Numerical Control Tool Path Generation using Space-Filling Curves and Pixel Models

by

Donald Peter Dragomatz

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 1995

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Waterloo to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Two methods for generating numerical control tool paths have appeared recently, one using pixel models, the other using Hilbert curves. The pixel model approach is able to represent many aspects of path generation, but requires large amounts of storage for physically large objects represented at high accuracy. The Hilbert curve approach uses the local refinement property of Hilbert space-filling curves to create an adaptive sampling approach that increases the density of path points only where necessary. In this thesis, I propose and implement a hybrid technique that uses elements of each method, retaining the major benefits of both without incurring the storage costs of the pixel model. The scheme supports the creation of both roughing paths and surface machining paths, but is non-optimal for the later. The addition of containment and exclusion zones to the method leads to a promising paradigm for partitioning object geometry into machinable regions.

Acknowledgements

I am indebted to many individuals and organizations for their help and support: to my supervisor Stephen Mann for guidance and patience while I waded through hundreds of papers; to Richard Bartels, for running CS788 the way he did thus allowing me to get started in this area; to Bruce Simpson, for his many pointers on how research is done; to Greg Veres, for his timely suggestion about frame buffer contents; to Terry Dragomatz, for help in machining prototypes. Ian Bell, Wilkin Chau, Ed Dengler, and Greg Hood deserve thanks for many philosophical discussions as does Rob Kroeger, owner of countless abstract thoughts.

I am grateful to the Computer Science Department and the Computer Graphics Laboratory for helping me attend an important conference at a key point in my reasearch. Thanks also goes to Jim Bandstra of Exco Engineering, a one-time student of mine, for teaching me some of the finer points of machining practice and to John Bisset and Gord England of Durham College for providing facilities for testing my paths. The Institute for Computer Research, UW Scholarships, and The Natural Sciences and Engineering Research Council of Canada have my gratitude for their financial support.

Finally, I would like to thank my friends and family for suffering with me when I was too tired to be of much help, too abstract to make sense or too practical to be abstract. I could not have made it through this project without you.

In memory of Mother, who never doubted …

Contents

List of Figures

Chapter 1

Introduction

It is common practice in present day industry to design objects with the help of computers. Typically, some artifact is modelled on the computer then analyzed using tools such as computer graphics renderings. Many properties of the object can be inspected in this way, but once the object has been deemed acceptable or desirable, the computer model must be recreated as a real world object.

There are several ways of duplicating the computer model. However, if the object is to become part of a mold or stamping die for mass production, then the shape must be re-created in a hard material such as steel. The most common way to construct such metal objects is to instruct a computer controlled machine tool to move a cutting tool through a block of material in such a way that it removes all excess material leaving behind the desired shape.

Creating an object in this manner requires a computer program that specifies the path of the tool tip. Even if a robot or stereolithography apparatus is used, a computer program and path is still required. Hence, the creation of tool paths is a crucial part of transferring the shape defined by computer models to real world artifacts.

Recently, a substantial amount of new research in the area of path generation has appeared in the literature. Two unique methods to come out of this work are the use of pixel models based on raster graphics techniques and the use of space-filling curves. Using pixel models as a basis for path generation allows many aspects of machining to be modelled at path generation time. Unfortunately, these models consume an excessive amount of storage for physically large models represented at high accuracy. On the other hand, space-filling curves have the useful property of local refinement meaning sections of the curve with a given number of nodes may be replaced by a similar portion with a higher number of nodes. This leads to an adaptive sampling method for path generation where computational effort is expended only in areas requiring it.

In this thesis I introduce an approach to 3-axis path creation based on both Hilbert space-filling curves and pixel models that combines the two in a manner that retains the major benefits of both while avoiding the costs associated with large pixel models. The method is point-based, which naturally fits the nodes of the space-filling curves, and initially showed promise as a technique for creating surface machining paths. Although it is useful for generating point-wise and level-by-level roughing paths, it is not a practical scheme for creating surface machining paths. It may be possible to modify the method to use swept surfaces and therefore make it a practical technique but, as discussed in the conclusions, there are problems in extending Hilbert schemes to include swept surfaces. The method is interesting from an intellectual point of view as it offers an unconventional and unusual perspective on the path generation process. As well, a promising and general paradigm for partitioning object geometry and modelling relations between different types of regions is obtained when containment and exclusion regions are added to the method.

The thesis is structured as follows. Chapter 2 consists of two sections, beginning with a tutorial on machining and tool path concepts; this is mainly for the benefit of those from computer science and mathematics who are not familiar with machining terminology. The second section presents and discusses computational and practical issues in path generation. Most of the ideas here are not new, but are found scattered throughout the literature.

In Chapter 3, I discuss pixel models and space-filling curves in more detail. This serves as a foundation for Chapter 4 where I describe my hybrid approach and an implementation. Some sample tool paths are presented here as well, and there is a discussion of some issues arising from the implementation. Chapter 5 ends the body of the thesis with some conclusions and suggestions for future work.

Two appendices are included at the end of the thesis. Appendix A is a classified bibliography containing references to a number of papers related to tool path generation. Papers are collected into groups, each dealing with a common topic. This should be useful both as further background for the material in the body of the thesis as well as for general reference. Appendix B contains a sample input file for the program described in Chapter 4. The file includes comments describing general layout of the file as well as the semantics of the various fields.

In general, SI (metric) units are used in this thesis. However, wherever I have used Imperial (English) cutting tools and refer to objects machined in Imperial units, measurements are reported in Imperial units.

Chapter 2

Background and Issues in Path Generation

This chapter looks at general background material related to path generation. The first section gives a brief overview of the concepts involved in machining. The second and final section presents a number of issues in path generation; it is a mix of ideas from machining practice, mathematics, and computer science. Included throughout the section are references to seminal or interesting papers related to the particular point under discussion. Since there is a large body of literature related to path generation, a bibliography of related work appears in Appendix A. Further references on some of the issues in this chapter may be found there.

2.1 Tool Path and Machining Concepts

The material in this section is tutorial in nature and covers broad topics such as machine tools and basic machining concepts, physical considerations and general machining strategy.

2.1.1 Machine Tools and Machining

A computer-controlled machine tool consists of several elements. There is a *table* upon which a block of material is placed. This block is called the *workpiece.* The workpiece is secured to the table by means of *clamps.* Material is removed from the workpiece by dragging a revolving *cutter* through the part. The cutter is mounted in a *tool holder* that is part of the *spindle.* A mechanical or hydraulic drive unit imparts the rotary motion to the spindle. The act of removing material from a workpiece is called *milling* or, more generally, *machining.* Machine tools that drag a revolving cutter through the work in this fashion are known as *milling machines.*

Cutters are made of hardened steel or similar materials and are roughly cylindrical in shape, though styles vary considerably. Along the sides of the cutter and on at least part of the bottom are sharp *cutting edges.* The cutter can potentially remove material to a depth equal to the length of the cutting edges along the axial length of the cutter. Above the cutting edge is the plain, cylindrical *shank* portion of the cutter, most of which sits inside the tool holder.

The bottom of the cutter may be flat or rounded, as shown in Figure 2.1. A rounded bottom cutter where the filleted portion is the same size as the cutter radius is called a *ball endmill.* A rounded tool with a slight corner radius is called a

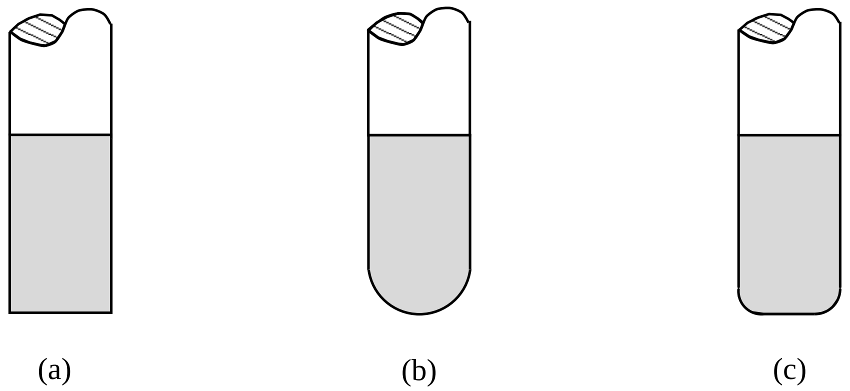


Figure 2.1: Schematic views of different kinds of milling cutters: (a) a *flat endmill*; (b) a spherically tipped, *ball endmill*; (c) a flat endmill having a corner radius or *bull endmill*. The shaded area indicates the relative axial length of the cutting edge.

bull endmill, and sometimes a *toroidal cutter* as well. A cutter with a flat bottom is simply called a *flat endmill*. Cutters may also have tapered (instead of cylindrical) sides and/or angled (instead of flat) bottoms. Flat endmills have the advantage of being able to remove material much more effectively than ball endmills, while ball endmills tend to be the cutter of choice for machining sculptured shapes. Toroidal cutters share the benefits of both, but are more difficult to position relative to a sculptured surface.

Specific shapes are machined by moving the cutter along a pre-computed path in space called a *cutter path* or *tool path*.[1] The path is part of an *NC program*[2] that contains several other elements as well: the speed at which to move the cutter through the work, rotational speed for the spindle, etc. The machine tool computer controller acts like an interpreter, reading program statements and executing them,

[1]In the literature, the word *toolpath* is also used.

[2]NC stands for *Numerical Control*, the traditional name for automated machine tool control.

causing the specified actions to take place. For example, the movement of the cutter is effected by commanding drive motors on the machine to move the table or spindle a specified amount in X, Y or Z. Some machine tools and their controllers have the ability to tilt the cutter relative to the work in one or two axes. Such machines are called *4-axis* or *5-axis* machines if they allow, respectively, rotation in one or two axes. More conventional units, allowing movement only in X, Y and Z, are called *3-axis* machines. Three-axis machines are older, more common and easier to program, but 5-axis machines can create a wider class of shapes.

Terminology used to describe the machining process includes *spindle speed* or *speed* which is the rotational speed of the spindle, *feed rate*, the speed at which the cutter is being moved along the toolpath, and *depth of cut* which is the distance from the end of the cutter to the top of the workpiece. On a given pass, the tool removes a swath of material the maximum width of which is the width of the cutter. The width of the swath together with the depth of cut and length of tool movement determines the volume removed by that motion. Multiplying this volume by the feed rate gives the *material removal rate* for the move. During roughing, one wants to remove as much material as quickly as possible, i.e., aim for as high a material removal rate as possible. While a high removal rate is desirable during finishing too, concerns over surface finish and final form tolerance mean that the removal rate actually used is less than optimal.

An important limitation of contemporary computer-controlled machine tools is that most controllers allow only linear and circular movements to be specified.[3]

[3]In addition, the circular movements must lie in a plane perpendicular to one of the three principal coordinate axes.

Some, however, allow movement along parabolic or cubic curves. A few experimental units permit entire surfaces to specified.

2.1.2 Physical Considerations

Machining is a physical process; velocities, forces and material properties must all be considered when developing an NC program. For example a soft, easy to cut material such as wood or wax will allow the cutter to be moved quickly through the material thereby sustaining a high material removal rate by using a high feed rate and a large depth of cut. As the hardness of the material increases, the feed rate (and hence the material removal rate) must be reduced. If it is not, the cutter is likely to deflect excessively or break, especially in the case of a long thin cutter passing through a piece of hard material. To reduce the load on the tool, the depth of cut, engaged width or the feed rate may be reduced (in some combination). Ideally, the feed rate varies according to tool load and material properties, meaning the feed rate varies on each tool move. Calculating the tool load is non-trivial so in practice a fixed, reasonable feed rate is used for a number of tool moves. This simplifies programming but leads to less than ideal cutting conditions.

Damage to the cutter is not the only problem arising from overloading the cutter. The shape produced is inaccurate, because the controller drives the *spindle* to the indicated coordinates, but the tip of the *cutter* occupies a slightly different position in space due to deflection. Hence the tool generates a slightly different shape as it removes material. Large diameter cutters are more rigid than smaller cutters, and are less prone to deflection, but larger cutters are incapable of fitting

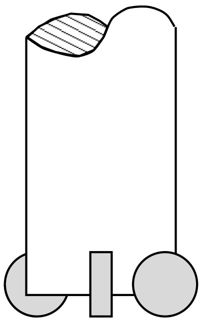


Figure 2.2: Example of a cutter without cutting edges in the middle of the bottom. To move vertically, the cutter needs to be clear of the work, use an existing hole in the work, or move from side to side while moving down.

into areas of high curvature and require higher horsepower spindle motors. The use of small cutters therefore cannot be avoided and tool load remains an important consideration.

Related to tool load and shape accuracy is the issue of tool wear. As the tool cuts the work, the cutting edges are slowly worn away. This means the tool will become dull over time and its shape will change slightly. Cutting with a dull tool increases the load on the tool; in extreme cases, the tool will break. Cutting with a worn tool means slightly less material is being removed than originally planned, which leads to an inaccurate final shape. Thus, for programs with very long runtimes (hours), tool wear becomes a factor.

Another situation that must be avoided is contact between the workpiece and the spindle, tool holder or cutter shank; such contact may again damage the cutter, spindle or workpiece. Likewise, the path must not cause the cutter to collide with clamps and other work holding devices. The cutting edge of a cutter has a relatively

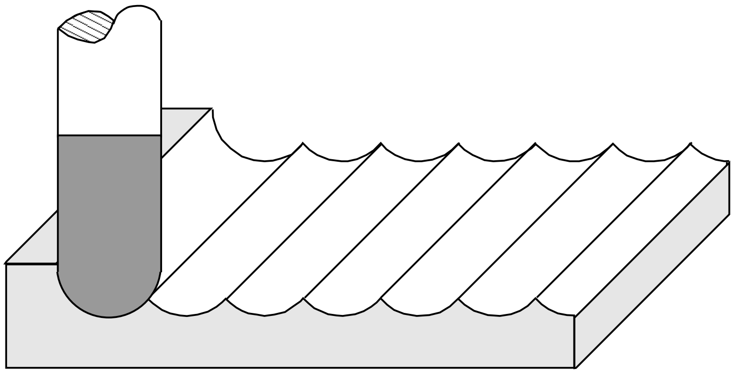


Figure 2.3: Ridges of material created by using a ball endmill. A flat endmill tilted at an angle to the surface gives a broader, elliptical trough and smaller ridges but does not eliminate the problem.

short length, so it is easy to cut too deep and have the shank contact the work. A similar issue arises with certain styles of cutter having cutting edges on only part of the underside and shank. Figure 2.2 shows an example. A cutter of this type can only plunge into the workpiece a small amount before it must move laterally (since it does not have any cutting edges in the middle of its bottom). Also, it cannot take a very deep cut.

A final consideration is that only a few shapes can be cut exactly (to within the accuracy of the machine). Most shapes must be approximated to within some acceptable form tolerance. There are two reasons for this. The first is due to the restricted range of motions allowed by present day computer controlled machine tools. As mentioned in Section 2.1.1, the tool can be moved only along straight lines or circular arcs. Thus, other curves must be approximated by some combination of linear and circular arcs.

The second reason shapes can only be approximated is because the shape of the

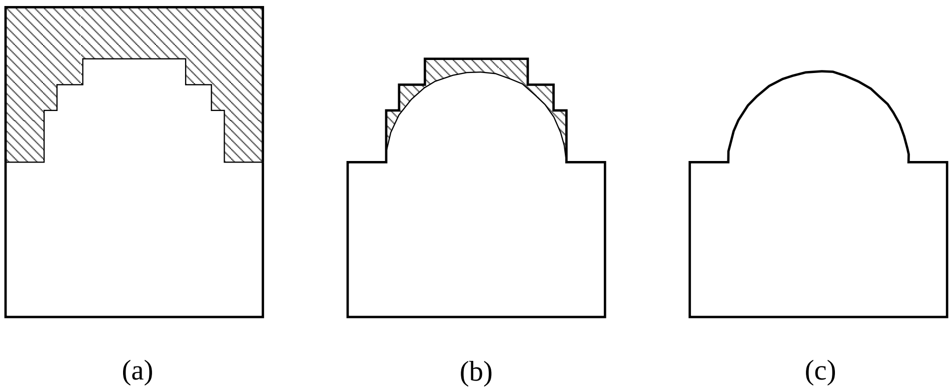


Figure 2.4: Typical machining strategy. Roughing paths reduce a billet of material (a) to an approximation of the final form (b) with a slight amount of excess material. Finishing paths then remove the extra (c) to produce the final shape.

tool almost never matches the shape of the surface. Most sculptured shapes are machined using roughly parallel passes of a ball endmill. This process leaves ridges of unremoved material, called *cusps* or *scallops*, as shown in Figure 2.3. The height of these ridges can be reduced by using a larger diameter cutter and by making adjacent passes of the tool be closer together, but they cannot be eliminated. Some systems make use of a flat endmill and a 5-axis machine to further reduce the scallop height. In any case, unremoved material and use of linear and circular approximations to tool movements mean that machined shapes do not precisely match the original.

2.1.3 Machining Strategy

In general the billet of raw material and the final shape are so different that several programs are involved in removing the material. To begin, a number of *roughing paths* are created, the purpose of which is to remove as much excess material as quickly as possible to create a shape that is slightly larger than the final shape. A roughing path only approximates the final shape, but it must be accurate enough to leave (an approximately) consistent amount of excess material behind, and *never* violate the final shape. Thus, coarse approximations are possible, but a no violation guarantee is important. Each roughing path uses a different size of tool; on physically large objects, there may be more than one path for a given tool, each covering a different area of the object. After all roughing paths have been applied several *finishing paths* are used to remove the remaining excess and create the desired shape.[4] Figure 2.4 shows the process. As in roughing, if the model is physically large, there may be more than one path created for a given size of tool.

The overall strategy of path development is therefore:

- Develop a sequence of as few roughing paths as possible, removing as much material as fast as possible, beginning with the largest cutter possible.

- Develop a sequence of as few finishing paths as possible, from largest to smallest, leaving no more than the specified amount of extra material.

Generally, the number of cutters should be minimized as tool changes are costly in terms of processing time.

A standing assumption in machining with a sequence of tool paths is that the previous path *did remove* the material it was supposed to. It may happen that the

[4]There is another phase known as *semi-roughing* (or *semi-finishing*) that sometimes appears between roughing and finishing. It is used to smooth out the irregularities in the roughly machined shape, leaving a more uniform layer of excess. Also, if the shape being machined is already close to the final shape (as in a cast or molded shape), then neither roughing nor semi-roughing are performed.

previous path was incorrect and left too much material such that the following path overloads and damages the cutter. While it is important that each path do its job, it is often difficult to verify that a sequence of paths is compatible in this sense.

2.2 Issues in Path Generation

Machining is still a craft-like activity. Though paths may be generated on a computer, the strategies and tactics used by the algorithms that create them must produce paths that adhere to accepted, local machining practice.[5] Ideally, systems for generating tool paths should be flexible and use a *policy-mechanism* approach to path generation.[6] However, few works in the literature take this view as they are typically concerned with a particular aspect of path generation. Furthermore, there is no general, clearly stated, accepted collection of path generation issues. Different papers discuss different aspects, meaning a complete set of issues exists only in bits and pieces throughout the literature. Therefore, this section presents a collection of what I believe are the main issues in path generation.

There are many elements in path generation. (Certainly the quick overview in the last section argues in favour of that view.) Nonetheless, there are six major elements, with a host of minor ones within them. The major elements are:

---

[5] Local meaning specific to the organization or individuals concerned with creating and using the tool paths.

[6] The idea of a *policy-mechanism* philosophy appears frequently in Computer Science in the area of operating system design. The idea is that algorithms used in large computer programs (such as an operating system) should not dictate the particular *policy* that users of the system must use. Rather, programs should provide *mechanisms* to allow users to implement their own policy. I think this idea would be very helpful to users of NC path generation systems.

- tool geometry
- machine type and machine limits
- path type and/or purpose
- object representation type
- tool positioning method
- path planning strategy (path topology)

Some issues are straightforward, such as the type of machine and its limitations. Others, such as path planning, are quite complex. Some elements are related to others. For example, the type of tool used is partially determined by the purpose of the path.

