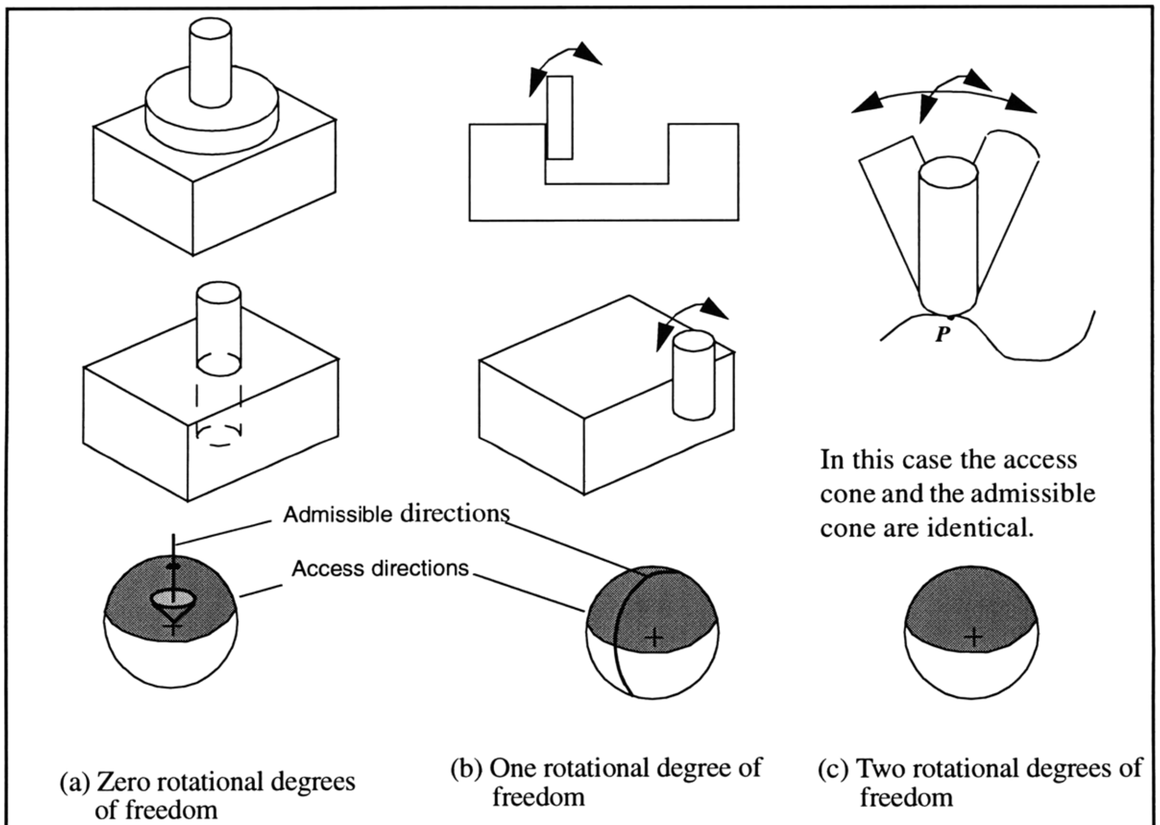

Admissible directions
Access directions
P
In this case the access cone and the admissible cone are identical.
(a) Zero rotational degrees of freedom
(b) One rotational degree of freedom
(c) Two rotational degrees of freedom

Figure 4.9: Tools, shapes and admissible directions

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 4.9(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 4.9(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 one of the surfaces as shown in Figure 4.10. 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 4.10.

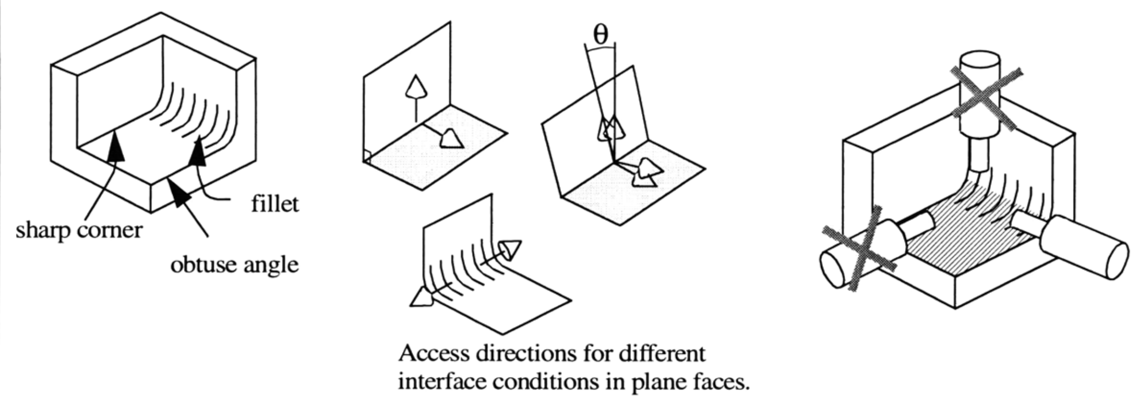

θ
fillet
sharp corner
obtuse angle
Access directions for different interface conditions in plane faces.

Figure 4.10: Edge conditions

4.3 Face-based finishing

After the component has been reduced to near net shape using global roughing, it can be finished to achieve the required surface and form accuracy. Finishing is usually performed with small, accurate tools, and at a light cutting rate. A face-based finishing strategy is proposed in this research. The central idea is to target each face for finishing independently without necessarily grouping them into features. This strategy is being pursued for two reasons: firstly, it bypasses the problem of feature recognition, and secondly, it enables the system to handle shapes that can not be expressed in terms of classical features. Global roughing and face-based finishing are the key ideas of art-to-part machining.

Milling tools can be used in four machining modes: surface milling, peripheral milling, face milling and shape milling. In surface milling, typically performed with ball-nosed or bull-nosed end mill, the profile of the work piece is generated entirely by the path of the cutting tool. In other words, the shape of the tool is not used to impart shape to the component. In peripheral milling, however, the side of the tool is used to impart flatness to the workpiece. Peripheral milling is ordinarily used to machine either flat or cylindrical. In face milling, the bottom of the tool is used to impart flatness to the surface. This technique is used only for flat surfaces. Finally, shape milling is the most specialized form of milling, and is used to impart shapes like chamfers, tapers and fillets being machined.

Because of this specialized nature of milling, different types of surfaces must be treated appropriately to maximize performance. In the following section, we describe our approach to tool path planning for flat faces; specifically, we will consider surface in which the curvature everywhere vanishes in every direction.

4.3.1 Plane faces

When surface finish requirements are reasonably stringent, plane faces must typically be machined by face or end milling. We refer to this as plane face machining, and describe it below. Since smooth plane faces must only be side milled or face milled, the access-direction for machining needs to be either parallel or perpendicular to the face. Furthermore, the entire face must be machined from the same direction, in the same setup. Discontinuous tool paths leave dwell marks and seam lines. Together, these criteria restrict the ways in which flat faces can be finished.

If the surface finish requirements on a nominally plane face are *not* stringent, then it can be generated by profile milling with a ball-nosed end mill. Profile milling is advantageous because it offers a larger range of accessibility. This freedom can be exploited to reduce the number of setups in machining. Rough surfaces can therefore be treated as curved surfaces for the purposes of tool path generation — however, we do not discuss surface machining in this paper. For more information, the reader may refer to [Lee 95].

Using edge conditions: The first criterion that needs to be considered in determining which direction a face can be accessed from is the condition at the edge between the face and its neighbors. This edge condition can be an acute angle, an obtuse angle or a fillet, and can limit the possible perpendicular/parallel access directions as shown in Figure

4.10. The least restrictive edge conditions are obtuse angles. For example, the top surface of an exposed boss can be accessed from every parallel and perpendicular direction. Sharp edges, however, dictate that the tool approach direction be perpendicular to the edge. Fillets can only be accessed along the edge. Together, these conditions restrict the number of access directions in to machine the component.[1]

Picking a tool: The face to be machined will nearly always neighbor a portion of the delta volume that was roughed. A tool diameter would already have been picked while generating the path for roughing. By querying neighboring voxels to the face as to which tool they were roughed with, it is possible to pick the maximum size of the tool that can be used for finishing. After an access direction is picked for finishing, as described below, the finishing process must be simulated to ensure access.

Assessing parallel access: Edge conditions limit the number of directions from which a face may be accessed from a parallel direction. The search space can be further pruned by using the results of visibility analysis. In most cases, this information is enough to either eliminate or reduce the number of possible access directions to a unique option as shown in Figure 4.10. If not, however, a third pruning step can be taken using a simple heuristic: the "depth" of a face must not exceed the length of the longest finishing tool. The depth of a face from a parallel direction is the height of the bounding box aligned with that direction. Furthermore, it is preferable to finish a face from a direction where this depth is minimized. The final set of directions remaining at this point must be checked for access by generating and simulating the tool path.

1. *"Don't care" edge conditions:* Designers often create edge conditions merely to "help" the manufacturer, even if the particular condition is not important to the functionality of the design. In the context of this research we are exploring the possibility of letting the designer use "don't care" edge conditions to denote the lack of a particular preference. This is similar to the feature relaxation techniques studied by previous researchers [Shah 95].

Assessing perpendicular access: A necessary check for perpendicular access is that *all* the tessellations on the surface must be visible in the normal direction. This information is readily available from the visibility analysis data. However, whether a face is actually machinable from that direction can only be determined by simulating the cutting process. Simulation can be performed either with the voxelized model, or by querying a solid modeler for intersection between the sweep of the tool and the part.

Harnessing commonalities: One of the motivations for the feature based approach is that by bunching groups of faces into features, it is possible, for example, to pick a single tool to perform the entire cut. Since the face based approach is fundamentally more "atomic", it is necessary to explicitly identify and exploit commonalities in the cutting plan. Schemes to perform this optimization are currently being developed in this research. One scheme is to consider neighboring faces with the same access direction, and to attempt to pick the same tool as the neighbor. This strategy can also be used in shape features like rounded edges and fillets. If a choice exists then an attempt will be made to pick a face finishing tool that can also impart the appropriate fillet or corner radius at the edge. This grouping is intended merely to optimize the process and reduce tool changes; explicit recognition of features will not be performed.

4.3.2 Finishing cylindrical faces and holes

Faces in which the curvature vanishes everywhere along exactly one direction can be machined by peripheral milling. Such faces are referred to as extruded surfaces, and we will refer to the zero-curvature direction as the principal direction. These shapes can only be accessed for peripheral milling from either side of the principal direction. The principal direction can usually be ascertained from the BRep file. A quick visibility check and depth check can be used to possibly eliminate one of the two access directions. Next, a tool path can be generated and simulated to test whether the feature is indeed accessible.

The special case of drilled holes: Drilled holes, unlike other ruled surfaces like milled holes and profiles, are very special entities; their entire shape, including the adjacent bottom face, is imparted by the shape of the drill. The drill performs most of the roughing and finishing, although an additional reaming operation may also be required. For this reason, drilled holes require special consideration similar to traditional feature based analysis. For every cylindrical surface, therefore, we will first investigate whether: 1) a characteristic shape like a conical bottom or counterbore can be located along the principal axis; 2) the diameter of the cylinder corresponds to standard ream or drill size; 3) depth of the feature corresponds to an available tool. If these criteria are met, then that cylindrical face and all associated entities will be marked for drilling. Furthermore, the convex hull of the drilling operation will be appropriately tagged in the voxel model so that no attempt is made to rough it. Other shape elements, like taps, will also be associated with the hole and appropriate tools will be selected. In this aspect, our approach resembles the technique developed by Regli [Regli 95].