2.2.1 Tool Geometry

The tool geometry is synonymous with the type of tool which in turn is partially determined by the type of path. As discussed in the previous section, the axial length of the cutting edge determines the maximum possible depth of cut. If a cutter like that in Figure 2.2 is used, the direction and amount of tool motion may also be constrained.

Sometimes the size of tool is given by the user, but some systems select the tool size based on the geometry of the object to be machined. Some systems develop paths for only one tool at a time while others develop paths for a range of tool sizes.

2.2.2 Machine Type and Machine Limits

A major distinction in the type of machine is whether it is a 3-axis or 5-axis machine tool. For 5-axis machines, only a limited amount of rotation is available in each axis. Exceeding the limits means the path is useless [CPJ93]. Common to both types are limits on the amount of movement allowed in X, Y or Z. Specifying a position outside the legal envelope invalidates the path for that machine tool. (Of course, on a larger machine, the same path would be acceptable.) There are similar limits on spindle speed, feed rate, and radius for circular tool movements.

A few machines allow parabolic curves; an even smaller number permit cubic splines. These higher-order paths are preferable to an equivalent sequence of piecewise linear movements for several reasons. First, they are more efficient in terms of storage for the path itself. A single circular move is equivalent to many small linear moves and hence requires space for only a single instruction. Second, higher-order paths are smoother kinematically and therefore slightly more accurate. For a sequence of linear moves each changing direction, the machine tool must decelerate, stop and change direction. Quite often, inertia carries the tool past the intended point.

The best way to handle machine limits is to include them as data for the path generation program. At present, however, most systems leave machine-specific conversions to a post-processing phase. Hence, it is possible to spend a considerable amount of computational effort to generate a reasonable path that turns out to be invalid for the intended machine tool.

2.2.3 Path Types

There are four general types of tool paths:

- point-to-point
- profiling
- pocketing and roughing
- surface machining

Point to Point Paths

Point to point paths have traditionally been associated with hole making operations such as drilling and reaming. For our purposes, point to point paths are more general. They include the method of roughing where the cutter is plunged vertically into the work at regular, grid-like intervals. This is not like roughing in vertical slices where the cutter cuts on its side; all the cutting is done on the bottom of the tool. This can be done using a chisel point drill or a slot drill.[7] Ball endmills can be used, but such cutters have zero cutting velocity at the middle of the bottom; that portion of the tool does not cut effectively and wears quickly, leading to vibration and chatter. Nonetheless, plunging in this manner has an advantage in that most of the cutting force is applied vertically, where the tool is the strongest. However, it is not a common method in current machining practice and only a handful of papers in the literature even mention it.

[7] A flat endmill having cutting edges running all the way across the bottom of the cutter.

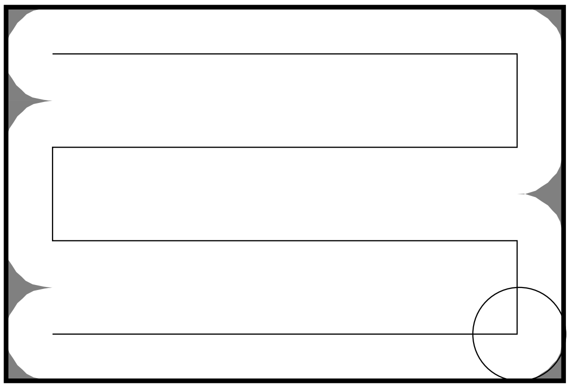


Figure 2.5: Unremoved material in pocketing. The material must be removed by passing the cutter tangent to the edges having excess. Material in the corners cannot be removed except by using a smaller cutter.

Profile Paths

Profiling is moving a cutter along a curve. Profile paths occur as part of pocketing and multi-layer roughing, especially if the region has been machined with a zigzag path. A final cut is required to remove the scallops of material left on the walls of closed profiles, as shown in Figure 2.5. If left uncleared, they may accumulate and causes excessive tool load, or interfere with non-cutting portions of the tool.

A distinction in profile machining is whether the cutter cuts *on* or *tangent to* the profile. When a cutter moves on a curve, it removes material on both sides of it. When a tool cuts tangent to a curve, it removes material on only one side. Profiles that require the tool to cut tangent to it can be handled by first offsetting the profile by the radius of the cutter then cutting on the offset. Figure 2.6 shows the two kinds of profile paths.

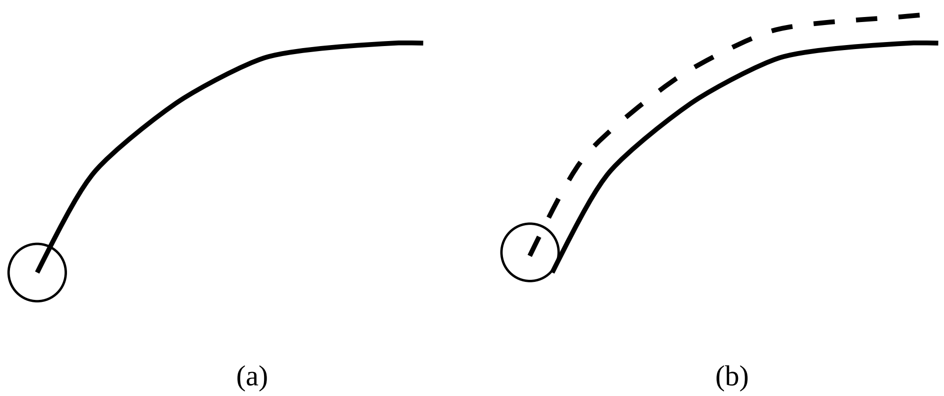


Figure 2.6: Cutting on a profile versus cutting tangent to it. In (a), the cutter cuts on the profile by moving the tool centre along the given path. In (b), the tool cuts tangent to the given curve by moving along a curve that is offset by the tool radius.

Pocketing and Roughing

Pocketing and roughing are sometimes thought of as two types of path but several authors in the literature [CC89a, Guy89, Hel91, LDV94] treat them as essentially the same.[8] Pocketing tends to be $2\frac{1}{2}$ dimensional in that 2D machining occurs at a fixed Z depth. Roughing involves repeating this for each of several decreasing Z levels.

In addition to requiring a collection of paths for each layer, roughing also raises the issue of *layer connectivity*, or how to connect the end of one layer with the start of the next. This can be non-trivial in the case of a cutter with little vertical plunging ability like that in Figure 2.2.

There are two styles of cutting commonly used: contour-parallel, and zigzag.

---

[8]The term *roughing path* includes any style of path used for roughing. The term *pocketing* refers to machining in a region bounded by steep, unremoved walls, usually called a *pocket*. The roughing paths that are similar to pocketing paths are ones that cut on a sequence of decreasing, horizontal levels.

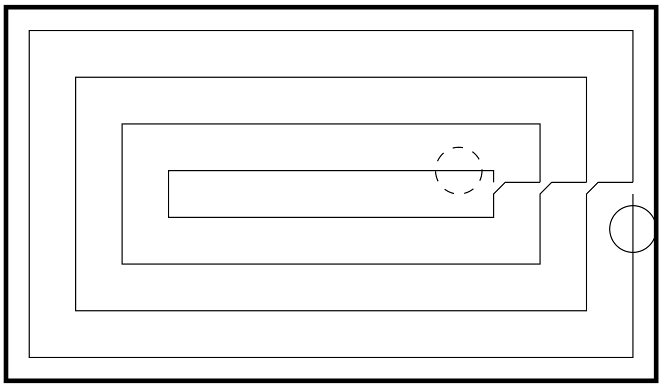


Figure 2.7: A contour-parallel pocketing path. The tool starts near the dotted circle and moves outward until in reaches the end. The small chamfers near the start of each pass are for illustrative purposes and would not normally appear.

Zigzag pocketing is shown in Figure 2.5. Contour-parallel pocketing involves moving the cutter roughly parallel to the boundaries in a somewhat spiral fashion as shown in Figure 2.7.

There are two approaches used to compute the passes of the tool for contour-parallel paths. One method uses Voronoi diagrams [Per78, CC89a, Hel91], the other is based on offsetting consecutive pairs of boundary elements. For the Voronoi-based approach, the overall 2D region must consist of lines and arcs, and needs to be decomposed into simpler, convex regions. For the contour offset approach, the outer profile is offset inwards by the tool radius. These offsets form passes of the tool. Linking them is not always trivial. Tests must also be performed to check for undesired offset degeneracies and loops. The next pass of the tool is found by offsetting inwards until the offsets degenerate to a point. Regardless which method is used, the tool usually enters the work in the middle of the pocket and cuts outward toward the boundary.

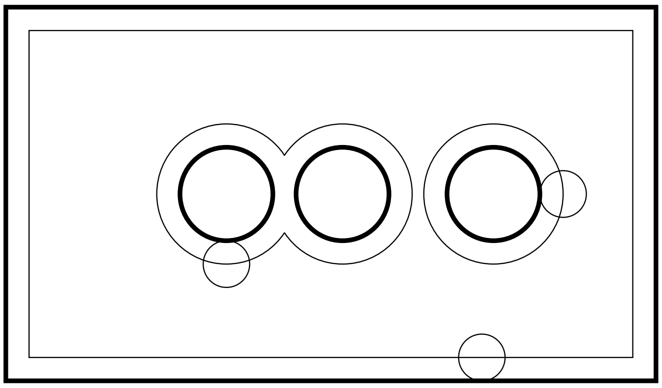


Figure 2.8: A pocket with islands. Only the final profile paths are shown. The cutter cannot pass between the two islands on the left, so the two paths to machine their profiles merges into one.

Typically, pocketing and roughing involve cutting around regions where the cutter is not allowed to enter called *islands*. In some cases unconnected islands may be separated by a distance smaller than the diameter of the cutter. In such cases, the tool cannot pass between them, and the two islands essentially merge into one. Figure 2.8 gives an example. The shape of islands can be arbitrary and many of them may be present in the region to be machined. For this reason, creating a pocketing path is a non-trivial operation.

For zigzag pocketing, the cutter makes alternating, parallel passes back and forth inside the pocket. The actual endpoints of the path are found by intersecting the planned passes of the cutter with an outer profile offset inwards by the cutter radius. In the cases of islands however, there is an issue of *region connectivity*. The cutter sweeps along until it bumps into the obstruction. At this point it can either:

- lift, move over the island, plunge down and resume cutting on the other side,
- move around the edge of the obstruction and resume cutting on the other

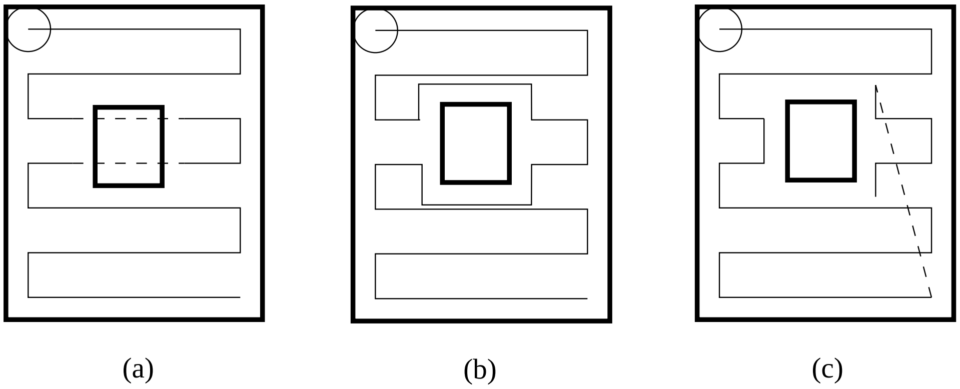


Figure 2.9: Region and pass connectivity in pocketing with islands. In (a), the tool is lifted over the obstruction; in (b), the tool moves around it. In (c), a new region is formed and cut after the current one is completed.

side,

- stop moving on that pass, move over and begin cutting the next partial pass.

The three strategies are shown in Figure 2.9. The first case can be very inefficient as the tool can spend a lot of time moving over the island. More importantly, the tool might not be of a type that can plunge vertically into the work. This means that access holes need to be drilled for each plunge location (unacceptable), or the cutter must spiral or ramp down to resume cutting at the current level. In any case, interrupting the cuts in this manner is often undesirable. The second method is acceptable if the obstruction is not too big, otherwise the tool will spend a lot of time moving around the island, passing over already cut portions of the object. Also, the tool will quickly become dull by rubbing on the already cut material. (It is possible to move the cutter away from the work slightly then move it forward at high speed to resume the interrupted cut.[9]) In the last case, an uncut region

[9] After making sure one such cut has already been made at cutting speeds.

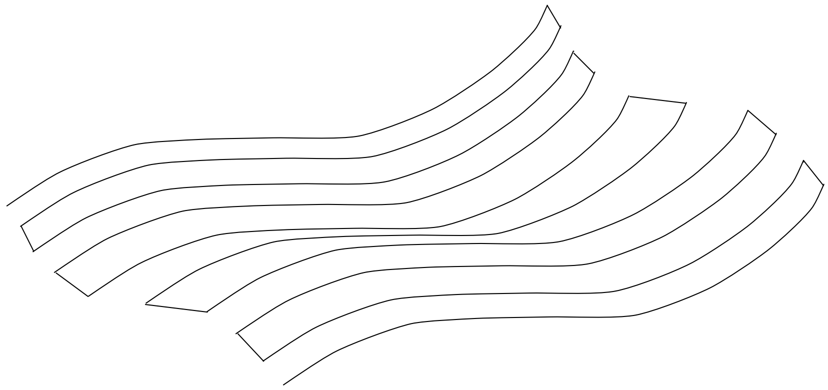


Figure 2.10: A typical surface machining path.

remains. This is not a problem, but it does require a mechanism such as a stack to remember them. Moving from the end of one region to the start of another also raises the tool engagement issue, though not as often as in the first case.

The method of roughing horizontally, layer by layer, is quite common. However, it is also possible to rough machine by removing vertical layers of material (as noted by Marshall and Griffiths [MG94]). Unfortunately, this requires a long cutter cutting on its side (rather than on its end). Hence, it is not a preferred method.

Surface Machining

Typically in roughing or pocketing, the tool cuts around a number of obstacles, but at a fixed Z depth. In surface machining, the situation tends to be the reverse: the cutter cuts back and forth across the surface, at a Z depth controlled by the shape of the object with few if any obstacles present. Also, only a relatively small amount of material remains to be removed. Figure 2.10 shows a typical surface machining path.

Another method, used for steeply sloping surfaces, is to cut in a top-down

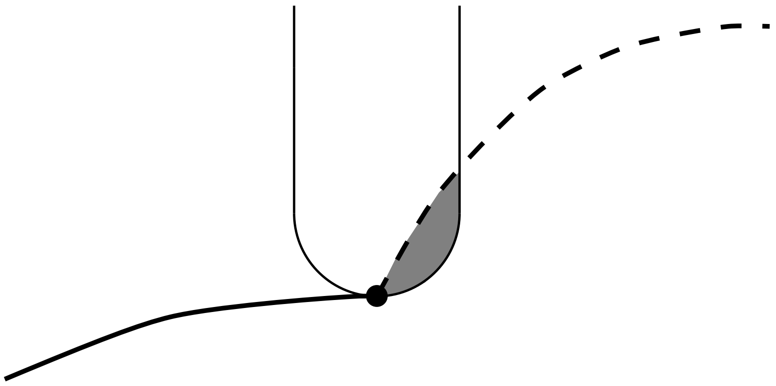


Figure 2.11: Gouging of a neighbouring surface. The tool unintentionally removes material from the surface on the right while cutting near the edge of the surface on the left.

manner. The tool machines on the surface at a specified level, then descends to the next level and repeats the process. The technique is similar to level by level roughing except that the tool always stays in contact with the final surface (the tool does not need to clear away material far from the surface since it has been removed during the roughing pass). Careful attention is paid to the distance between levels as it determines the scallop height on the final shape. This is often the preferred way to cut pocket-like sculptured shapes, though only a few systems in the literature use it. (The papers in Section A.6 of Appendix A include some references on this technique.)

Because the Z depth is controlled by the surface, the surface representation scheme tends to determine (at least partially) the algorithms used and the style of paths that result. Hence, many issues are algorithm or representation dependent. Some systems, for example, consider only one patch at at time, even though they may develop a single path to machine a collection of surfaces. Such systems may not consider an adjacent surface when moving the cutter near the common boundary with that surface, leading to special treatment being required near boundaries with other surfaces, as shown in Figure 2.11. Similarly, the system might develop sub-paths for each surface and simply link them together, rather than running the cutter over the whole collection at once. However, linking the paths in an efficient manner can be non-trivial, depending on the topological arrangement of the patches and where each path begins and ends [CC89b].

2.2.4 Object Representation

The type of object representation determines what geometric information is readily available and what must be computed. It also determines what information can *realistically* be computed. For example, some representations support estimation of tool load by giving an estimate of the width and depth of cut for a given tool move. Some representations support the generation of partial or *cleanup* paths that send the cutter into only the areas of the model containing unremoved material. The choice of object representation therefore determines the types of path generation algorithms that are possible.

There are four general types of object representation:

- point-based
- curve-based
- surface-based
- solid-based

Point-based representations include both pixel-based representations and formulations based on non-regular point sampling. Curve-based representations are 2D planar profile geometry for pocketing and roughing. Surface-based representations include common surface schemes such as tensor-product B-spline surfaces, trimmed NURBS surfaces as well as surface approximations such as piecewise linear (triangular mesh) approximations. Solid representations include CSG and B-Rep solids, octree models and even tensor-product solids.

An issue with surface models is whether or not the path generation system requires topological information for a collection of patches (for reasons mentioned above). Also, some surface-based path generation methods are applicable only to four-sided, tensor-product patches. While not as common, triangular surface patches and other alternative formulations exist. For triangular meshes and other surface approximations, there is an issue of approximation tolerance and how much of the allowable final form deviation it consumes.

2.2.5 Tool Positioning

Tool positioning is a component of all tool path generation programs, though trivial for 2D pocketing and profiling. Tool positioning is used for surface machining paths, point-wise roughing paths, and profile paths based on curves that follow a sculptured shape. When a likely path position has been found, the tool positioning strategy is used to find the location of the tool tip (and hence the *path point*). The method used depends on the object representation and the tool geometry. Generally, there are four positioning methods:

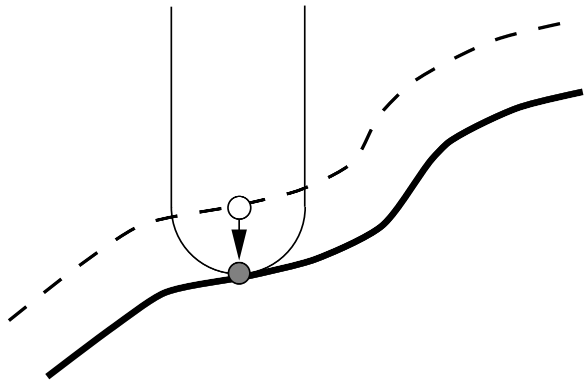


Figure 2.12: Tool positioning based on uniform offset surfaces. The offset surface passes through the ball centre and the tool tip position is found by a simple translation.

- offset surface approach
- cutter contact method
- projection or highest point approach
- APT method

The offset surface approach is shown in Figure 2.12. It involves finding the uniform offset surfaces of the given model, giving the locus of ball centres for a ball-endmill. Paths are planned using the offset surfaces and points are translated down to get tool tip locations.[10] Issues with this approach include detecting and removing loops in offset surfaces and filling of gaps that may appear in the offset patch collection. The use of uniform surface offsets restricts the method to ball endmills (a fairly serious restriction). Note that this method can be applied to triangular meshes and pixel models as well [Hwa92, KKS87].

[10] Actual methods vary slightly. For example, the whole offset surface collection could be translated down instead.