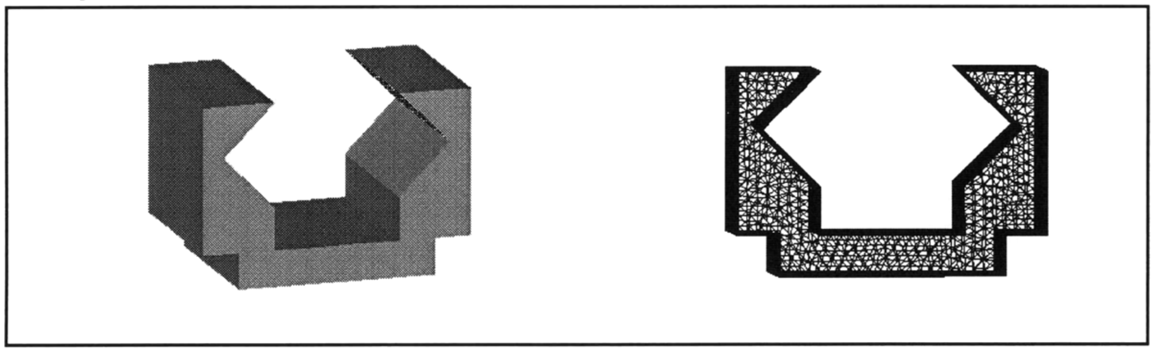
Chapter 5: Examples & Illustrations

The algorithms described in the previous chapters are general, they can be applied to both 3-axis and 5-axis machining. But in case of 3-axis tool path generation, we can take advantage of the restricted DOF to improve the performance of the algorithm. In this chapter we will illustrate both 3-axis and 5-axis machining. We start with a CAD model, perform the visibility analysis, generate the setup directions, and finally generate tool paths.

5.1 3-axis machining

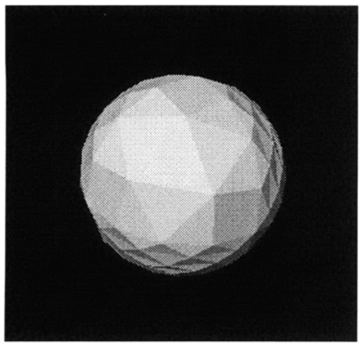
Step-by-step results of three axis machining.

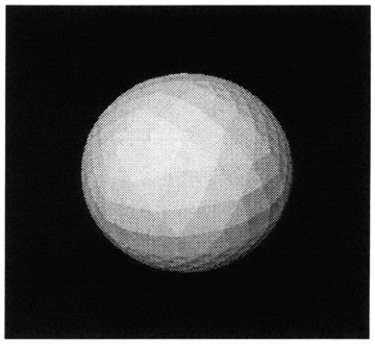
step1 - CAD model: the CAD model and the triangulated mesh were created using ACIS geometric



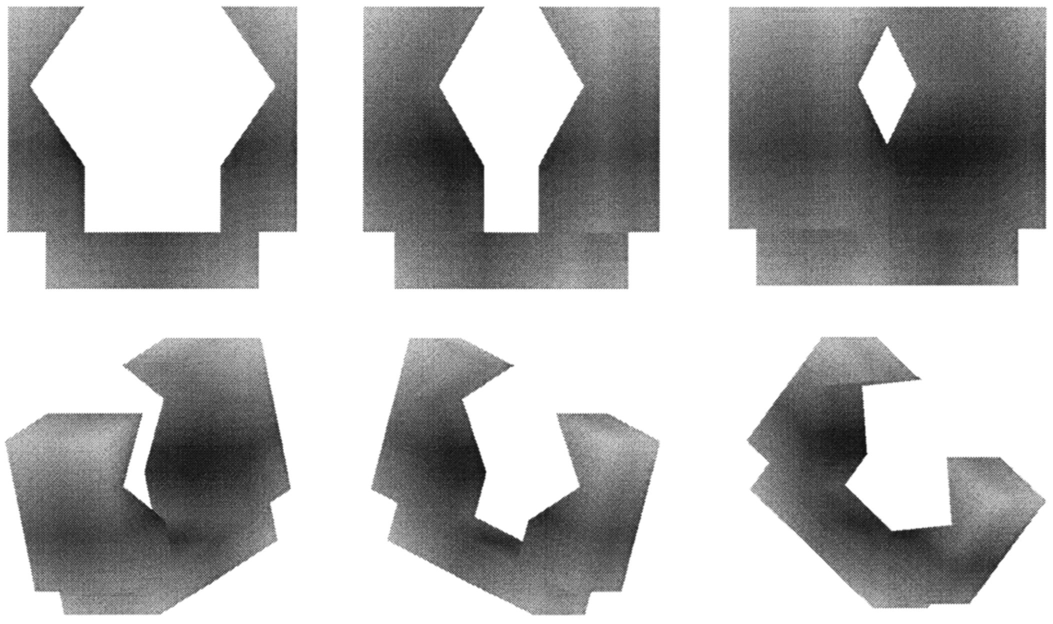
modeler. Each triangle of the model is given an unique color-code to perform the visibility analysis.

step2 - Sampling of the gaussian sphere: Gaussian sphere is approximated using a triangular mesh. The level of discretization can be controlled in our algorithm. The centroid of the gaussian triangles represent the orientations along which the visibility analysis has to be performed. In addition to these orientation some special orientations based upon the surface and edge conditions are added.


sphere approximated with
256 triangles

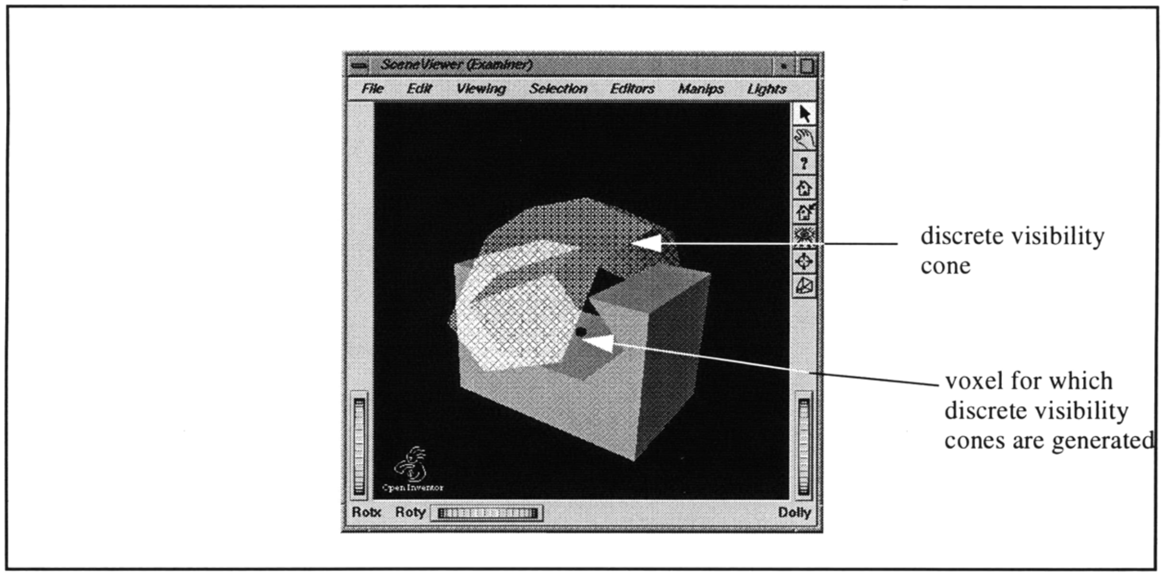

sphere approximated with
1024 triangles

step3 - Performing visibility analysis: The triangulated model is viewed form the set of directions



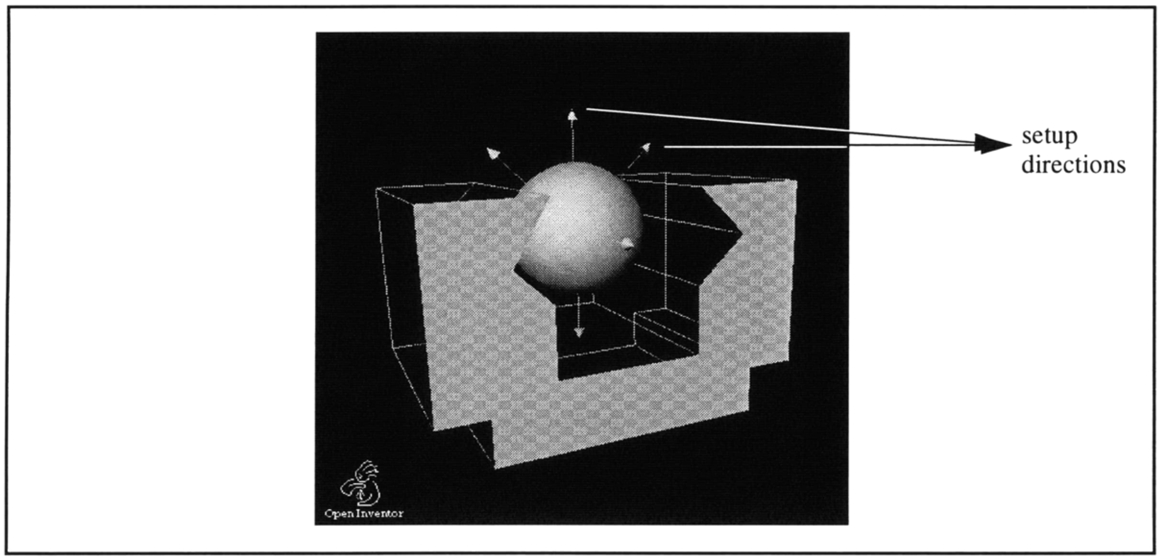
obtained from the earlier step. Based on the colors visible in the scene for every orientation, the triangles visible in that orientation are identified.

step4 - Constructing discrete visibility cones: From the results of the above step, the set of orientations


SceneViewer (Examiner)
File Edit Viewing Selection Editors Manips Lights
discrete visibility cone
voxel for which discrete visibility cones are generated
Open Inventor
Rotx Roty Dolly

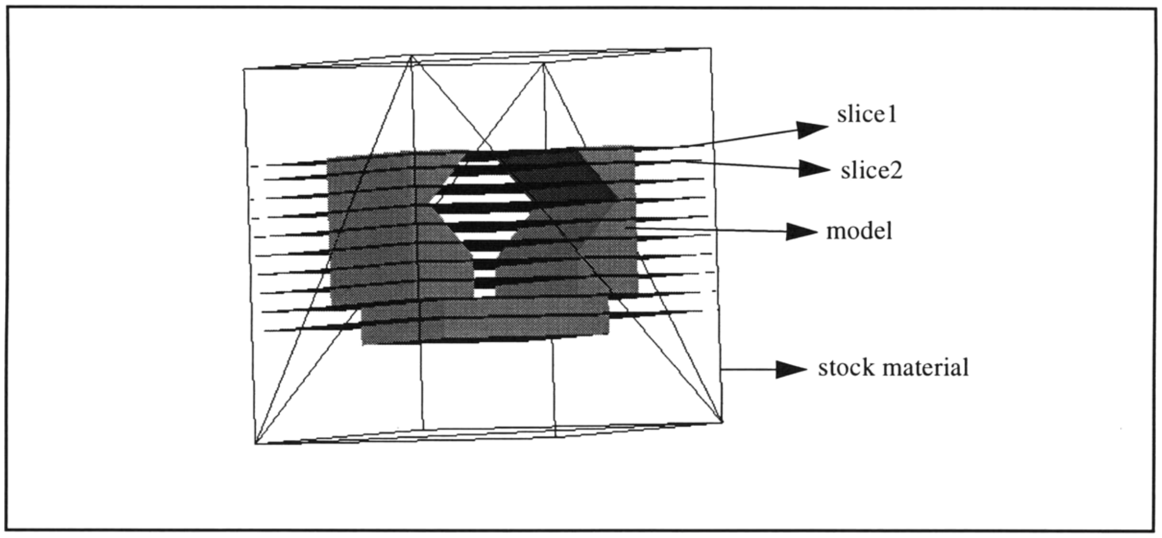
from which a triangle is visible are identified. From the sampling of the sphere (to generate the orientations to do the visibility analysis) we know that each orientation represents a gaussian triangle on the sphere. We generate discrete visibility cones by grouping together all the triangles corresponding to the orientations along which the triangle is visible.