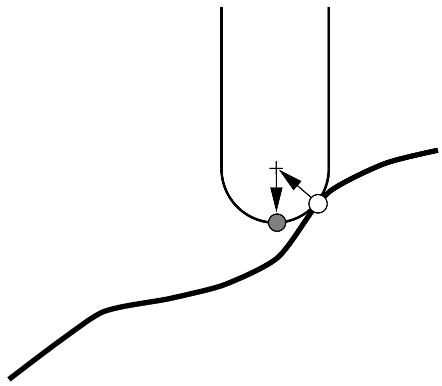


Figure 2.13: Tool positioning based on the cutter contact point. The tool tip position is found relative to the contact point in the surface, which is known. This method is not limited to ball endmills, but is more involved for other tool shapes.

The cutter contact method finds the tool tip location by using the surface normal vector, and tool geometry to compute the tool tip position relative to a given point on the surface. It is shown in Figure 2.13. Since it is a local method, problems of *gouging* can occur where the curvature of the surface exceeds that of the tool near the tool tip position. Another form of gouging occurs when the tool is cutting near the edge of the given surface and gouges an adjacent unconsidered surface as shown in Figure 2.11. Some systems consider gouging only in the cutting direction by checking the curves used to generate the path [HNF$^+$86]. Other systems generate potentially incorrect points then detect and correct gouge points in a second pass [CJ89].

Figure 2.14 shows the projection or highest point method of tool positioning. The tool is conceptually dropped onto the surface; the method finds the Z position of the tool tip such that the tool just touches the surface and no surface points are inside the tool tip. It thus finds interference-free positions. It is used with

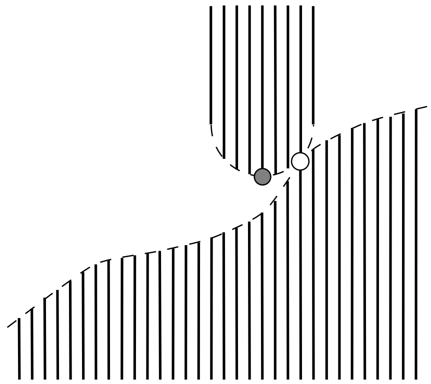


Figure 2.14: The highest point tool positioning method. The vertical lines represent pixels. The method can also be used with triangular mesh models.

point/pixel-based systems and some mesh-based surface representations.

The APT-based approach, shown is Figure 2.15, is a method from the APT language[11] whereby the tool is positioned to three surfaces simultaneously using an iterative, Newton-based scheme. The three surfaces are called the *part surface, drive surface* and *check surface.* The part surface is the surface of the object; the tool is positioned tangent to it and the drive surface simultaneously. The check surface is a form of edge handling. It represents a surface the tool must not go past as it moves along the part surface–drive surface intersection. A problem with this approach is that the iteration is dependent on a good initial estimate and reasonable geometry; this is not always the case in practice. Hence, the other methods have become more popular in recent times.

[11]A very old NC programming language (roughly contemporary with the first FORTRAN compiler).

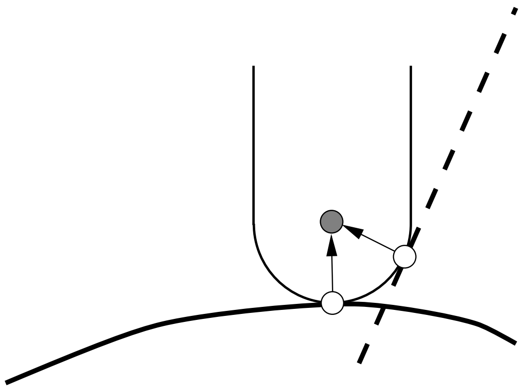


Figure 2.15: The tool positioning method used by the APT language. The heavy curve represents the part surface; the dashed line is the drive surface. The method positions the cutter tangent to both simultaneously. A third element, the check surface, limits movement perpendicular to the direction of cut.

2.2.6 Path Planning/Path Topology

Given that the type of machine tool and tool path has been specified, and that the tool geometry, tool positioning algorithm and representation type to be used are known, the next task is to find locations for the tool spread across the object in a pattern consistent with the type of path being constructed.

Path Topology

The term *path topology* is a vague term that generally refers to the pattern of the tool path in a broad sense. Two common topologies are zigzag and contour-parallel paths, but others are possible too. Paths with a space-filling curve topology are created by using nodes on the curve to plan points on the path. Even a straightforward zigzag path can be generalized such that the topology of the path is difficult to characterize precisely.

One generalization is to allow the passes of the tool on the surface to have a uniform scallop height rather than a uniform distance. Doing so means the passes become more like contours on a contour map. Linking them may be non-trivial, depending on the distribution of the passes. A generalization of iso-parametric zigzag paths is to remove portions of some passes in regions where the shape and parametrization of the surface cause the passes to be too close together [EE94]. This gives a collection of subpaths with the proper density, but linking them together in an efficient and acceptable manner may be difficult. Characterizing the topology of the final path may also be a challenge.

In this thesis, the term path topology means simply the path pattern or style. I do not attempt to define it further.

Path Planning

Given that the style of path is known, tool positions, passes and connected passes must be constructed so the tool visits the specified regions, cutting within tolerance, using the specified topology. This is the task of path planning.

Path planning involves many issues discussed above. In addition, it must also consider, simultaneously, three levels of detail: points, curves, surfaces (or areas, or regions). In the case of level-by-level roughing, another dimension appears in the form of the current cutting level.

The algorithms used depends on the type of path but also on the object representation. The above example of truncated iso-parametric zigzag paths has no counterpart in a pixel-based representation. (In many ways, parametric surfaces are a special case. The existence of the parametric domain offers the choice of plan-

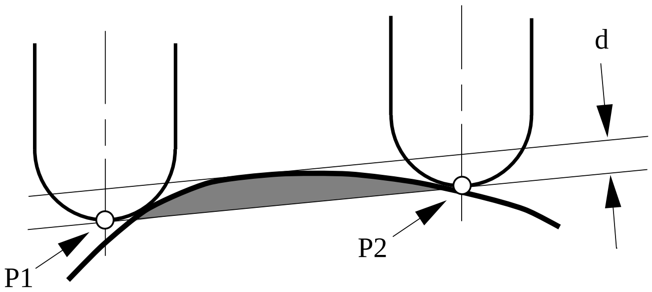


Figure 2.16: The step along computation for surface machining. The problem is to find *P2* relative to *P1* such that the amount removed, d, is within tolerance. The shaded area represents the material removed the tool moves from *P1* to *P2*.

ning the path in either parametric or Cartesian space. It is possible, for example, to apply space-filling curves to path planning in either domain.)

Path planning can be broken into three general areas:

- point planning and connectivity
- pass planning and connectivity
- region planning and connectivity

The areas may be viewed from either a bottom-up or top-down perspective. For example, a cutter will be in a certain region, moving on some specific pass and the next point will be found. Or, the tool will be at a point on a current pass in a current region.

Point Planning and Point Connectivity

Point planning deals with finding the next point at which to invoke the tool positioning algorithm. This is usually done in terms of a parameter called the *step* *along* or *step-forward* distance, but it also involves the type of path being planned and the topology being used. The step along computation finds the next point for tool positioning based on the current location of the tool, the shape of the surface, the tool geometry and the acceptable shape deviation tolerance. Recall that the movement from location to location is most likely to be in the form of a line segment; in the case of a locally curving sculptured shape, the cutter may cut into the surface by some amount. The step along calculation, as illustrated in Figure 2.16, attempts to limit this to an acceptable amount.

Point connectivity planning is not a computation like the step along computation. Typically, the previous tool position will be connected to the next. However, in the case of a point-wise roughing path, there is *no* connectivity between points. (The cutter operates at each site independently hence ordering of the points with respect to each other can be somewhat arbitrary.) In the case of 5-axis paths, the tool orientation must not change drastically between two connected path points [LJ94].

In the case where the tool positioning algorithm is unable to obtain a suitable location (e.g. raises the tool to avoid interference, or positions the tool on the edge of a surface), the connection between the last point and the present one may drag the cutter through an unacceptably large lump of material. This situation must be detected and handled (even if only by adding a point to lift the cutter up and over the problem area).

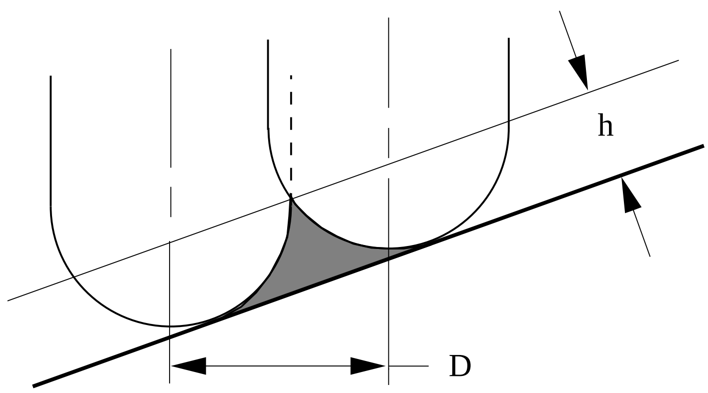


Figure 2.17: The stepover computation for surface machining: find D to limit ridge height h to an acceptable amount.

Pass Planning and Pass Connectivity

Pass planning involves computing the information required to guide the cutter as the next pass is constructed. Typically, this involves finding an initial pass then finding the distance between successive passes of the cutter. It also involves finding the starting point and recognizing the end of the pass.

A common method of path planning is to move the tool along iso-parametric curves on a parametric surface. Another way is to construct curves in the XY plane and use these to guide the cutting direction for the passes of the cutter. Yet another way is to slice the surface with a plane and use the intersection curve as the basis for the current tool pass.

The distance between passes, shown in Figure 2.17 is called the *stepover* distance. This is similar to the step along computation in that the surface shape, tool geometry, and tolerance are used to find a distance to move the tool over for the next pass such that the height of the ridge of material left uncut between the passes is acceptable.

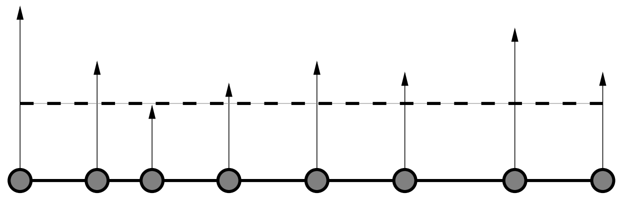


Figure 2.18: Planning the next pass in surface machining. The stepover value is found at all points on the current pass as it is being constructed and the minimum value is used.

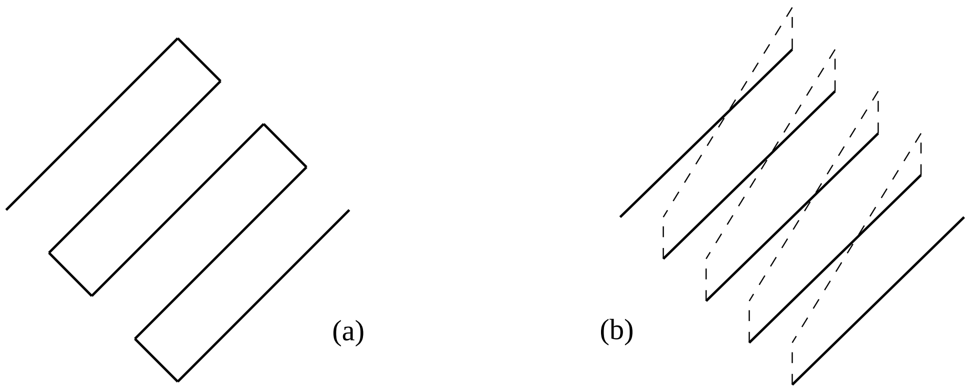


Figure 2.19: Two types of pass connectivity. In (a), the common zigzag approach is used. In (b), passes have been connected for one-way machining. The dashed lines indicate non-cutting movement.

As mentioned above, when machining a sculptured shape, the stepover required for the next pass (to maintain a specified scallop height relative to the current pass) may vary considerably. A common method is shown in Figure 2.18. The stepover is calculated at each point on the current pass and the minimum is selected as the actual stepover.

Once passes of the cutter have been constructed, there is the issue of how to connect them together. This depends on the type of path and the type of machining specified. A common way of linking passes that are roughly parallel is to move from the end of the last pass to the closest endpoint of the next. This gives the common, zigzag paths seen previously. In some cases, it may be desirable to have all passes move the cutter in the same direction. In such cases, the tool must be lifted off the work and connected to the appropriate starting point of the pass. This style, shown in Figure 2.19, is called *one-way* machining.[12] Note that point-oriented paths have no notion of a pass, hence there is no pass connectivity. (The passes can be thought of as having degenerated to a single point.)

Region Planning and Region Connectivity

The passes of the cutter need to be planned such that they move the cutter through all the areas of the object that need to be machined by the cutter. Hence, the object may be decomposed into separate regions, each of which is machined separately. The sorts of regions that are possible depend on the object representation type and the purpose of the path. For example, the geometry for a pocketing path may be broken up into simple convex polygonal regions. Similarly, a CSG solid model might be machined on a primitive by primitive basis. A surface model might be machined patch by patch, especially if an iso-parametric-based algorithm is being used.

In some cases, it is undesirable to have the geometry of the model affect the topology or region connectivity of the path. For example, the user may want the tool to sweep back and forth over the entire model (as much as possible), rather than machine in a patch-by-patch manner. In that case, the entire object is treated as one conceptual region. However, if the 2D profile of the periphery of the object is a

[12]The direction of tool motion is an important factor in the physics of machining; the direction can have a serious impact on surface finish for some materials.

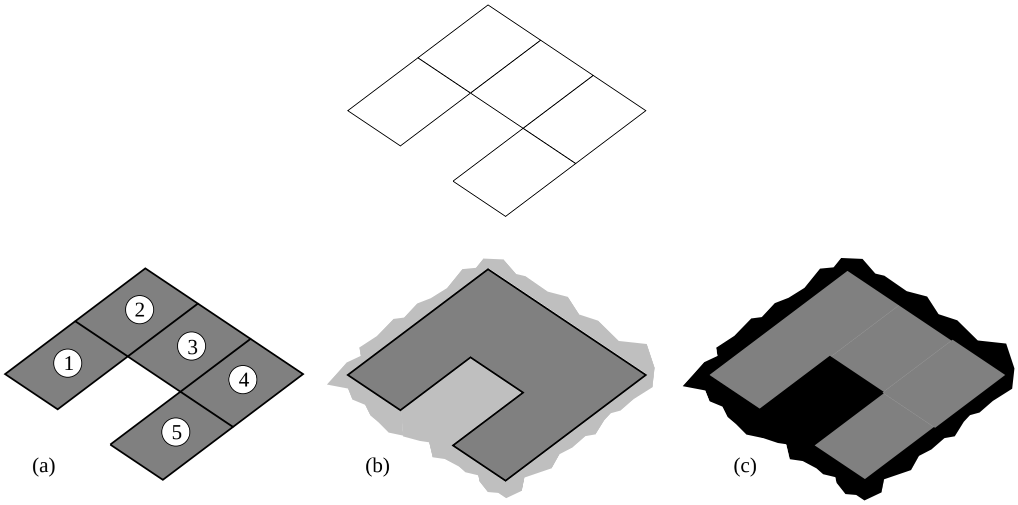


Figure 2.20: Three interpretations of region connectivity for a set of five surface patches (top). The patches can be processed by: a) Creating paths for each patch in isolation then merging them; b) Merging the five into one area and allowing the cutter to move freely between patches; c) Merging the five, but restricting the tool to stay within the bounds of the surfaces, like a pocket. Other combinations are possible as well.

non-convex polygon (see below), then the tool may still have to avoid interruptions due to the shape of the boundary. In the case of parametric surface patches and an iso-parametric algorithm, the algorithm naturally cuts patch-by-patch. While it is possible to merge at least some of the parametric domains, in general this would require inverting a high degree polynomial which is too expensive to use in practice.

In general, the region in which the cutter must remove material can be considered an *active region*[13] – a region where the tool must cut. A region where the tool must stay out of is a *forbidden region* (or an island, in pocketing). A region where the tool is free to move about is a *free zone* or *unrestricted region.* Generally, a tool

[13]The terminology of this section is my own but the ideas are taken from various places in the literature.

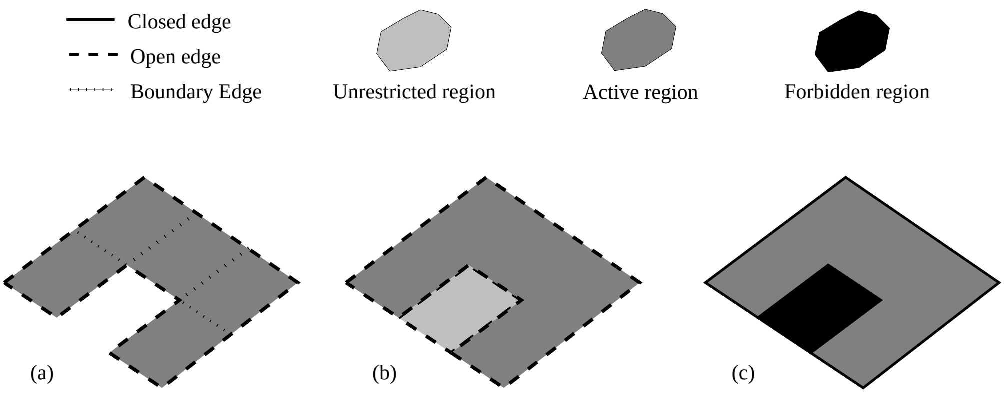


Figure 2.21: Regional decomposition. The geometry of the previous figure has been converted to regions and edges. The forbidden region in (c) is not strictly necessary; a closed boundary would have been sufficient.

will only pass through a free zone on its way to another region or while cutting the edge of an adjacent active region. Figure 2.20 shows how five surface patches can be interpreted in different ways.

Related to the concept of regions is the boundary between them. A forbidden region implies a closed boundary, i.e. one where the tool must cut tangent to the edge of the forbidden region. An unrestricted zone implies an open edge with an active region since there is no potential for interference if the cutter cuts on the boundary of the active region. The boundary between two surface patches (assuming a surface-based object representation) may give rise to a boundary edge in the case where the path planning algorithm needs to explicitly handle such features (as in Figure 2.11). Figure 2.21 redisplays the interpretations of Figure 2.20 using the nomenclature for regions and edges. Figure 2.22 shows how a mix of open and closed edges gives rise to a shape that is neither a pocket nor an open collection of surface patches.

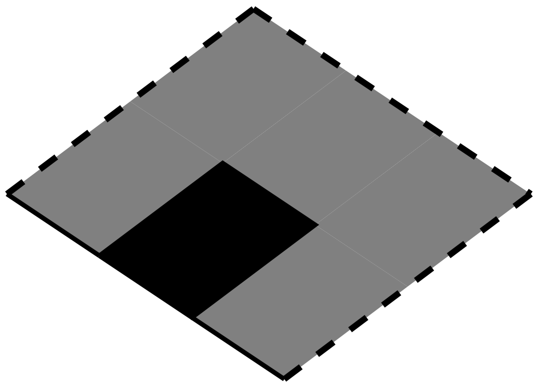


Figure 2.22: A different interpretation for the patches in Figure 2.20. The cutter is allowed to move on three open edges, but must cut tangent to the forbidden region and two closed edges.

Still other sorts of regions exist as well. In pocketing using Voronoi diagrams, the diagram gives a partitioning into regions of the given geometry. In pixel-based systems, it is possible to compute curvature maps of the surface and define a collection of regions based on curvature and shape [Elb92]. Different tool positioning algorithms can be applied according to the type of region a tool position falls in.

In some cases *region* connectivity is absent. An example is a zigzag path that lifts the cutter over obstacles in a manner similar to that in Figure 2.9 (a). In that case, the adjacent region is visited on each pass and does not appear in the final path (except for the extra tool motion it causes). If regions are not handled at the pass level, there is an issue of how to connect them. Again referring to Figure 2.9, the extra regions may be placed on a stack and processed recursively. Another scheme, common to pocketing, is to process uncut regions that are accessible from an already cut area. This is done in order to minimize the need for drilled holes that may be required to assist the cutter in entering a closed, uncut region.