step5 - Generating setup directions: All the triangles with the set of orientations along which they are



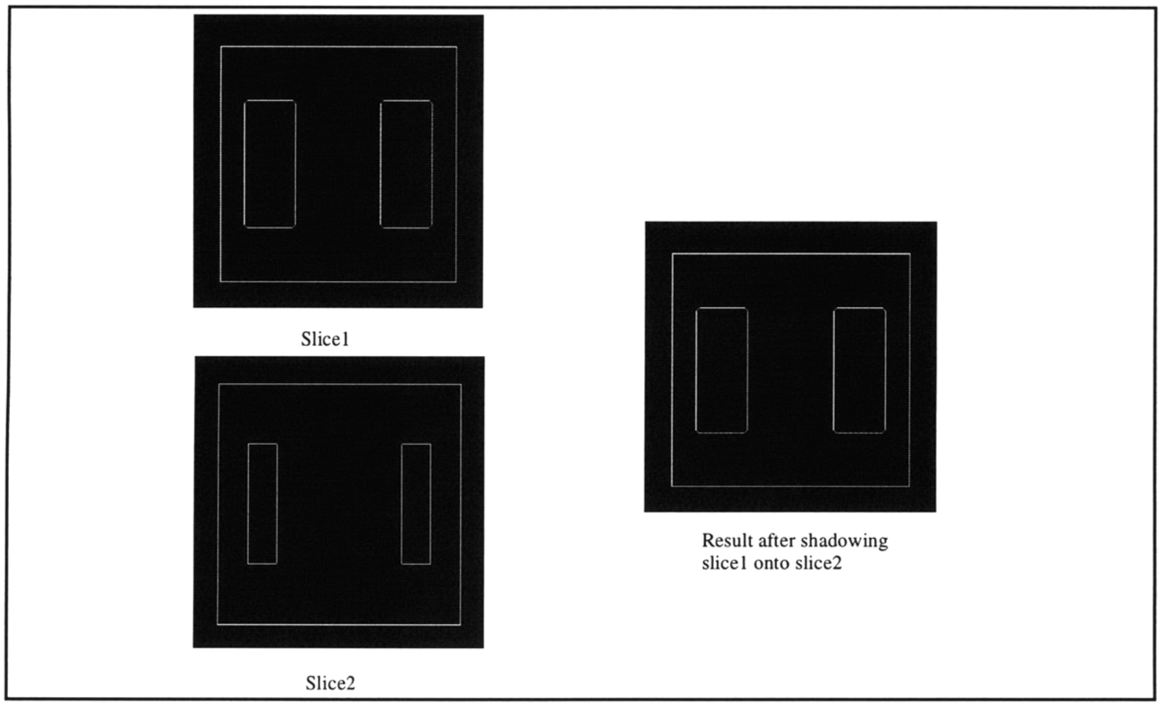
visible are passed through the minimum set cover algorithm (Quine-McCluskey algorithm) to extract the minimum set of directions along which all the triangles of the model can be visualized.

*step6 - Slicing the workpiece perpendicular to the setup direc*tion: Orient the workpiece along one of



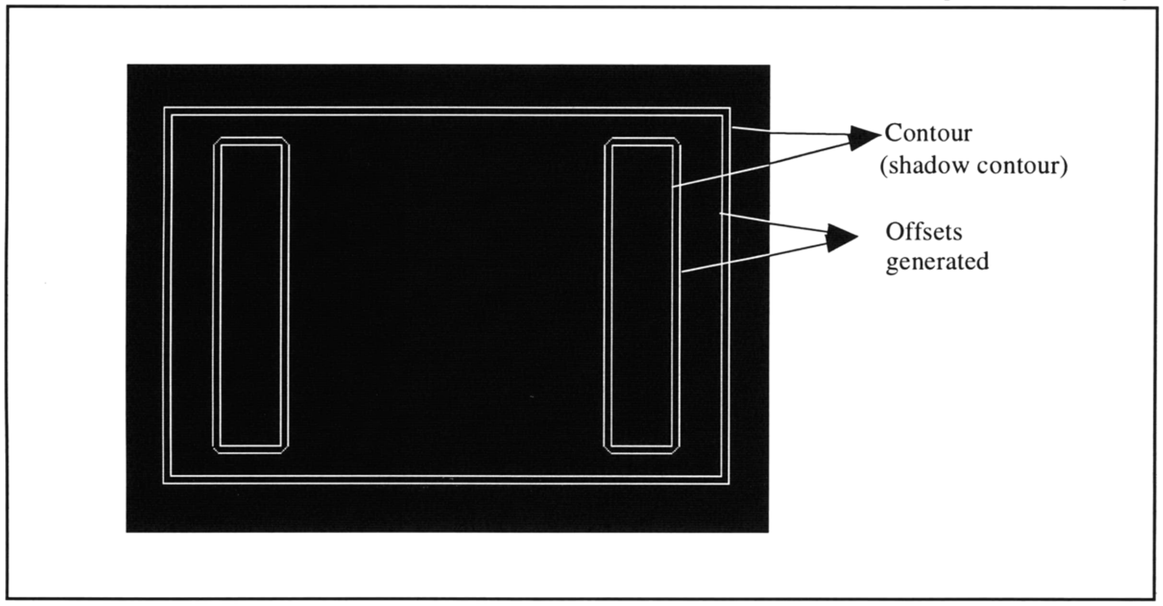
the setup directions obtained from step5. Then slice the triangulated mesh and the bounding box perpendicular to the setup direction. At present these slice planes are equi-spaced, but work is under progress to place these slice planes optimally.