Other Path Planning Issues

If the path is a multi-level pocketing or roughing path, there is an issue of *level connectivity* as mentioned before. Different types of cutting strategies are possible based on the type of tool. A related issue is that different patterns can be used for each level. More generally, it is possible to use different strategies for each *region*, though doing so could cause problems later for the path linking step.

A final aspect of path planning is whether or not the representation stores the shape of the model after the previous path has been applied to it. If it does, then localized, *cleanup cuts* may be possible for subsequent tools. Briefly, these are paths that only send the cutter into areas the previous cutter was unable to access. Thus, the tool does not traverse the entire model, but only visits those regions where excess is present. Of course, connecting the disjoint collection of cleanup paths is also an issue.

Chapter 3

Hilbert Curves and Pixel Models

This chapter examines the use of Hilbert curves and pixel models in tool path generation. Section 1 looks at pixel-based systems and the operations they support and examines their strengths and weaknesses in tool path applications. Section 2 gives a similar treatment of a system based on Hilbert curves. A hybrid system that uses elements of both is the subject of the next chapter.

3.1 Pixel Models

A pixel model is a very simple data structure, a 2D array of height values. Creating a pixel model is conceptually simple and straightforward to implement. A uniform grid is positioned above the object, and parallel rays are cast one per grid cell toward the object. The distance along the ray from the grid to the object is recorded in the array representing the grid.

The centre of the grid cell defines the x and y coordinates of the intersection point, and the depth represents the z value along the negative Z axis. The object is thus represented by a set of uniform point samples. Of course, the pixel model can be interpreted in other ways. The cells can be considered small parallel squares in space. The size of each square is the distance between grid centres. The middle of the square is set on the ray intersection points. Another interpretation is to consider the intersection points as the corner points of small bilinear surface patches. A volume interpretation is also possible using either of these schemes, though a second height value is required to define the volume. Finally, the pixel model can be considered to be a grid of scalar values, or *image data.*

In practice, a different method is used to create pixel models. The object is approximated by a triangular mesh and the nodes of the mesh are projected onto the same conceptual grid as before. Height values for the centres of grid cells falling inside the triangle are found by linear interpolation of the corner nodes. Repeating the process for every triangle in the tessellation of the object gives a pixel model representation of the shape. This process is fundamental to computer graphics and is implemented in hardware on high performance graphics workstations.

Since the triangles are planar entities, the linear interpolation introduces no error in moving from triangular mesh to pixel model. However, the conversion from an object representation type such as non-uniform rational B-spline (NURB) surfaces to triangular mesh involves approximating the surfaces, hence, there is some inaccuracy in converting an object to a pixel model.

A different type of inaccuracy is inherent in the pixel representation itself. Since the height of the object over the cell is represented by the height at the middle of

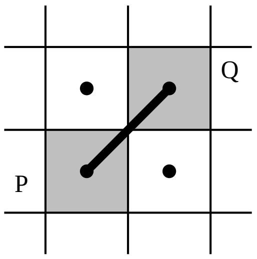


Figure 3.1: Neighbouring pixels in Theorem 1.

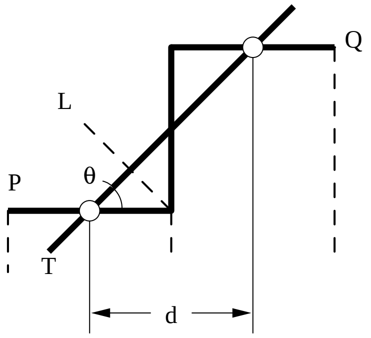


Figure 3.2: Geometry for the proof of Theorem 1.

the cell, small spike-like features occurring between cell midpoints may not be represented at all. Fortunately, as the size of the cells decreases, the amount the object can deviate from the midpoint of the pixel decreases. In fact, the following result holds for pixel models:

Theorem 1 *Let d be the diagonal width of a pixel as well as the diagonal distance between the centres of two adjacent pixels P and Q, as shown in Figure 3.1. Let the object be represented by a triangular mesh and let T be a triangle in the mesh that passes through the midpoint of P and Q. Then the error between T and pixels P and Q is at most $\frac{d}{2}$.*

Proof: Consider the pixels to be solid square columns of varying heights. In two dimensions, the pixels appear as shown in Figure 3.2; T passes through the midpoint of the top of P and Q and makes an angle θ with the top of P. Form a line L, as shown, having an angle of 45° with the top of P. Points on T above P and above L are closer to the vertical face of pixel Q while points on T above P below L are closer to the top face of P. In either case D, the maximum value of the minimum distance, is:

$$D = \frac{d\,sin(\theta)}{2}$$

Taking the limit as θ goes to infinity:

$$D = \frac{d}{2} \tag{3.1}$$

□

Pixel representations have an advantage over many other data structures in that the regularity of the structure often leads to simple but effective algorithms. Also, the same regularity permits parallel processing, though the operation and algorithm determine if parallel processing is possible for a given application. Another advantage is that pixel models can be used as a common representation for other representation schemes. Many different shapes can be modelled using a wide variety of mathematical formulations then converted to pixel form and operated on using a common algorithm. A final, related benefit is that high performance graphics hardware and image processing libraries can be used to assist in creating and manipulating the pixel models.

By far, the biggest disadvantage of the pixel representation is the storage requirement for physically large objects rendered at high resolution. A simple example is the following. An object 300 mm by 300 mm represented by a pixel model with pixel size 0.01 mm (100 pixels per millimeter) and using 24 bits (3 bytes) to store the depth values would require 2.7 billion bytes (2.7 Gbytes) of storage. Larger objects and/or finer tolerance increase the storage demands quadratically. While gigabyte storage units are now commonplace, the processing time also increases with larger grids. Hence, model size remains problematic.

Another undesirable property of pixel models is that edges and other curve-like features are represented only indirectly. Detecting and processing features (such as the intersection between a hemisphere and a plane) is costly and non-trivial.

The limitation of storing only one height value per pixel is not a disadvantage for applications in 3-axis milling path generation. The fixed tool direction of 3-axis machines means that the tool can only access the topmost parts of the object;

undercuts and similar features cannot be cut. Thus, the pixel model representation matches exactly the class of surfaces the machine tool is able to produce.

3.1.1 Pixel Models in Tool Path Generation

The fundamental approach to using pixel models for path generation is to create pixel representations of both the object and the tool, and then create the path by using only pixel operations. The most important elements of path generation with pixel models are the following: tool positioning, path planning, and modelling of advanced or unusual features.

Tool Positioning

For tool positioning, there are three main approaches used:

- highest point method
- offset surface method
- cutter contact point method

The above methods are very similar to what was discussed in Chapter 2.

The pixel equivalent of the highest point method is given in a paper by Saito and Takahashi [ST91]. The idea is to model the tool tip using vertical height values for each pixel relative to the height of the bottom centre of the tool tip. The tool model is positioned such that the centre of the model coincides with a pixel of interest in the object model. For all pixels on the object that coincide with pixels for the tool, the difference of the object height and the tool height is taken. The maximum of all such values gives the height for the tool tip at that location. The x, y coordinate of the pixel plus the height gives the point representing the tool tip location. Mathematically, if $Z(x, y)$ is the grid of object heights and $H(i, j)$ is the grid of tool tip heights, then the tool tip height z is given by:

$$z = \max_{ij} \left(Z(x+i, y+j) - H(i, j)\right), \quad where: \; i^2 + j^2 < r^2. \tag{3.2}$$

In their paper, Saito and Takahashi consider the two pixel models as image data and construct an entire grid of tool tip positions.

The pixel-based offset surface approach is given in a paper by Kishinami, Kondo, and Saito [KKS87]. They call their method the *inverse offset method* because they find the discrete approximation of the offset surface by finding maximum height values for all pixels in the object grid by positioning an inverted tool image at every pixel. This approach is more general than the typical uniform offset surface method because any tool shapes can be represented in pixel terms and the general tool offset surface found. Saito and Takahashi note that their method is equivalent to the inverse offset method.

A technical report by Thomas [Tho90] uses the third approach. He uses the object surface normal vector (stored for every pixel) and the object height at a pixel along with the tool ball radius to find the tool tip position in 3D Cartesian space. To check for and eliminate gouging, he checks tool heights relative to the object using computed tool tip positions. Since the location is not likely to be exactly on the centre of a pixel, he uses bilinear interpolation to find the tool height at the coordinates of the object pixel centre. If the tool is below the surface at this location, the difference is saved as an increment required to raise the tool. The maximum of all such values is the amount that eliminates all gouging.

Support for Advanced Features

A major advantage of using the pixel model representation comes from the many aspects of path generation it supports. For example, Saito and Takahashi create paths for a given size of cutter based on the shape of the object after the previous cutter in the sequence has been applied. Similarly, they define an image of height data that represents the maximum depth of cut that can be taken by the tool as defined by the length of the cutting edges along its vertical axis. If the raw billet of material (or intermediate shape) implies a deeper than feasible cut, the maximum safe depth is used. Similarly, the tool holder and spindle can be modelled and checked for interference. (In fact, several pixel model implementations [CCPK94, ST91, TSAO89, Tho90] feature this check.)

As mentioned in Chapter 2, tool load is an important issue and the volume removed by a given tool movement can be approximated using a pixel model [ST91]. To begin with, a model is required for the state of the object before the path is applied. The line joining two path points is broken into pixel-sized steps and the tool is positioned at each pixel; the (pixel) Boolean intersection between the tool and the object at a given location gives the static volume removed by positioning the tool at that location. Total volume for the move is the sum across all pixel locations. The operation is sequential as the nth position of the tool must not include material removed by the $n-1$st position. Based on the cutting volume, tool geometry, material and other properties, the force on the tool can be estimated and suitable feed rates selected.

Another important property of pixel models is their ability to support the creation of cleanup cuts (as described in Chapter 2). In pixel terms, the difference in shape between a partially finished object and the desired shape gives an indication of where excess material remains. Instead of creating paths that traverse the entire model, subpaths that remove only the remaining excess can be created using suitably sized cutters. The process is repeated until only a small amount of excess remains. Both Thomas [Tho90] and Saito and Takahashi [ST91] give algorithms for implementing cleanup cuts.

Path Planning and Path Topologies

In terms of path construction, the easiest style to implement is a zigzag topology, based on scanning of rows or columns in the grid. A side-benefit of the pixel representation is that special seam handling is not necessary; any of the above tool positioning techniques will correctly position the tool along the boundary between two surfaces.

Several other styles are mentioned in the literature, though none is described in detail. Saito and Takahashi suggest iso-parametric paths built by utilizing another grid containing the parametric u and v coordinates for each pixel. Approximately constant values could be traced and used to guide path point selection. In the same way, they suggest tracing constant Z values in the height data as a way to produce contour paths. While the method is feasible, the authors give no idea how much computational effort is involved. A paper by Suzuki, Kuroda and Sakamoto [SKS+91] gives a similar method based on a bilinear patch interpretation of the pixel model. Takeuchi et al. [TSAO89] describe level-by-level and drilling (point-based) roughing paths and zigzag finishing paths in their implementation. A paper by Choi et al. [CCPK94] describes similar path types plus something they call *pencil cutting*, which they do not define.

Advantages and Disadvantages

The pixel model approach has several benefits, many of which are inherited from the pixel representation: simple algorithms[1] (in most cases), potential for parallel processing, and uniform, common basis for shape representation. In terms of path generation, the method supports: general tool shapes (though drafted cutters are not well represented), cleanup cuts, tool load estimation, and general tool and holder collision checks. The three tool positioning algorithms mentioned previously all give interference-free tool positions and with the pixel representation explicit seam handling between pairs of adjoining surfaces is not required.

Of course using pixel models in path generation also inherits the storage and processing concerns of the pixel representation. In general, storage of order $O(xy)$ is required, where x and y are the number of pixels in the X and Y directions, respectively. Processing time is order $O(xyr^2)$, where r^2 is the number of pixels in the cutter representation [ST91]. A similar concern for path generation is wasted computation when the offset surface approach is used. The whole offset surface is computed, but only a subset of the points is used to form the path. Picking the points to be used is an issue, too, as there are many to choose from and some form

[1] A paper by Vepsäläinen [Vep90] shows how paths can be defined strictly in terms of the *erosion* operator from the field of image morphology.

of filtering is required.

Another disadvantage of path creation in pixel space is that graphics hardware can be used to create the pixel models initially, but is of little benefit in performing pixel-level operations like tool positioning. This is because the graphics hardware is designed to rapidly convert triangles to pixels; pixel to pixel operations are not accelerated.

Another pixel model weakness that interferes with path generation is the indirect representation of curves and edges. Because of this, rather complicated algorithms are required to trace along edges, making boundary following passes (as discussed in Figure 2.5) expensive. Further and more serious is the fact that region boundaries are also difficult to define and use. In pocketing, for example, an outer boundary is offset inwards by the tool radius in order to determine the area where the tool centre may roam without colliding with the edges. In some cases, the offset may be null, implying the cutter cannot fit in that region. In pixel terms, the only way this can be determined is by trying to position the tool in the general area and finding out the tool falls in a forbidden region for all points in the region. Again, this leads to wasted computation. Thus, the low-level pixel representation is not sufficient to support efficient path generation where region inclusion tests or edge following are required.

3.2 Space-filling Curves

Hilbert space-filling curves are introduced in this section, followed a description of a path generation method based on Hilbert curves given in a paper by Griffiths. In addition to being one of the two approaches in the literature that use space-filling curves, it also forms the basis for the hybrid scheme described in the next section.

3.2.1 Hilbert Curves

The curves known as *Hilbert curves* are a set of continuous, non-differentiable curves defined on the unit square. Hilbert curves are a class of *space-filling* curves. A space-filling curve has the remarkable property that in the limit it passes through every point in the two-dimensional unit square, even though it is a one dimensional entity [Sag94].

Hilbert curves are uniquely determined by a parameter known as the *order* of the curve. A curve of order N is recursively defined in terms of a curve of order $N-1$. It is constructed by concatenating four affinely transformed copies of an order $N-1$ curve. Figure 3.3 shows Hilbert curves of order zero through five.[2] Note how a curve of order three contains four copies of a curve of order two, a curve of order two contains four copies of a curve of order one and so on.

The nodes of Hilbert curves define a uniform sampling of the unit square. Figure 3.4 shows the sampling implied by a Hilbert curve of order two. The sampling density doubles as the order of the curve increases by one. Since the curve fills the unit square in the limit, arbitrarily dense samples of the square can be obtained using the nodes of a Hilbert curve of suitable order.

Another important property of Hilbert curves, extremely useful for tool path

[2]Hilbert curves of order zero are usually not defined; however, it is a useful interpretation since a Hilbert curve of order 1 becomes four connected, affinely transformed copies of a curve of order zero. Hence the order one case is defined in the same way as higher order curves and a point becomes the base case.

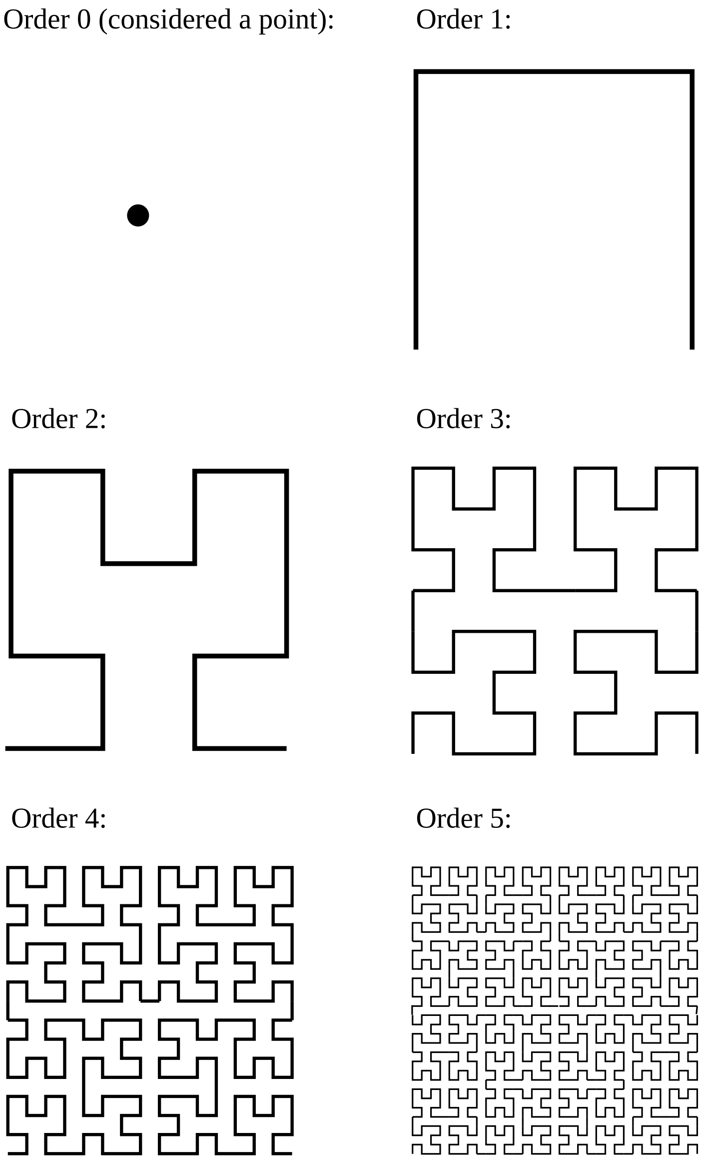


Figure 3.3: Hilbert space-filling curves of order zero through five.

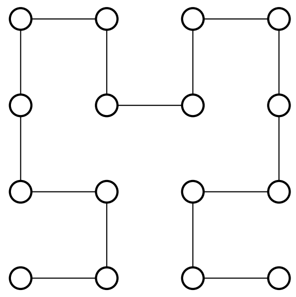


Figure 3.4: Point samples implied by a Hilbert curve of order two.

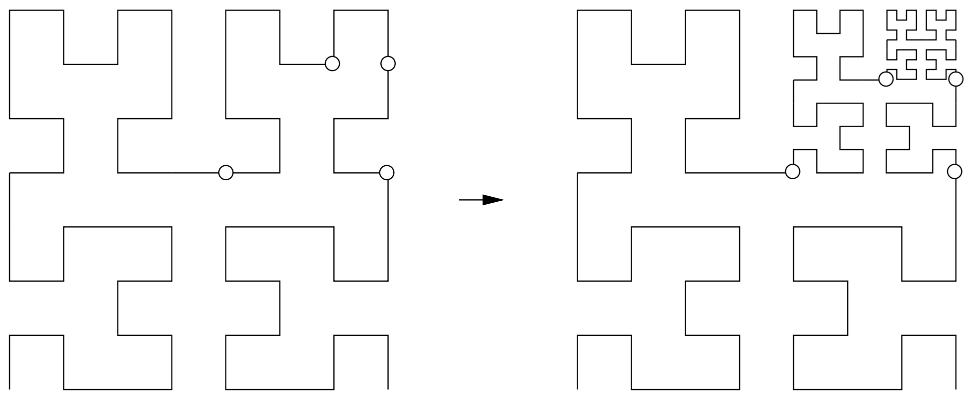


Figure 3.5: Local refinement of a portion of a Hilbert curve. The upper right portion of the curve has been recursively refined twice.

applications, is *local refinement*. It is easy to replace a portion of a curve with an appropriately transformed curve of higher order. Figure 3.5 illustrates the idea. Replacing sections of the curve of order N with sections of order $N+1$ gives an effective way to refine the sample density in selected areas of the unit square.

3.2.2 Hilbert Curve Tool Paths

A paper by Griffiths [Gri94] describes a method for creating tool paths for ball endmills using Hilbert curves, offset surfaces and ray tracing. The method supports both roughing and finishing path generation; in fact it supports the generation of a *sequence* of roughing paths, plus a final finishing path.

Griffiths' scheme uses the uniform offset surface method to provide tool positioning. It is therefore limited to ball endmills. To deal with possible self-intersections of offset surfaces, a ray is cast into the object from above, parallel to the tool axis. If multiple intersections are obtained (due to surface self-intersection or overlap of adjacent offset surfaces), the highest value is used.