step7 - Shadowing of contours: In 3-axis machining, the tool is always aligned along the vertical direc-



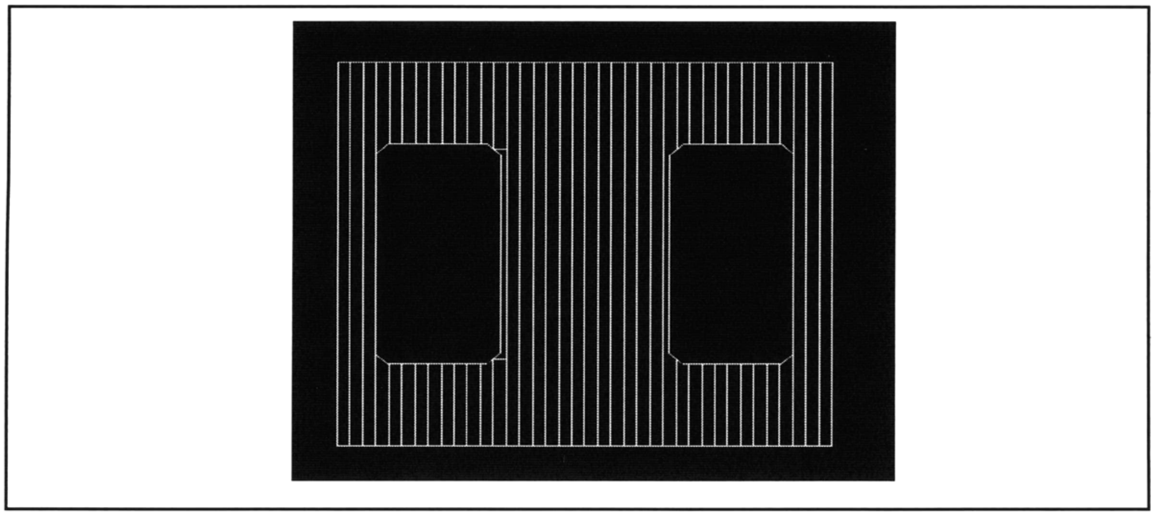
tion. Therefore the amount of area the tool can sweep in a lower slice is at most the common area between that slice and all the slices above it. As shown in the figure in step7, slice1 and slice2 are the first and the second slices along that setup direction. Maximum area the tool can reach in slice2 in that setup is obtained by shadowing slice1 onto slice2.

step8 - Generating offset contours: As mentioned previously, the tool cannot sweep to the boundary of



the contour, as it has a finite diameter. In order to avoid ploughing of the tool into the workpiece, the shadow contour generated in step7 is offseted by tool radius. The material enclosed within the offset contour is machined during roughing.

step9 - Generating Zig-Zag tool paths: We generate Zig-Zag tool-paths for roughing.

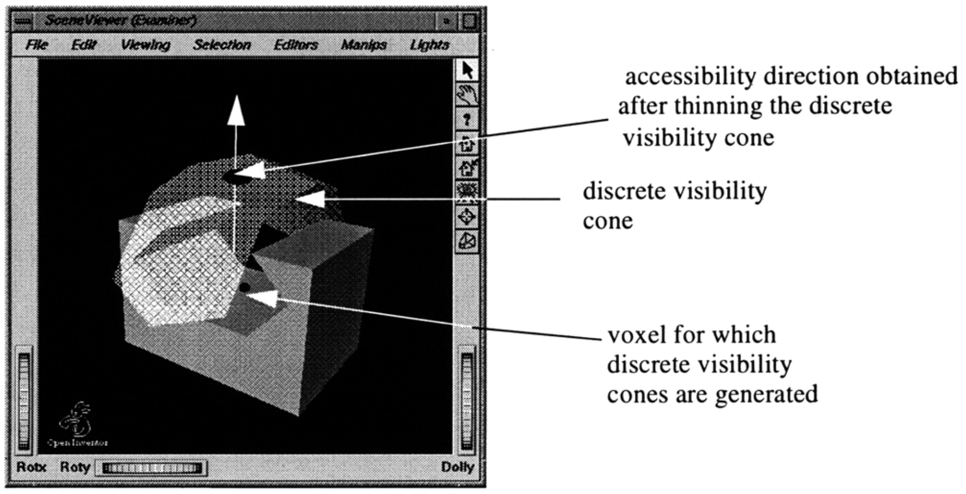


step10 - Face-based finishing: Face based finishing is similar to generating Zig-Zag tool paths for a contour, but they are going to be generated by taking the face boundary as a contour.

5.2 5-axis machining

Step1 through 6 is same for both 3-axis and 5-axis machining.

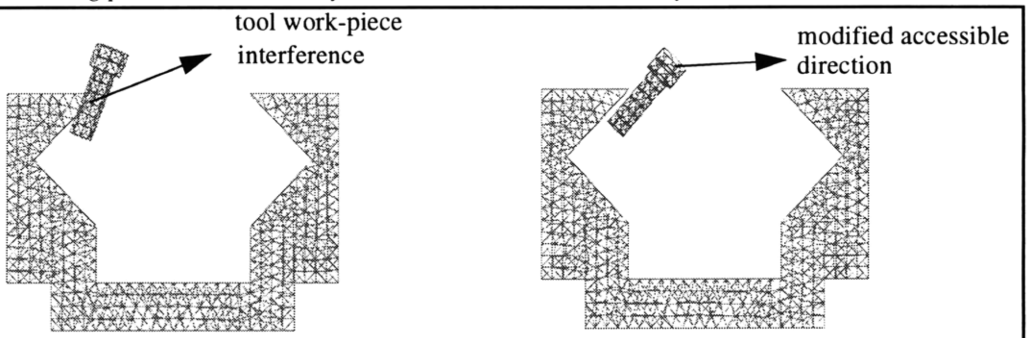
*step7 - Thinning discrete visibility cones to generate probable accessibility direction:*The discrete acces-