To plan the tool path and to reduce computational overhead, a bounding rectangle known as the *work area* is introduced. The initial work area bounds the collection of patches that define the object to be machined and also serves as the domain for the Hilbert curve that forms the tool path. The work area is first subdivided into equally sized smaller work areas each no wider than the diameter of the tool. The ordering of the subdivision is such that visiting the midpoints of these areas traces out a Hilbert curve in the plane. For each smaller work area, a list is kept of the offset surface patches that are within its boundaries in the plane. Thus, work areas are used as a bucketing scheme to reduce unnecessary intersection tests.

At the middle of each smaller area, rays are cast into the object to determine a height value to be associated with that area. The collection of work areas and object height values along with the implied Hilbert ordering gives a coarse surface machining path. To create a path that machines the object to the required accuracy, work areas are further subdivided in a manner that retains the Hilbert ordering on the entire collection of areas.

The method used to determine the accuracy of the path is unclear. Accuracy is based on whether or not the portion of the object that falls within the work area can be cut "sufficiently accurately by centring the cutter on the highest intersection point on the ray". If this is the case, the point defined by the work area midpoint and highest ray intersection is added to the tool path. If not, the work area is recursively subdivided. Exactly how this assessment is made is not stated. Griffiths says only that "[t]he depth of recursion depends upon the local complexity of the surface, the radius of the cutter, and the accuracy required."

The Hilbert ordering of the path points gives a path where the tool is constantly changing directions. This is undesirable from a kinematic point of view as the machine tool must repeatedly slow down, stop, and change direction. Griffiths uses an optimization based on a second pass through the points that reorders sections of the path to a zigzag traversal. The collection of path points is recursively bisected and sections wherein all points were created at the same level of subdivision are reordered, but retain the same start and end points. Thus, between groups of points, the same connectivity is retained. Figure 3.6 illustrates the idea.

It turns out that the bisection visits the points in the same Hilbert ordering that was used to generate the work areas from which the points were created. Thus, isolated points not reordered (from deep subdivision levels) are still connected to neighbouring sections through the original Hilbert ordering. More importantly, the groups of reordered points are also connected to each other in Hilbert order. If

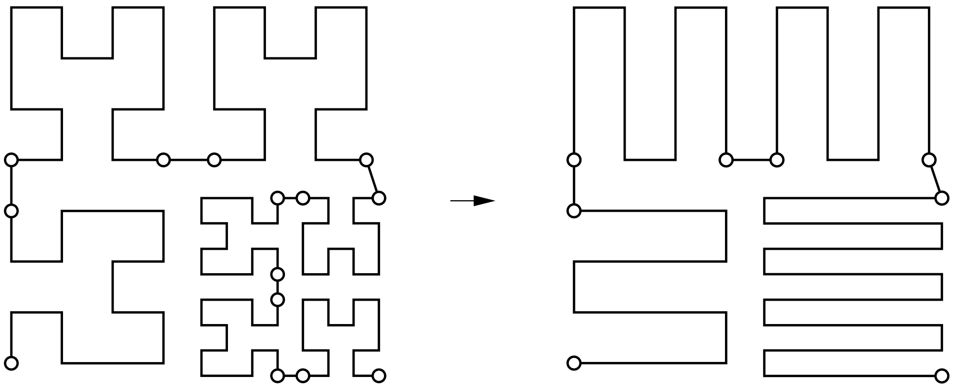


Figure 3.6: Reordering of points in sections of a Hilbert tool path. The Hilbert connectivity is retained between reordered regions.

a large number of points in one section are reordered, the path loses much of its Hilbert character. If many small groups are reordered, the result is less effective.

To create roughing paths, the initial subdivision of work areas is used, along with their associated height values. A sequence of cutting depths is specified and paths are generated for each layer; the result is a sequence of planar roughing paths. The middle of a work area defines the x, y coordinates of a path point; the z values are determined with a test. For each work area, the object height will either be less-than or greater-than the current cutting level (equality is treated the same as less-than). When the height is less-than the current level, the current level is used as the height value. If the height is above, the last cutting level greater than the work area height is used.

Two second pass optimizations are used for roughing paths. The first is a zigzag simplification, similar to what was outlined above. The list of path points is revisited and recursively bisected. If all points in a section are at the current cutting level, the section is reordered. Since all points are on the same plane, each

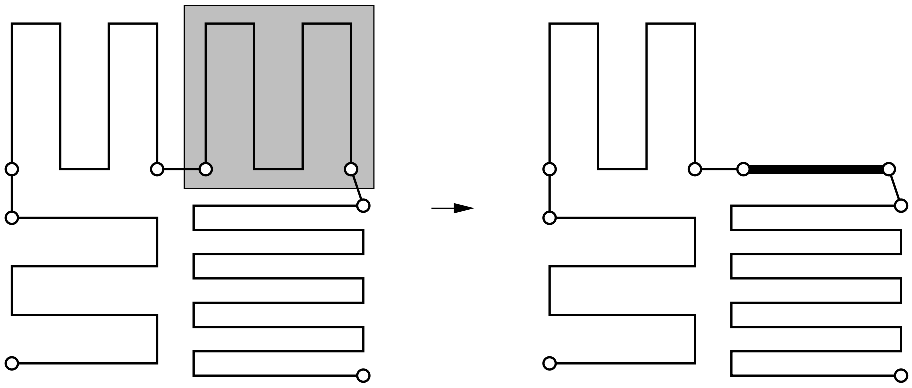


Figure 3.7: Skipping areas in a roughing path. All points in the indicated area are above the current cutting level; they are replaced by a skipping movement. Again, the Hilbert connectivity is retained between skipped regions.

zigzag pass is defined only by a start and end point, greatly reducing the number of points in the path.

The second optimization involves removing redundant points. For a section of points lying entirely above the current cutting level, the points are discarded, replaced by a line joining the first and last points of the section. See Figure 3.7. This is acceptable, since the discarded points are not used to machine at that level. As in zigzag simplification, the Hilbert connectivity between skipped sections is retained. Thus, skipping moves are compatible with the rest of the path. To eliminate tool-object collisions, the Z value used is the maximum of all Z values for the points in the section.

Advantages and Disadvantages

The major advantage of Griffiths' method is the adaptive refinement property: effort is expended only where necessary. This means the path planning algorithm can add more points in areas requiring them. The position of the points is not constrained by the topology of the path (in a sense, the topology comes after the points). Another advantage related to refinement is that all paths are developed based on the initial work area decomposition for the largest tool. Paths for smaller roughing and finishing tools are all further refinements of the coarsest path. This property allows implementations to use the results of the path for the previous tool in the sequence in the generation of the next tool path. Not only does this save computational effort, but it also fits well with the general requirement for a sequence of roughing paths and a sequence of finishing paths, as outlined in Chapter 2.

A computational benefit inherited from the local refinement property is that coarse- to medium-grained parallelism is available by assigning each cell in the initial decomposition to a separate process or processor. In fact, there is some choice in the level of granularity used. The initial work area can be broken into 4^n separate jobs.[3] The local refinement property ensures that each work area is indeed independent of the others.

Two advantages of Griffiths' roughing scheme are that explicit 2D geometry is not required to describe each level and that his skipping scheme provides an alternative to the types of region connectivity we saw in Chapter 2. However, it is not clear how much tool movement could be involved in skipping successive regions or if such compound skips are combined into just one move. More serious is the necessity to pass the tool along the edge of each layer as described in Figure 2.5.

[3] Griffiths' implementation splits each region in half, so 2^n separate jobs are possible.

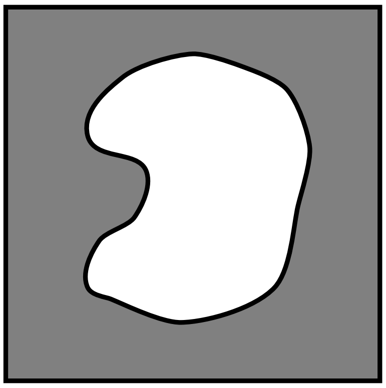


Figure 3.8: Example of a non-square object and a bounding Hilbert region. The path should have as few points in the shaded area as possible, for efficiency reasons.

It is not clear how to do this using Griffiths' idea. Hence, Griffiths' roughing path method is only part of a complete roughing path generation system.

Related to edge cleanup is the ability to machine non-square objects, especially collections of regions that have one or more closed boundaries. The meandering Hilbert curve must not be allowed inside the forbidden area and simply lifting the cutter when the path passes through such a region is not a very practical alternative. The path should also keep the cutter on the work. Portions of the path that have no contact with the work should be removed. See Figure 3.8.

An advantage arising from the use of offset surfaces and ray tracing is that gouging is not a concern, nor is the need to remove loops from offset surfaces. A problem that remains is gaps that form between offset surfaces. These must be filled. The use of surfaces and ray tracing does not allow easy determination of where excess material has been left behind. Hence, it could be difficult to implement cleanup cuts using Griffiths' approach. A limitation of using *uniform* offset surfaces is that only ball endmills can be used. This is a serious restriction as many roughing

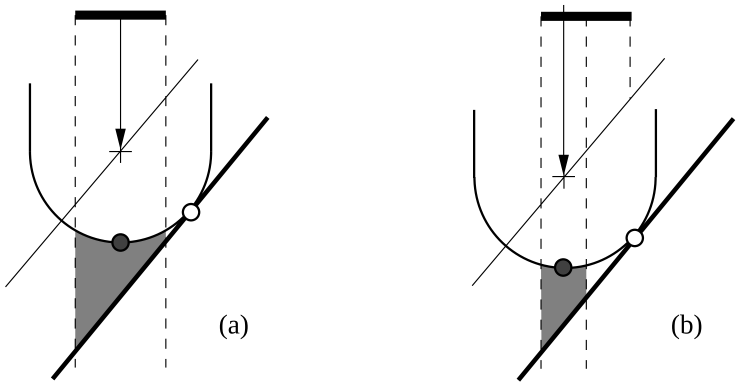


Figure 3.9: Termination problem for tool tip based tool positioning. When the surface slopes steeply, the contact point with the surface is on the edge of the tool. It may also be outside the work area, meaning further refinement will not improve the situation.

paths use flat or toroidal cutters, not ball endmills.[4]

An unclear but important part of Griffiths' approach is the criteria used for subdivision termination. Griffiths does not state precisely how his algorithm determines when to stop. This is an important part of the method. It is possible that the accuracy assessment may lead to a failure to terminate. This can happen when the tool tip is centred on the midpoint of the work area and the surface normal tilts steeply away from the tool axis. When this happens, the point of contact between the tool tip and the surface no longer lies within the work area. See Figure 3.9. This means the work area might *not* be cut sufficiently accurately and that further refinement may not improve the situation because the smaller work area is in fact farther from the contact point. It is not clear from his description if Griffiths' implementation suffers this problem.

[4]It might be possible to use a *general tool offset surface*, as described by Brechner [Bre90a, Bre90b], but gaps between surfaces would still be a problem.

In the next chapter, the Hilbert curve path planning algorithm is adapted to a hybrid method that uses both Hilbert curves and pixel models. The Hilbert portion of the method differs only in that the midpoint of the work area represents the *contact point* of the tool and object, rather than the tool tip. Making this change avoids the problem illustrated in Figure 3.9.

Chapter 4

A Hybrid Method

A hybrid pixel model and space-filling curve method is the subject of this chapter. The method uses the Hilbert-driven refinement of the last chapter along with pixel-based tool positioning and accuracy assessment. The work areas or *Hilbert regions*, as they are called in this chapter, form a partition of the initial Hilbert region. A pixel-based assessment method ensures that each region is machined to within tolerance by positioning the contact point of the tool at the midpoint. If this is not possible, further subdivision is performed. Covering the object with in-tolerance Hilbert regions in this manner leads to a tool path that recreates the given shape to within the specified tolerance. The method retains major benefits of both the pixel model approach and the Hilbert curve and ray tracing method discussed in the last chapter.

Section 1 of this chapter outlines the hybrid method and Section 2 describes an implementation. Particular attention is given to accuracy assessment and the handling of containment and exclusion profiles. Section 3 ends the chapter by presenting some results. Included are sample paths, photographs of machined objects and a discussion of some problems and issues that appeared during implementation.

4.1 The Hybrid Approach

The pixel model has several advantages while its primary deficiency is its storage requirements and processing time. This section describes a hybrid system that uses the local refinement property of Hilbert curves to limit the amount of storage required for pixel models. The method retains several key pixel model advantages.

The basic idea is simple: replace the ray casting and offset surface model with a pixel model. Doing so makes sense as the Hilbert path planning method works in the plane hence it is independent of the tool positioning algorithm. The subdivision algorithm needs only to know when to terminate, but any tool positioning algorithm providing such information can be used. Thus, pixel tool positioning and accuracy assessment are compatible with Hilbert path planning.

To reduce storage demands, a *local* pixel model is used. The storage for the pixel model is reused on each call to the tool positioning algorithm. Thus, there is no global pixel model, only a collection of Hilbert regions and a buffer of storage for the pixel model. Another buffer is used for tool positioning.

The combination of Hilbert-driven refinement and local pixel models is sufficient to retain the obvious benefits of both methods. Griffiths' implementation features gouge-free tool positioning using offset surfaces and ray tracing; moving to pixel tool positioning using a highest point method retains that property. In addition, the need to perform explicit seam handling disappears, as the pixel representation takes care of it for free.

On the pixel side, the local pixel model can be used to model general tool shapes; there is no *a priori* reason ball endmills must be used. As well, collision detection between tool holder and workpiece is possible as is testing for tool shank and part interference. The cost of doing so is a slightly larger pixel buffer. It is not immediately clear, however, if the potential to perform cleanup cuts or to do tool load estimation still exists.

4.1.1 Tool Positioning and Accuracy Assessment

The termination criterion for the Hilbert subdivision must ensure that as Hilbert regions become smaller they also get closer to the contact point with the tool. Doing so avoids the problem illustrated in Figure 3.9. A simple solution is to use a contact point positioning algorithm and to let the midpoint of the Hilbert region be the position of the tool-surface contact point. Using the surface normal at the planned contact point, an initial tool tip position in the plane is found. The highest point method then finds the tool tip position in 3D Cartesian space. If the curvature of the surface beneath the cutter is everywhere less than that of the tool, then the planned contact point and the point of contact implied by the computed tool tip position will be the same. (There are cases where this assumption does not hold. Section 4.2.5 gives exceptions; Section 4.3.2 discusses some solutions.)

To assess the accuracy of the computed tool tip position, the grid of incremental height values for the tool tip is offset by the calculated tip position. The grid of

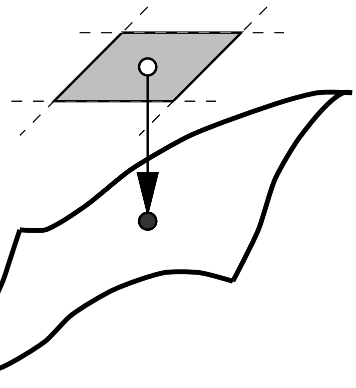


Figure 4.1: Contact point based tool positioning, part one. The centre of the Hilbert region defines the position of the contact point in the XY plane.

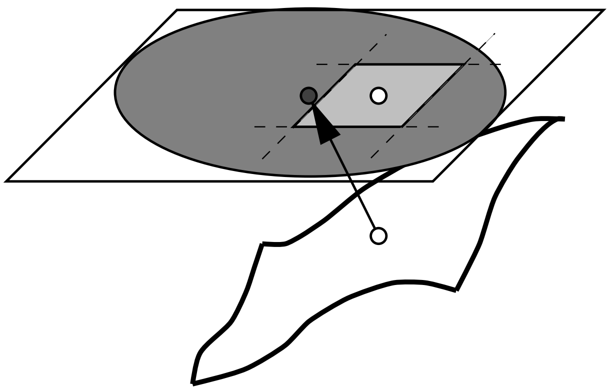


Figure 4.2: Contact point based tool positioning, part two. The surface normal vector is used to find the tool tip position in the plane.

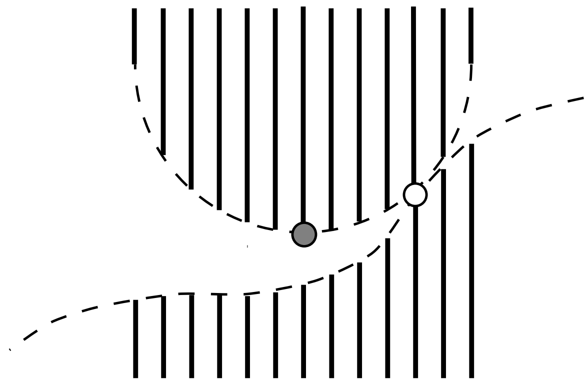


Figure 4.3: Contact point based tool positioning, part three. The highest point method is used to find the tool tip position in 3D Cartesian space.

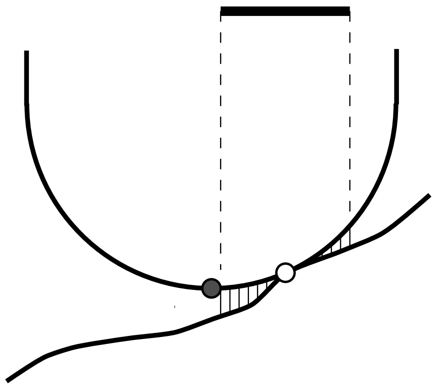


Figure 4.4: Contact point based tool positioning, part four. The maximum distance between tool tip and surface is measured within the Hilbert region. If the value is acceptable, the subdivision is terminated.

surface heights is subtracted and the resulting increments give a measure of how much deviation there is between the tool and the surface. (The deviation will be zero at the tool contact point, unless the tool was raised to avoid gouging.) The maximum difference between tool and surface *within the Hilbert region* indicates whether or not the surface is being cut sufficiently accurately within the region. If it is not, further subdivision is necessary. Figures 4.1 through 4.4 illustrate the method.

4.1.2 Profiles

Using a local pixel model avoids the storage cost of a global pixel model, but it does not get around the pixel model problem of poor edge definition. The Hilbert curve approach has the same problem; cleaning up a given level in roughing is not easy (if at all possible) to do. To avoid these shared difficulties, the hybrid approach uses explicit, planar bounding curves to define edges of roughing levels and to handle edges in general.

Introducing edge curves has a number of advantages and few disadvantages. The edge profiles can be offset by the tool radius to find the region in which the tool centre may travel freely (as in pocketing). Conversely, the offset edge profiles can be used to determine when the planned tool position is inside a forbidden region (or beyond the outer edge of the object shape). The offset profiles can be used to construct paths to remove excess material in level-by-level roughing.

A disadvantage of introducing explicit boundaries is that they limit the applicability of the method because the edge profiles must be supplied. In level-by-level roughing, the edge profiles are plane–surface intersection curves, which are not always straightforward to compute. However, this is not a serious restriction. Most surface modelling systems have plane-surface intersection facilities. In fact, the combination of Hilbert regions and profiles gives a slight generalization for both pocketing and surface machining. Pockets with open sides and surface machining paths with obstructions may both be handled using the same mechanism.

4.2 Implementation

The implementation of the hybrid scheme involves four elements: pixel model generation, Hilbert curve generation, tool positioning, and profile handling. Each are discussed in more detail below. Following that is a description of how the elements are combined to create tool paths.

4.2.1 Pixel Model Generation

A graphics library called OpenGL [Ope92, NDW93] is used to create the pixel grids. OpenGL has facilities to define NURB surfaces, matrices, and to render pixel images. For tool path applications, an orthographic or parallel projection matrix must be used. To create a pixel image, a collection of surfaces are given to OpenGL which tessellates them, projects the nodes onto the pixel plane, and performs the linear interpolation to obtain depths for each pixel. If graphics hardware is available, OpenGL uses it to create the pixel image. After the image is created, the program uses an OpenGL function call to copy the depth data into a buffer in the program's address space.

An interesting feature of OpenGL is an additional type of pixel image known as a *stencil buffer*. The stencil buffer is the same size as the image of depth data and is cleared to zero before a depth image is computed. Pixels values in the stencil buffer are incremented by one each time the corresponding depth pixel is written to. When the depth image is complete, pixels in the stencil buffer that are non-zero indicate pixels in the depth image that contain valid data.