SceneViewer (Examiner)
File Edit Viewing Selection Editors Manips Lights
Rotx Roty Dolly
accessibility direction obtained after thinning the discrete visibility cone
discrete visibility cone
voxel for which discrete visibility cones are generated

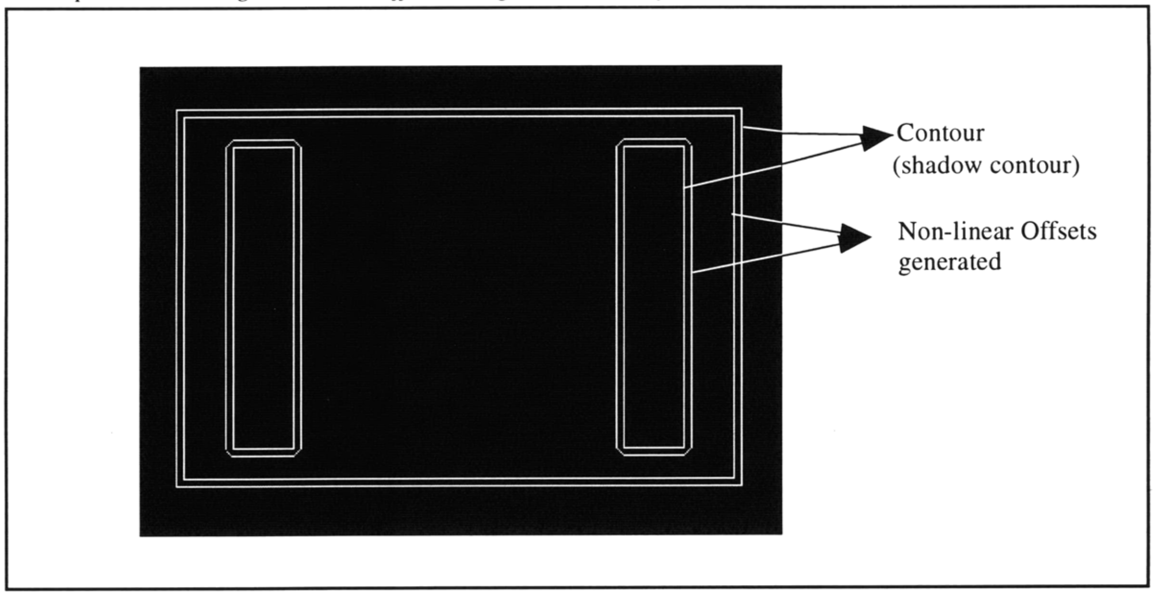
sibility cone is thinned according to the algorithm illustrated in the previous chapter to get the accessibility direction.

step8 - Tweaking probable accessibility direction to obtain accessibility direction



At the extremities of the delta-volume the its likely that the tool interferes with the work-piece when it is oriented along the initial accessible direction. This accessible direction is modified to eliminate the tool work-piece interference.

step9 - Generating non-linear offsets: Legal accessibility direction for each element of the contour is



found in the previous step. This determines the orientation of the tool when it is in the vicinity of that element. The center if the tool should come no closer that r/cos(θ) to eliminate interference. So, non-linear offsets are generated with the offset distance being r/cos(θ), where theta is the angle made by the tool with the setup direction when it is in the vicinity of that element.

step10 - Zig-Zag machining and Face-based finishing: This step is similar to the 3-axis machining except that, here the tool is given an orientation along the tool-path based on the output of step7 and 8.

Chapter 6: Conclusion and Future work

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, "Art-to-Part manufacturing", for CAD/CAM integration. The key idea here is the use of visibility (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. Instead of identifying and generating tool paths for each specialized feature, we developed techniques to generate tool paths for an arbitrary 2D contour. Then repetitively apply this technique to machine the entire workpiece. This approach is an extension of concepts developed by numerous other researchers in the areas of CAM, surface machining and robotics.

We use graphics boards as a special purpose geometrical modeling engine to perform visibility analysis. We take advantage of the fast rendering capabilities of the graphics boards to extract the visible part of the model along a pre-defined set of directions. We then construct visibility cones for all the triangles in the model and various points with in the delta-volume. This visibility data imposes a constraint on the orientation of the tool along the tool path.

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 (Roboticists would say that we are computing the legal configuration space of the machine by a full search). 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 the Art-to-Part approach as the one that is potentially more comprehensive, and one which can overcome some of the limitations of feature-based machining.

6.1 Future work

This thesis is a seed work in developing a fully automated CAD/CAM system. We have modules to generate tool paths for a CAD model once the tool to be used is given. But to develop a system useful to the industry, couple of important modules have to be added: 1) Tool selection strategy, 2) Identifying tool holder interference and eliminating it, and 3) Optimizing the position of the slice planes.

Tool selection strategy: This module supplies a set of tools to be used by the tool path generation routine. By a simple heuristic, it would estimate the time taken by each tool to machine the contours generated by slicing the workpiece. Keeping the surface finish limitations in mind, it would select a set of tools to minimize the total machining time. It would take into account the actual machining time and the time taken for tool changing.

Identifying tool holder interference: Right now we take identify and eliminate the tool-workpiece interference. But while machining deep pockets, its possible that the tool holder might come in contact with the workpiece. To increase the robustness of the system this interaction has to be taken care of.

Optimizing the positioning of slice planes: In our present system, we position the slices at regular intervals. This is common practise in stereo-lithography, for example. Obviously, this would be inefficient, as it would be possible to miss important features such as horizontal faces and inter-surface edges. This can lead to rough cutting with much material left out for finishing. Our approach is instead to place slice planes at critical points on the model, like sharp corners, horizontal faces and sudden changes in slope. This permits us to generate much more “tight” roughing tool paths, and hence achieve better cutting performance during finishing.