The stencil buffer is used to determine if a given position is completely off the model by checking that all stencil buffer values for the current stencil image are zero. If so, the depth image contains no pixels that are part of the object hence the tool cannot be positioned using the highest point method.

4.2.2 Hilbert Curve Generation

The technique used to create Hilbert curves is based on a method from a 1985 paper by Griffiths [Gri85]. Rather than generate line segments or curve vertices, the method subdivides a given window into subwindows. Drawing lines through the midpoint of the subwindows in the order they are computed creates a Hilbert curve. (Griffiths' tool path generation method also uses this technique.) An interesting feature of the algorithm is that windows are divided in two, not four, but the Hilbert ordering is still retained.

Griffiths' subdivision algorithm represents the initial window with a record data structure specifying the corners of the window. Hence, the initial region could be non-square. In the hybrid system implementation, the initial window is constrained to be square so that subwindows will also be square.

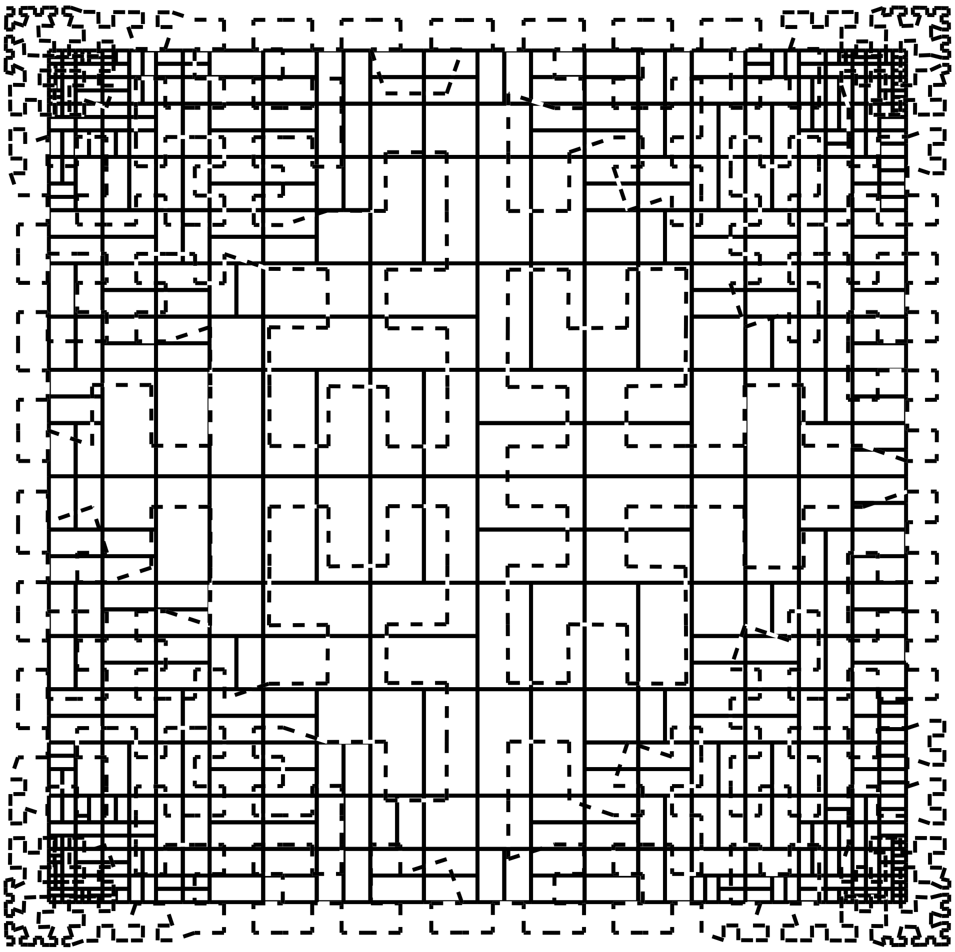


Figure 4.5: Example of a Hilbert decomposition. The boundaries of the Hilbert regions are solid lines; the dashed line represents the path of the tool tip. The shape is a sphere of radius 2.3 inches; tool size is $\frac{1}{2}$ inch. Points on the path drift away from their corresponding Hilbert region midpoints wherever the surface normal tilts away from the tool axis.

As the window is subdivided, smaller subwindows are created and stored in a binary tree. Thus, an entire collection of windows of various sizes is stored, giving a quad-tree decomposition of the initial region. The Hilbert ordering property, important for path generation, is built into the nodes of the tree. Hence the Hilbert tree contains more information than a pure quadtree. Figure 4.5 shows a Hilbert decomposition. Rectangular regions are those created by halving the parent region.

4.2.3 Tool Positioning and Accuracy Assessment

The tool positioning technique for the hybrid scheme is different from that used by Griffiths in order to ensure that the problem discussed at the end of the last chapter does not arise. Recall that if the tool tip is aligned with the middle of the work area (i.e. the tool axis passes through the midpoint of the Hilbert region) the recursive subdivision may fail to terminate in steep regions because the contact point is no longer within the work area. The hybrid method therefore places the *contact point* of the tool at the midpoint of the Hilbert region, meaning that as the Hilbert regions become smaller, the maximum distance between tool and surface becomes smaller. Since the deviation decreases as the region size decreases, the Hilbert subdivision will eventually terminate. Exceptions occur in the case of steep regions, where convergence will be slower, and in highly curved areas, where the distance cannot decrease beyond a certain amount; see Section 4.3.2 for a discussion.

One disadvantage of the proposed method is that flat endmills cannot be used because the contact point is usually on the outer edge of the cutter; a solution is discussed in Section 4.3.2.

The positioning technique requires the surface normal vector at the planned point on the object. OpenGL does not include facilities for creating grids of surface normals nor does it have the ability to find the surface normal at an arbitrary pixel. Thus, the normal vector at the planned point of contact is estimated using a second pixel model, centred on the contact point. Four normal vectors are constructed (one for each quadrant) and the mean value is taken to be the surface normal.

The Hilbert region is defined by the current subwindow in the refinement scheme outlined above. The mid-point of the Hilbert region gives the planned contact point; this is the point about which the first pixel model is constructed for surface normal estimation. The surface normal and tool ball radius are used to compute the coordinates of the tool centre with respect to the contact point. A second tool-sized pixel model is created about the tool centre and the highest point algorithm finds the Z value of the tool tip at that location.

To assess the accuracy of the computed tool position, the Z value of the tool tip pixels that fall inside the current Hilbert region are compared to the Z value of the corresponding surface pixels. The maximum difference is returned to the Hilbert subdivision algorithm along with the computed tool tip position. The size of the Hilbert window reduces with each subdivision, but the size of the pixels used in the tool shadow region and the accuracy assessment remains constant. Thus, for small Hilbert regions, only a few pixels near the tool tip are used.

A constraint on the tool positioning algorithm is that the Hilbert region must lie entirely within the tool shadow region. This ensures that the tool covers the Hilbert region and does not leave any material unremoved. As described earlier, ensuring

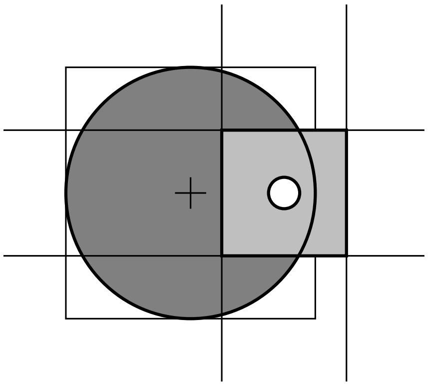


Figure 4.6: Example of an invalid tool position. The Hilbert region (light shade) falls partially outside the tool shadow region (dark shade).

each region is cut within tolerance in a Hilbert partitioning of the object ensures an accurately machined shape. Figure 4.6 shows a tool position where the Hilbert region is partially outside the tool shadow region; this may happen in areas with steep slopes. The positioning algorithm returns a code to indicate if the proposed position is valid.

Since the initial Hilbert region may be larger than the object, there is potential for planned tool positions to lie entirely outside the object. The stencil buffer pixel model is used, as described above, to determine if a given position is completely off the model. In such cases, the positioning routine returns an appropriate code. If the planned position is partially off the model, the position is acceptable but only valid pixels are used to assess the accuracy of the tool position.

4.2.4 Profile Handling

Profiles are specified as a connected sequence of planar non-uniform B-spline curves. A given profile may contain more than one B-spline curve, but the endpoints of

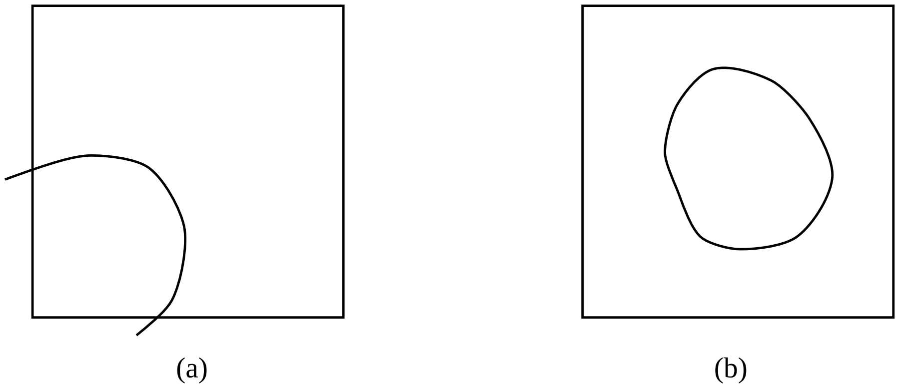


Figure 4.7: Example of open and closed profiles.

successive curves must be coincident and the curves must have the same direction. There are two types of profiles, *open* and *closed*. Closed profiles have coincident start and end points and thus form a closed curve. Open profiles do not have coincident start and end points, but must intersect the initial Hilbert region twice. Figure 4.7 gives two examples.

There can be at most two profiles specified; each profile is converted to a planar polygonal approximation. If desired, each polygonal profile can have an offset applied to it. The offset can be on the left or right side, as determined by the direction of the profile. After the offset operation, open profiles are closed by using the initial Hilbert region to define additional polygon segments. The closed profiles are merged with each other, if they intersect. A restriction of the implementation is that there may be only one resulting profile. If two profiles remain after merging, only the first is used.

The profile may be either a *containment region* or an *exclusion zone*. Exclusion zones are a generalization of islands; the path does not enter the zone. A containment region has the opposite effect as the path is constrained to lie inside it. Note

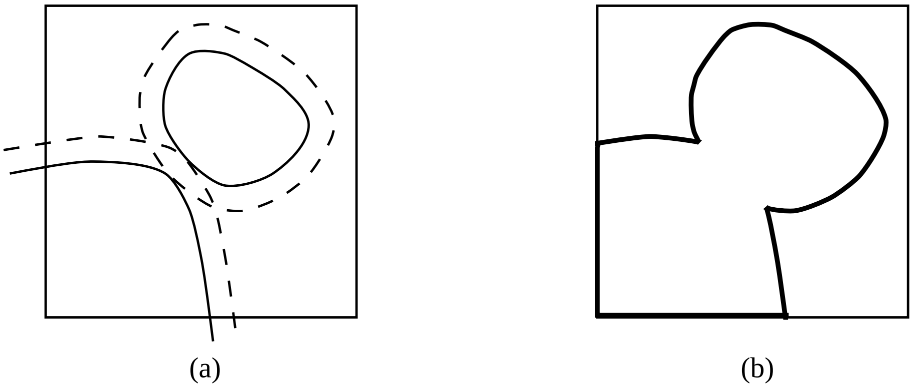


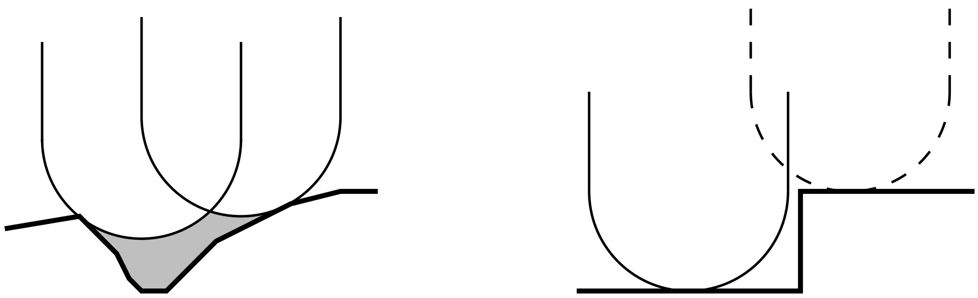
Figure 4.8: Example of two profiles being merged. In (a), an open and a closed profile are offset (dashed). After being offset, the open profile is closed against the Hilbert region then merged with the other closed profile. The region could be a containment or an exclusion region.

that the initial Hilbert region becomes a kind of initial open containment region as the tool never goes completely beyond it. In the interior, the path may be restricted by another containment region or it might be kept from entering an exclusion zone.

4.2.5 Path Generation

The four components are combined to create tool paths in the following way. An initial Hilbert window is defined on the plane $z = 0$; the tool axis is parallel to the Z axis. The initial Hilbert region is recursively subdivided without accuracy assessment into regions that are approximately 70% of the tool diameter.[1] For roughing paths, this is the end of Hilbert subdivision. For finishing paths, the pixel

[1]The value 70% is chosen because that is slightly less than the size of the square inscribed in the circle of the tool. Making the regions this size ensures that no excess remains whenever the tool turns a corner.



(a) (b)

Figure 4.9: Problems with concave regions and jump discontinuities. In case (a), further subdivision may help, but the tool will not be able to remove all the material. In case (b), the solid tool outline position will leave some material uncut in the corner. The dashed position will never be within tolerance, even if further subdivision is applied.

model tool positioning and assessment algorithm are used as soon as the region size less than or equal to 70%.

The finishing path subdivision continues until the region is cut within tolerance or until the regions become very small. Regions can become very small if the tool is near a surface concavity or if the tool is along the edge of a *jump discontinuity*. In both cases, a small amount of subdivision may help, but there is a point where further processing will not reduce the measured deviation, as is apparent from Figure 4.9.

The Hilbert decomposition constructs regions in Hilbert order, associating a path point with each leaf region in the Hilbert tree. The zigzag reordering step recursively descends through the tree of regions and reorders all points associated with a given region (as shown in Figure 3.6). In extreme cases, the initial Hilbert region may be reordered so that every point lies on a zigzag path, with no Hilbert path segments left. The other extreme is no reordering, where the reordering criteria are satisfied by a region containing only one point.

For roughing paths, points that are below the current cutting level are translated up to it. The zigzag simplification reduces each straightline zigzag pass to two points, as in Griffiths' implementation. Griffiths' region skipping algorithm is used to remove sets of points from Hilbert regions lying entirely above the current level (see Figure 3.7).

The containment or exclusion profile, if specified, is used to clip the path after zigzag simplification. In the implementation, a pass around strategy (like that shown in Figure 2.9(b)) is used to connect portions of the original path disconnected by one or more boundaries. The simplified path is traversed and crossings from active region to forbidden region are detected. Where the path moves from an active zone into an exclusion area, the point of intersection is added to the path and the points inside the excluded area are discarded. The excluded portion is followed until another crossing is detected; points on the exclusion zone boundary from the initial intersection point to the second intersection point are used to build path points to add to the path. In this way, the excluded portion is removed and the tool follows the boundary to the other side of the interruption and resumes using points on the simplified path.

The difference between a containment and an exclusion zone is essentially one of parity for the crossings test. If the tool starts in an exclusion zone, this fact is recorded and the path begins where it exits the zone. The next crossing will be the start of an entry-exit pair. If the path ends in an exclusion zone, the last point on the path terminates the algorithm, the intersection with the crossing into the zone having already been added to the path. Beginning the path inside a containment region is the same as being in an active portion: points are output because no entry into a forbidden region has been detected so far. This is the default condition.

The implementation reads the surface and profile information from an input file while tool size, type of path and machining tolerance are given on the command line. A sample input file is shown in Appendix B. The output of the program is a sequence of path points formatted for viewing in a 3D visualizer.

4.3 Results

This section contains results obtained using the implementation of the last section. Examples of various types of paths are included as well as photographs of results. The section ends with a discussion of some problems and issues stemming from experiences with the implementation.

4.3.1 Tool Path Examples

An example path, based on the initial Hilbert decomposition, is shown in Figure 4.10. The object is the top half of a hemisphere of radius 2.3 inches. The square above the path represents the initial Hilbert region; its size is 3.0 by 3.0 inches. The cutter is a $\frac{1}{2}$ inch ball endmill. The path points in this case are defined by the midpoint of the initial subdivision; the tool is dropped vertically onto the object. Figure 4.11 shows the effect of further refinement. Each Hilbert region is half as big as in the previous path. Figure 4.12 shows the path used as a point-

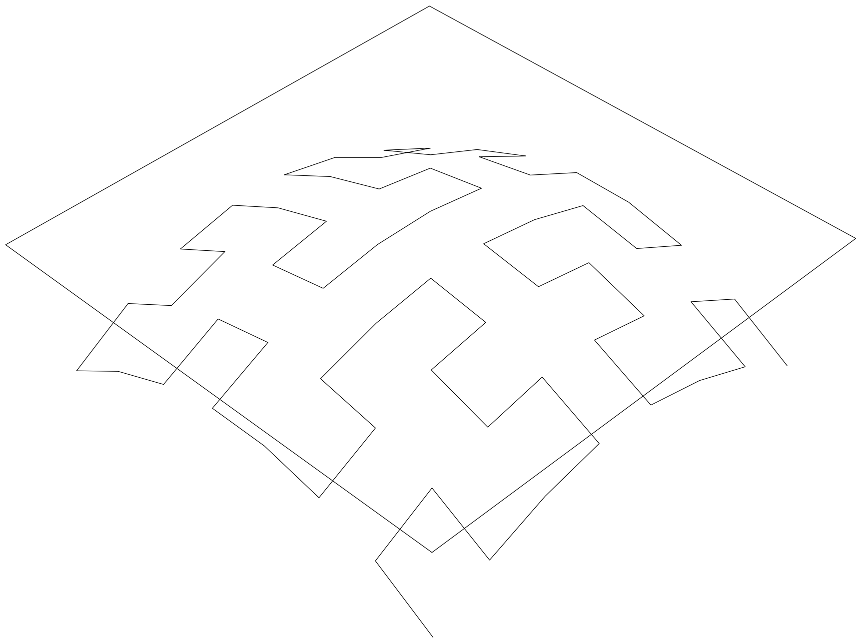


Figure 4.10: Path points for an initial Hilbert subdivision. The shape is a 2.3 inch radius hemisphere; the Hilbert region is 3.0 by 3.0 inches. Each Hilbert region is 70% of the diameter of the tool, $\frac{1}{2}$ inch in this case.

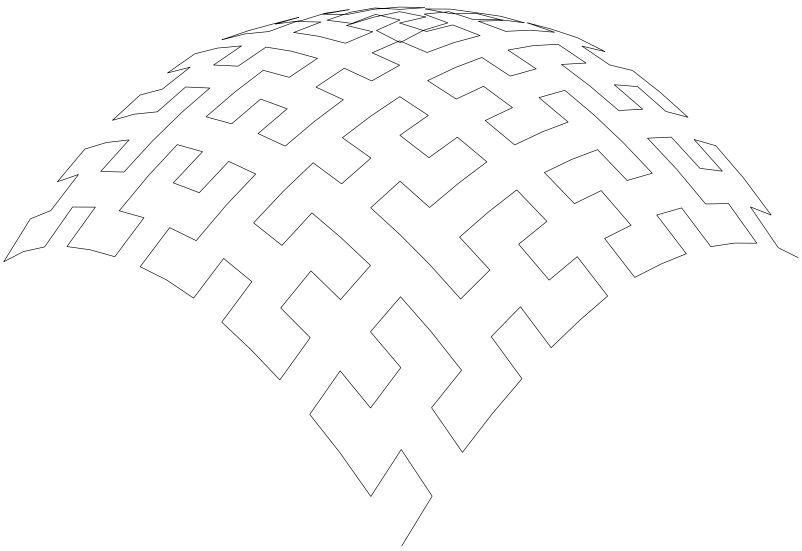


Figure 4.11: Further refinement of the initial Hilbert path. In this case, each region is half the previous size.

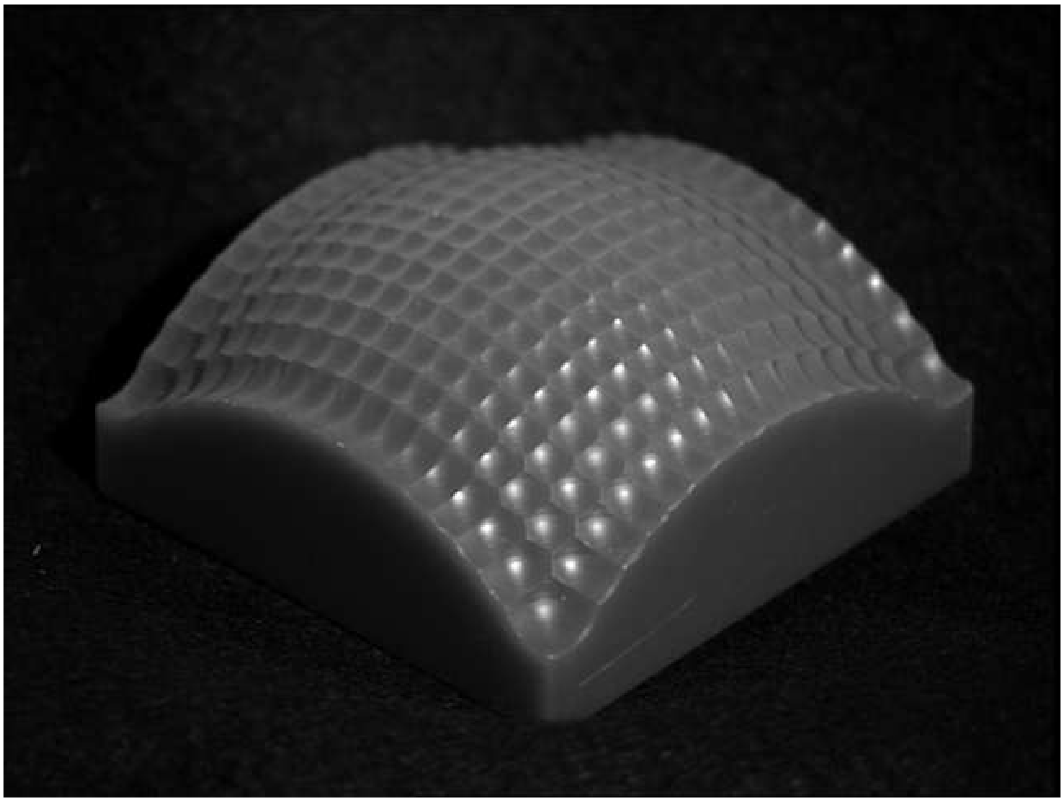


Figure 4.12: An object machined using point-based roughing. The points are from the path in the previous figure.

based roughing path. The tool plunges vertically to each point in the path from a position above the top of the raw billet of material.

Figure 4.13 shows the Hilbert-ordered points for a surface machining path for a hemisphere of radius 2.8, using a $\frac{1}{2}$ inch ball tool and a 3.0 by 3.0 inch Hilbert region; the tolerance is 0.095 inches. Note that in the corners the sample density increases due to the increasing slope of the surface. Figure 4.14 shows the result of the zigzag simplification process on the path; an object machined with the path is shown in Figure 4.15. Areas where the sample density is constant are refined, but the transition regions are not refined. The result is a path that looks like a collection of small zigzag paths connected in some arbitrary manner.

Note too the somewhat skewed tool movements in the transition between densities. These occur because the midpoints of smaller regions do not, in general, line up with the midpoints of larger regions. For shapes with reasonable features, the sampling density increases and decreases continuously; n-fold jumps in sampling density do not occur. In any case, the greater the difference in sample density, the closer the smaller Hilbert region will be to the larger. This means the more the tool movement deviates from centre, the smaller the adjoining region must be, which implies it must be closer to the midpoint of the larger region. Hence, there is a bound on the amount of skew in the path movements.

Figure 4.16 illustrates a sculptured shape 3.0 by 3.0 inches machined with a $\frac{1}{2}$ inch ball endmill. In Figure 4.17, a containment region is used to restrict the path to a non-square domain. Figure 4.18 shows the path for the same shape, but with an exclusion zone. In both cases, no offset has been applied to the profile though,

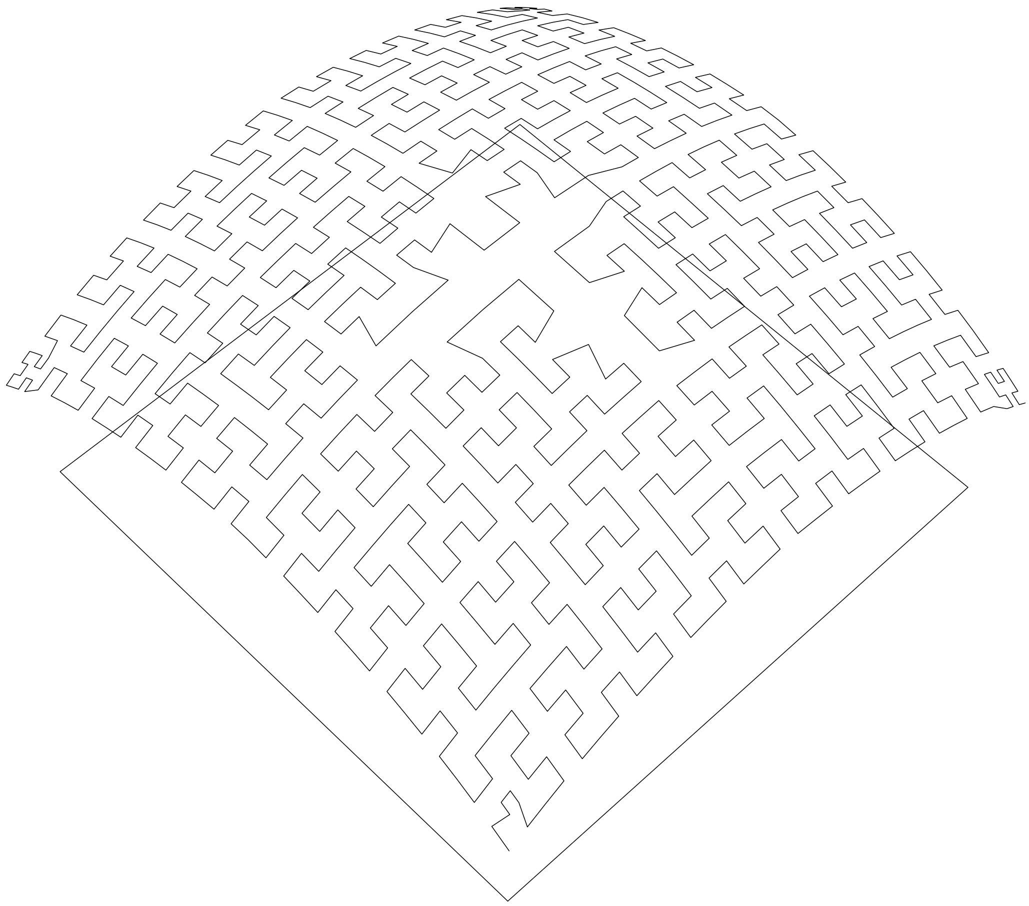


Figure 4.13: Points for a finish machining path.

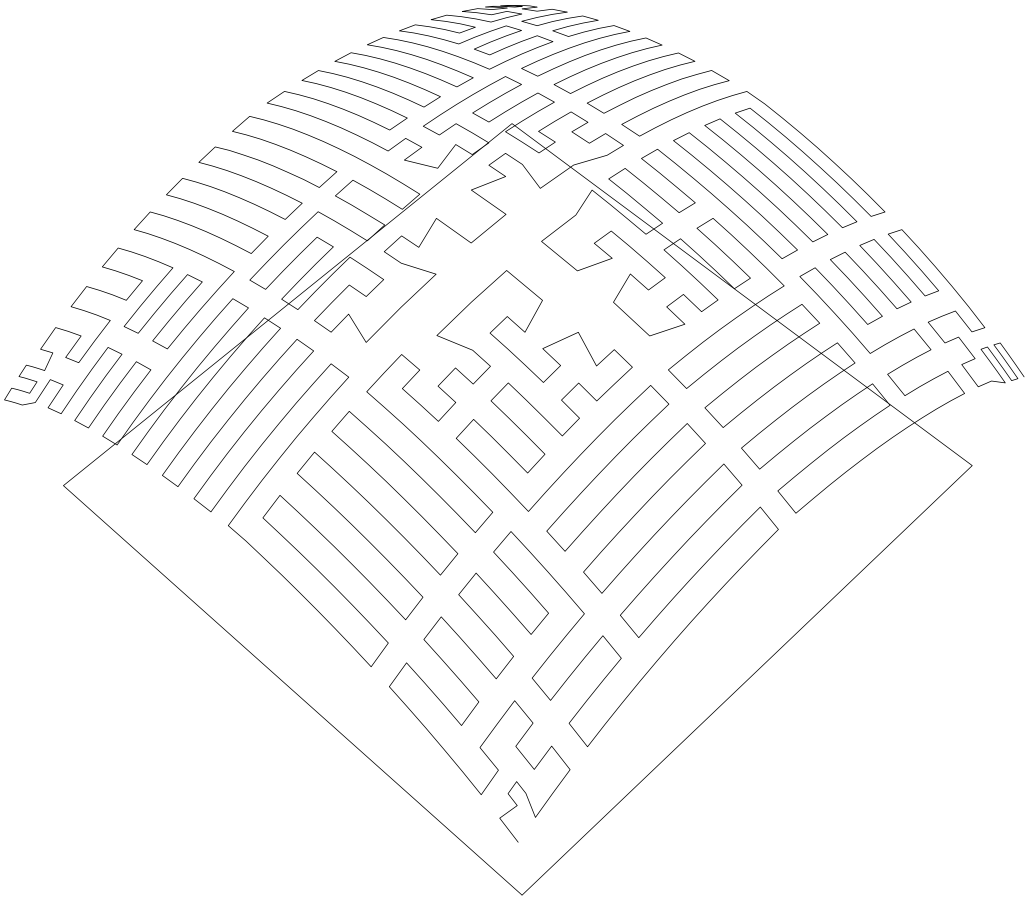


Figure 4.14: Reordered points for a finish machining path.

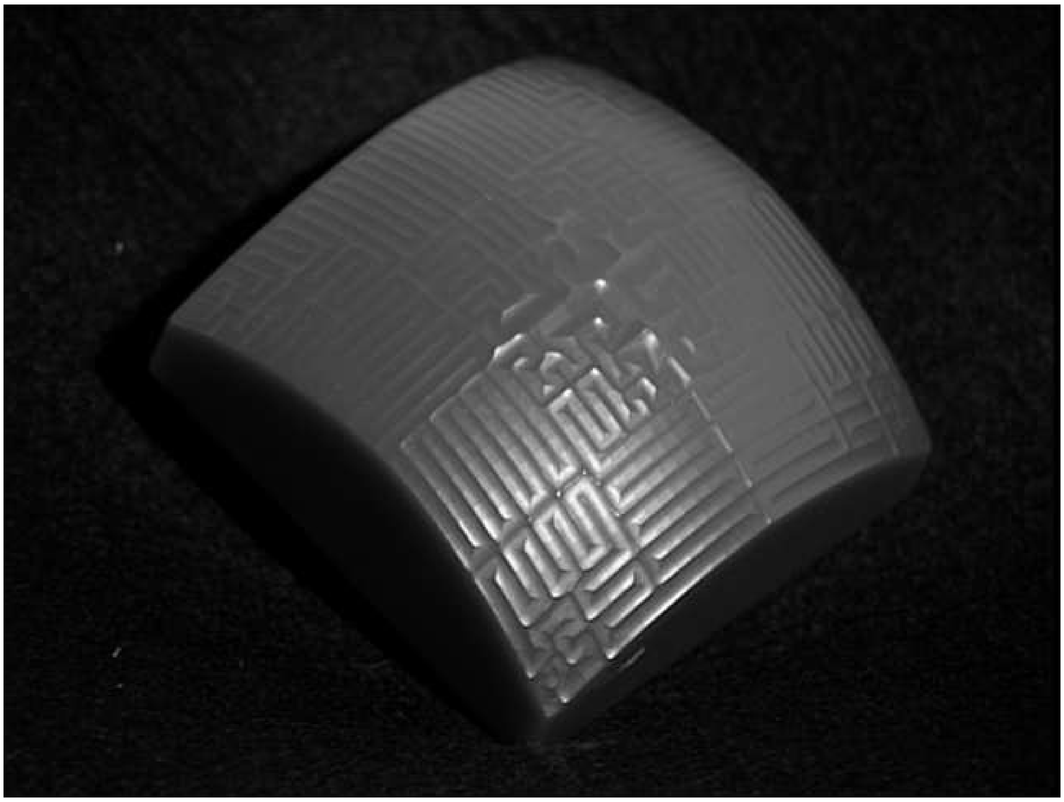


Figure 4.15: Sample machined part, using the path from the previous figure.

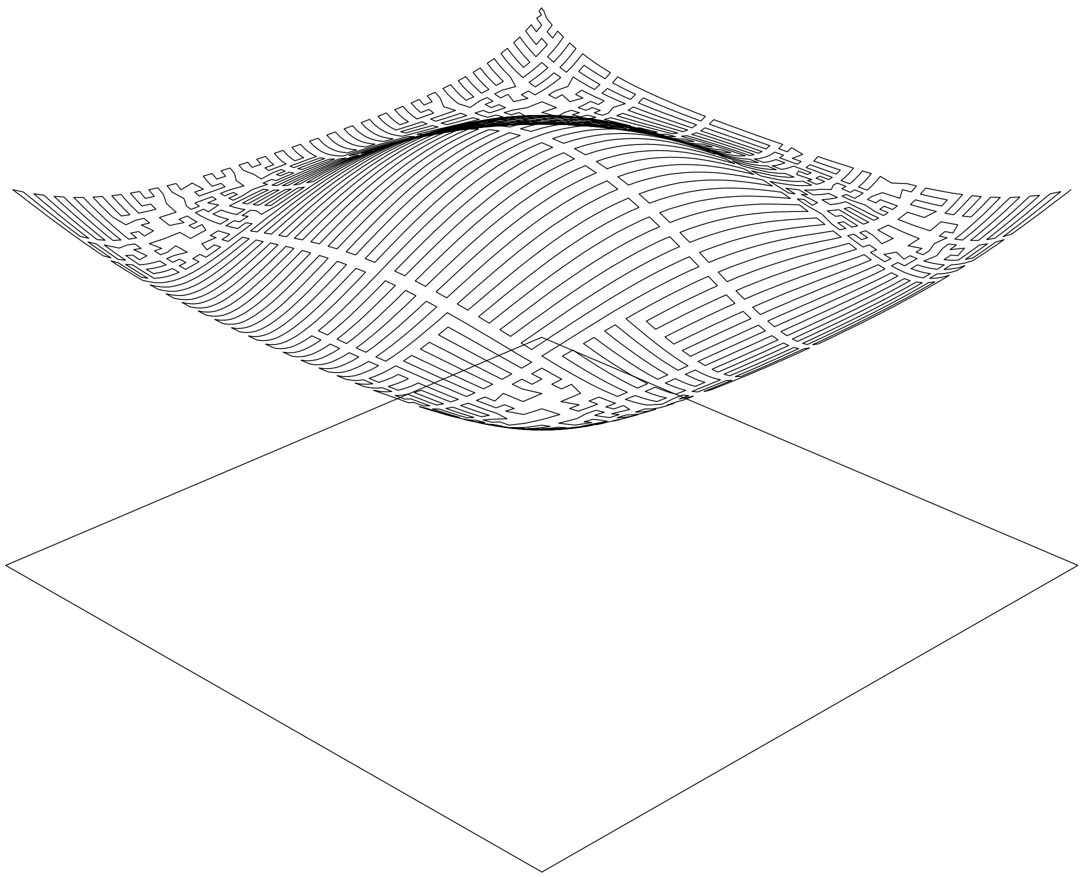


Figure 4.16: Surface machining path for a sculptured shape. The same shape is used in the next two figures to demonstrate containment and exclusion regions.

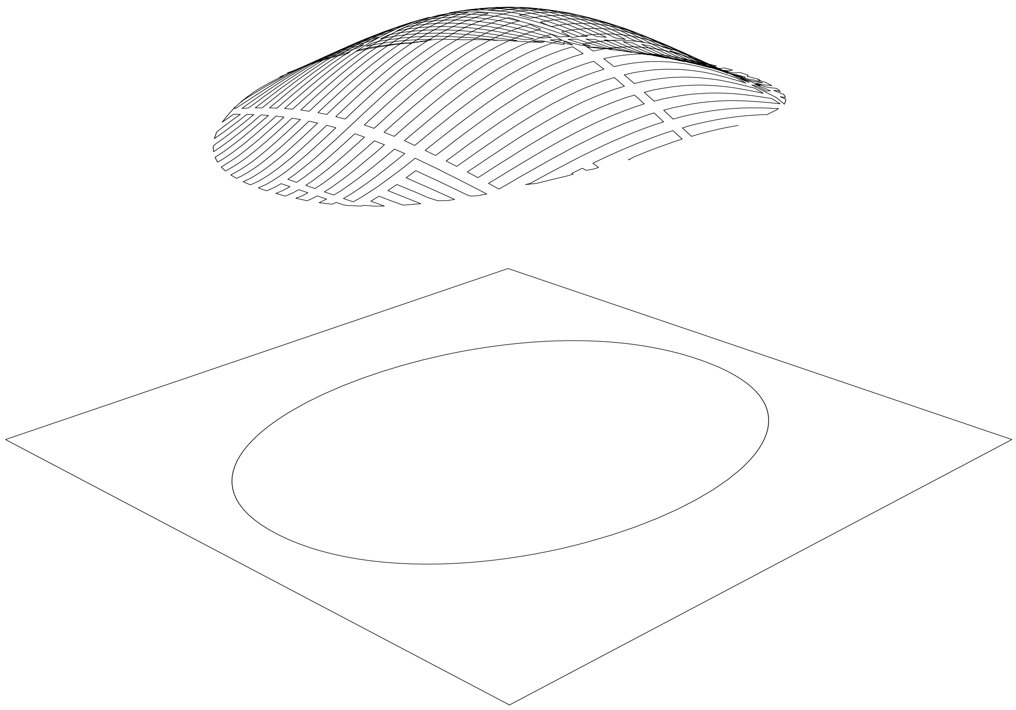


Figure 4.17: Surface machining path with a containment region.