This summarizes our technique of tool path generation. The concepts described here have been tested experimentally, as illustrated with the screen dumps in the figures. The construction of a comprehensive Art-to-Part CAM system is underway.

References

[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.

[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.

[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 89]

Choi, B. K. and Jun, C. S. "Ball end-cutter interference avoidance in NC machining of sculptured surfaces.

[Chou 89]

Chou, J. J. *Numerical control toolpath generation for for regions bounded by freeform curves and surfaces*, Ph. D> Thesis, University of Utah. 1989.

[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.

[Elber 94]

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

[Faux 81]

Faux, I. D. and Pratt, M. J., *Computational Geometry for Design and Manufacture*, [Ellis Horwood, Chichester, England. 1978.

[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: Principals and Practice,* Addison-Wesley, 1995.

[Gadh 92]

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

[Gindy 89]
Gindy, N. N. Z. "A hierarchical structure for form features" *Int J. of Prod. Research.*, **27**(12):2089-2103, 1989.

[Gupta 94]
Gupta, S., Kramer, T., Nau, D., Regli, W. and Zhang, G. "Building MRSEV models for CAM applications" *Advances in Engineering Software,* Vol **20** pp 121-139, 1994.

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

[Gurbuz 95]
Gurbuz, A and Zeid, I 'Offsetting operations via closed ball approximation,' *Comput.-Aided Des.*Vol 27 No 11 pp 805-810, 1995.

[Hayes 89]
Hayes, C. C. "Automating Process Planning; Using Feature Interactions to Guide Search," Journal of Manufacturing Systems, **8**:1-15. 1989.

[Held 93a]
Held, M 'A fast incremental algorithm for computing the voronoi diagram of a planar shape,' *Computing with virtual worlds* (CGI'93) Springer-Verlag, pp 318-329, 1993.

[Held 93b]
Held, M *et al* 'Pocket machining based on contour-parallel tool paths generated by means of proximity maps,' *Comput.-Aided Des.* Vol 26 No 3, pp 189-203. 1994.

[Held 1991]
Held, M. *On the computational geometry of pocket milling.* Lexture Notes in Computer Science, Springer-Verlag, London. 1991.

[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.

[Hoschek 85]
Hoschek, J 'Offset curves in the plane,' *Comput.-Aided Des.* Vol 17 No 2 pp 77-82, 1985.

[Hummel 86]
Hummel, K. E. and Brooks, S. L. "Symbolic Representation of Manufacturing Features for an Automated Process Planning System" ASME WAM. 1987.

[Jerrard 89]
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.

[Kuragano 88]
Kuragano, T., Sasaki, N. and Kikuchi, A. "the FRESDAM System for designing and manufacturing freeform objects" *USA-Japan Cross Bridge. Flexible Automation* edited by R. Martin II, 931-938. 1988.

[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, NL. K. Shape Classification in Computer Aided Design, Ph. D. Thesis, Computer Laboratory, University of Cambridge, U. K. 1980.

[Latombe 91]
Latome, J-C. *Robot motion planning*. Kluwer Academic Pres. 1991.

[Laxmiprasad 97a]
Laxmiprasad, P. and Sarma, S. "5-axis pockets: definition and tool-path algorithms" Working paper, 1997.

[Lee 92]
Lees, Y. S., Choi, B. K. and Chang, T. C. "Cut distribution and cutter selection for sculptured surface cavity machining" Int. J. of Prod. Res. 30(6):1447-1470.

[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.

[Loney 87]
Loney, G and Ozsoy, T. "NC machining of freeform surfaces" *Computer Aided Design* **19**(2):85-90. 1987.

[Lozano-Perez 81]
Lozano-Perez, T. "Automatic planning of manipulator transfer movements," *IEEE*

Transactions on Systems, Man 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.

[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.

[Oetjens 87]
Oetjens, T. J. “Automotive CAD/CAM die: the sculptured surface” Proceeings of Autofact ‘87.

[Oliver 86]
Oliver, J. H. “Graphic verification of NC milling programs for sculptured surface parts” Ph. D. Thesis, Michigan State University. 1986.

[Oliver 90]
Oliver, J.H., and Goodman, E.D., "Direct Dimensional NC Verification," Computer-Aided Design, Vol. 22, No. 1, pp. 3-9.

[Rameau 93]
Rameau, J. and Supline, R. “Penetration analysis of solids” *2nd ACM Solid Modeling* (‘93), Montral, Canada. 1993.

[Regli 95]
Regli, W. *Geometric algorithms for the recognition of features from solid models*, Ph. D. Dissertation, University of Maryland.

[Roberts 96]
Roberts, Chell. Personal conversation, Kansas City, Nov. 1996.

[Saito 91]
Saito, T and Takahashi, T. “NC Machining with Z-Buffer Method” *Computer Graphics* 25(1): 207-216, 1991. (proc Siggraph 1991)

[Sakurai 90]
Sakurai, J. and Gossard, D. “Recognizing shape features in solid models” *IEEE Computer Graphics and Applications*, September, 1990.

[Samet 90]
Samet, H. *The design and analysis of spatial data-structures*, Addison-Wesley 1991.

[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.

[Suh 90]
Suh, Y S and Lee, K 'NC milling tool path generation for arbitrary pockets defined by sculptured surfaces,' *Comput.-Aided Des.* Vol 22 No 5 pp 273-284. 1990.

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

[Shah 94]
Shah, J. J., Mantyla, M. and Nau, D. S. *Advances in feature based manufacturing,* Elsevier. 1994.

[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.

[Tiller 84]
Tiller, W and Hanson, E G 'Offsets of two-dimensional profiles,' *IEEE Computer graphics and applications* Vol 4 No 9, pp 36-46, 1994.

[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.

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

[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.

[Wuerger 95]

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

[Yap 87]

Yap, C K ‘An O(nlogn) algorithm for the voronoi diagram of a set of simple curve segments,’ *Discrete Comput. Geom.* Vol 2 No 4 pp 365-393. 1987.

[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.