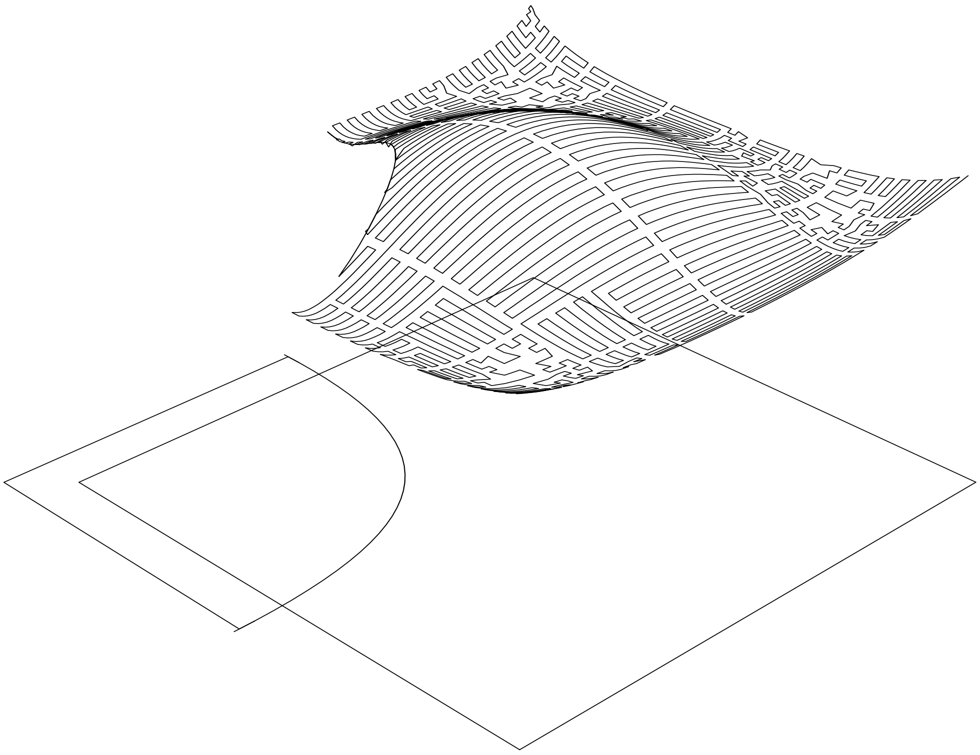


Figure 4.18: Surface machining path with an exclusion zone.

typically, the profile would be offset by the cutter radius on the active region side.

Figure 4.19 contains a roughing path for a small hemisphere of radius one inch. The bounding region is again 3.0 by 3.0 inches. The path uses the Griffiths region skipping strategy. Note that path points that change levels are connected directly. This is slightly incorrect, as the cutter should move up then over for an upward movement and over then down for a downward movement. The implementation was not corrected because the profile-based paradigm provides an alternative, as shown in Figure 4.20. The same shape is cut with a roughing path on the same level, but the non-cutting portion has been truncated by a profile curve. The profile is the intersection of the object and a plane on the current cutting level, offset by the tool radius. Figure 4.21 shows the same path with a profile path added to clean up along the object–cutting level intersection. The machined object is shown in Figure 4.22. Note that cleanup cuts have produced smooth circular edges on each level, but that unremoved vertical scallops remain on the front and back walls. These have been left on purpose; normally, they would be removed.

Figure 4.23 shows how the hybrid method could apply to pocketing. Islands are modelled by exclusion zones; in this example, they are closer than one tool diameter so they merge when each profile is offset. The tool clears out the pocket and stays out of the exclusion zones.

4.3.2 Discussion

A number of issues arose while implementing and using the hybrid method. These include general implementation concerns and problems with point-based roughing

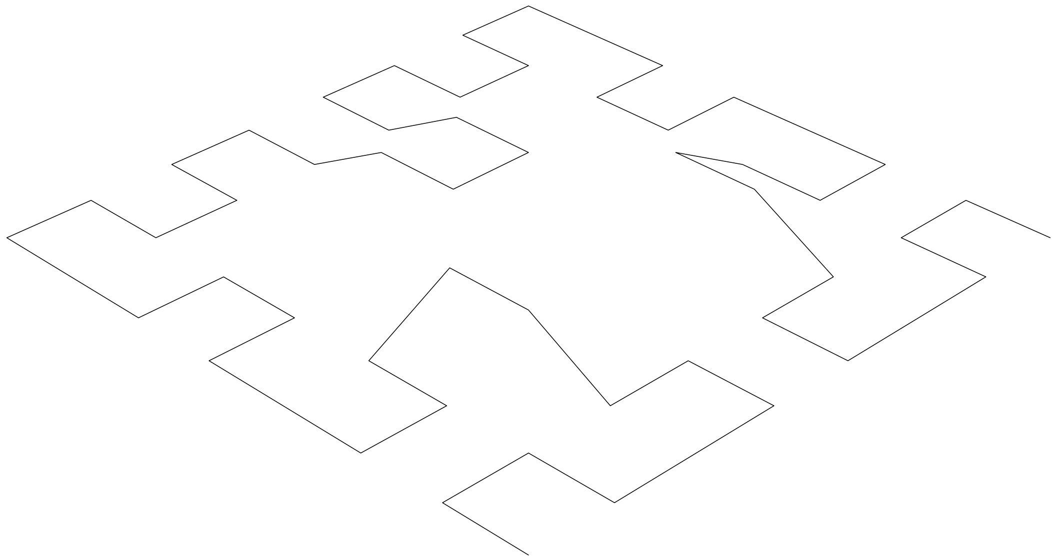


Figure 4.19: Roughing path based on Griffiths' method. The initial Hilbert region is 3.0 by 3.0 inches; the shape is a hemisphere having a radius of one inch. The cutting level is 0.2 inches below the top of the hemisphere. The tool is a $\frac{1}{2}$ inch endmill.

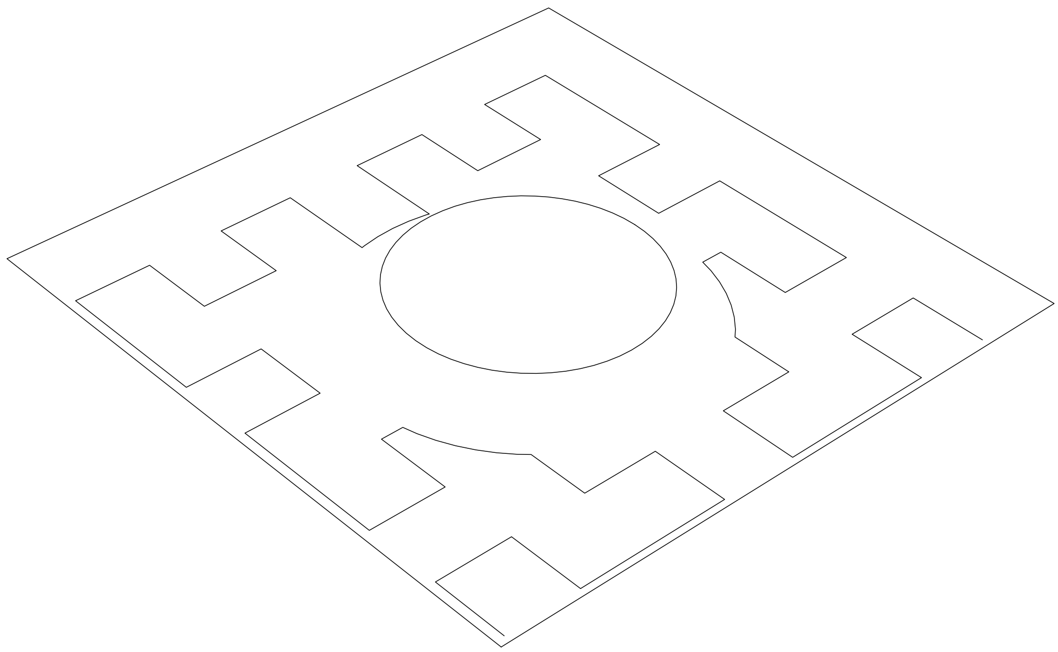


Figure 4.20: Hilbert-based roughing path using profiles. The object, tool, and cutting level are the same as in the previous figure.

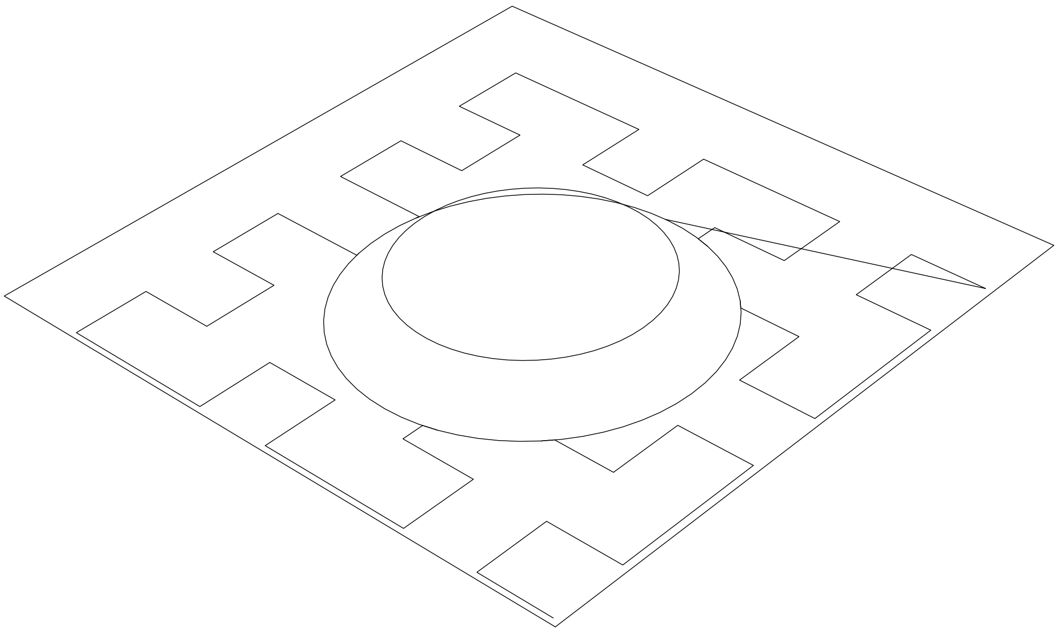


Figure 4.21: Hilbert-based roughing path with a boundary cleanup pass. The path is the same as the last figure, except a profile path has been added to trace around the uncut region in the middle.

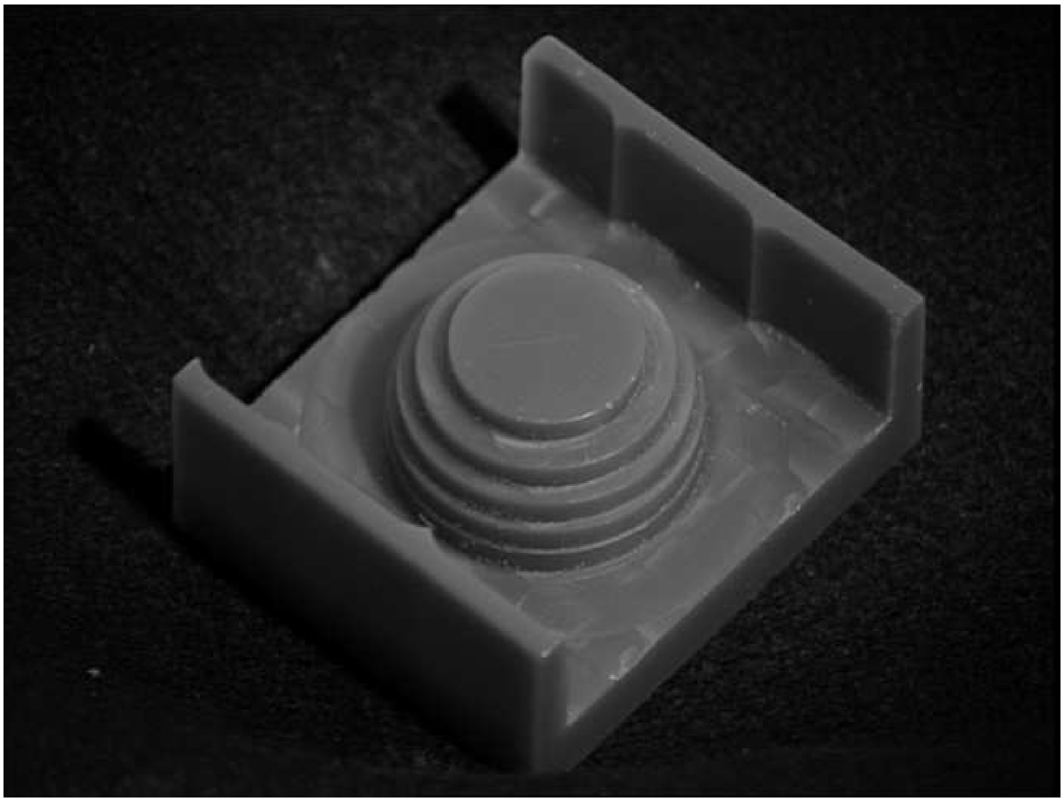


Figure 4.22: Rough machined part, using level-by-level roughing. The object is a one inch radius hemisphere and each level is 0.2 inches high. Roughing paths in the previous figures are for the top level only.

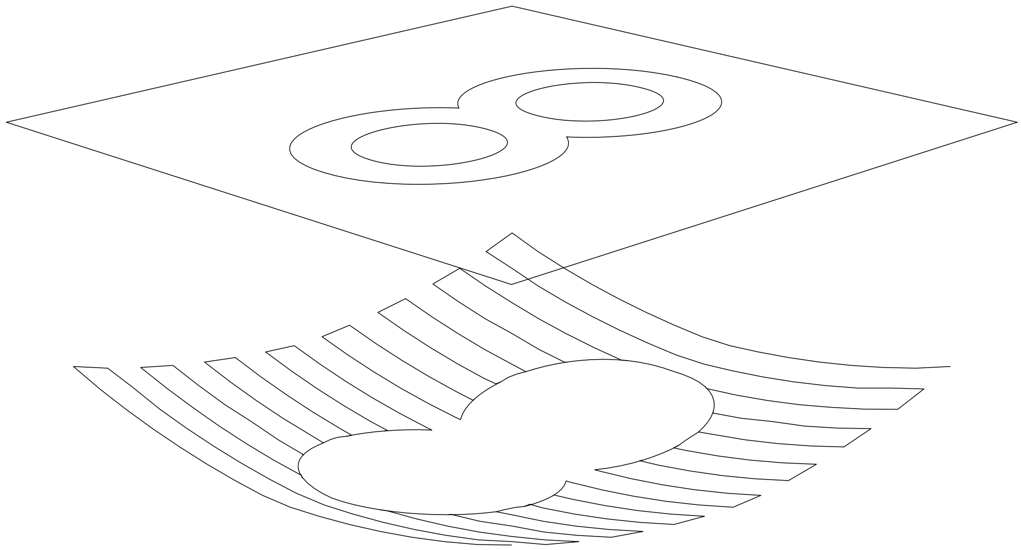


Figure 4.23: Application of the hybrid method to pocketing.

and point reordering. Tool positioning and accuracy assessment is the subject of a somewhat more lengthy discussion.

General Implementation Issues

A few issues appeared during implementation that deserve mention here.

The surface normal vector estimation technique only worked for smooth, gently curving shapes. Shapes with sharp corners and wrinkle-like folds lead to inaccurate vectors, meaning the tool tip deviated considerably from the expected tool position. Interestingly, the tool did not gouge the work at these locations since the highest point method removed any gouging that might have otherwise occurred.

The pixel model creation turned out to be computationally more expensive than anticipated. The need to estimate the normal vector with a second pixel model doubled the cost of each tool position. Using a rendering facility that computes accurate normal vectors would help. Another idea would be to use a point-in-polygon test to avoid calling the tool positioning routine unless the tool is in an active zone. Currently, the region handling is independent of the path planning step.

Conceptually, the graphics hardware can be used to create both the tool and local surface pixel models as the hardware implements the grid creation algorithm described in Section 3.1. In practice however, the use of current workstation graphics hardware is a limiting factor for implementations. The resolution of a typical high performance workstation monitor is 1280 by 1024 pixels. This means that large cutters cannot be rendered at high resolution; there are not enough pixels available to create the pixel model in one operation. Of course, the pixel model

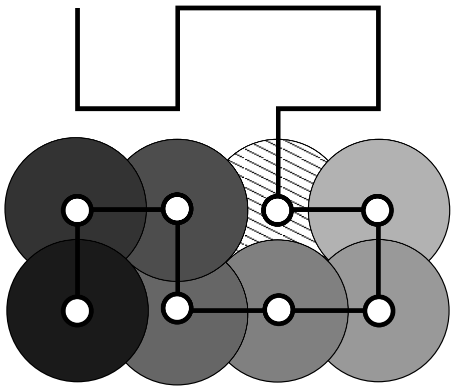


Figure 4.24: Inconsistent load in point-based roughing. The hatched region contains little material.

could be broken into four (one per quadrant) as the positioning algorithm is easily parallelized.

Point-based Roughing

A pleasant surprise with the hybrid implementation was that point-based roughing was obtained essentially for free. The Hilbert sampling is effectively a point-wise area fill – exactly the property required for making point-based roughing paths. The path data from the initial Hilbert decomposition was used to instruct a ball endmill to "drill" to the specified depth starting from above the object, as was shown in Figure 4.12. Note that while a ball endmill was used for proof of concept, a chisel point drill is the tool of choice for practical applications. In such cases, a pixel model of the conical drill tip would be used in tool positioning, not the usual spherical shape.

The method shows promise, but there is a problem. The Hilbert ordering on the tool positions cause the tool to leave some regions temporarily uncut while cutting most of their neighbours. The spacing of the path points, at 70% of the tool diameter assures that the tool shadow fully covers the Hilbert region, but it also means the tool removes some excess from neighbouring areas. Regions where most of the neighbours have been visited have little material left. Hence, some regions to be cut contain little material, others quite a lot. Figure 4.24 illustrates the problem. Reordering the points could help to even the load, but more investigation is needed.

Point Ordering

A problem with finish machining paths is that reordering is often limited in scope. The path becomes a collection of small zigzag subpaths, with too much Hilbert character remaining. It seems reasonable that more zigzag refinement is possible (consider for example the points in Figure 4.14); perhaps a less local approach to reordering would have more success.

Since the surface machining paths are based on tool tip positions, some reordered sections of the path retain some lateral changes in tool direction (as opposed to the case of all points on a pass being coplanar; see Figure 4.5). This is not a problem however, as the changes in direction are relatively small and infrequent.

A minor point related to point ordering is that finish machining paths, as implemented, contain all the generated points. In the case of a flat surface, the path is reordered to a pure zigzag topology but many points could be removed. Point removal would probably have to follow reordering; it is unclear if information from the Hilbert tree could be of help.

Tool Positioning and Accuracy Assessment

Most of the interesting results from the hybrid implementation concerned tool positioning and accuracy assessment. There were four major issues that became apparent:

- the accuracy assessment algorithm is too conservative;
- concave regions and jump discontinuities are not handled well;
- tool positioning along profiles could be improved;
- the use of Hilbert regions in tool positioning could be improved.

The accuracy assessment method uses difference in heights along the tool axis. For non-horizontal surfaces, the maximum vertical difference will be greater than maximum distance from the surface to the tool as measured perpendicular to the surface. An easy way to correct this is to multiply the maximum height by the cosine of the angle between the surface normal vector and the tool axis. In most cases this would reduce the value returned for maximum deviation, hence lead to quicker convergence of the Hilbert subdivision and fewer subdivisions.

Handling of concave regions and jump discontinuities turned out to be more complex than first thought. For concave regions, a method is required to detect their presence so the Hilbert subdivision does not attempt further refinement. It may be that a few more subdivisions be worthwhile, so just detecting and halting refinement may not be desirable. A possible solution would be to monitor the improvement in deviation as subdivision proceeds. Once the deviation does not improve, halt subdivision and mark the region as out of tolerance.

The handling of jump discontinuities is different and more difficult. A point does not stand in a *jump relation* with a given neighbour unless it is connected to it. Since points can be reordered, jump relations may appear and disappear. Thus, handling of the jump must wait until after point reordering. The problem with doing so is that jumps cause trouble for Hilbert subdivision at *point creation time.* Referring to Figure 4.9(b), the dashed tool position and refinements from it will always report an out of tolerance condition because some pixels in the tool shadow fall beyond the edge of the higher ledge. In this case, what is needed is to detect the case and to halt subdivision. Further Hilbert subdivision will not improve the situation. In any case, the two positions should not be linked directly as doing so would remove most of the feature.

The problem of tool positioning along profiles is related to handling jump discontinuities. If the profile represents a closed boundary, then the tool will move along the wall, cutting tangent to it. If a ball endmill is used, there will be a fillet of unremoved material left behind; this cannot be avoided, so accuracy assessment for profiles should not consider this excess. However, a profile path must still consider concave regions and jumps, as well as step along distance in the cutting direction. Thus, part of the tool shadow region must not be used for accuracy assessment, but what portion to use and what portion to ignore and how to determine this for a given tool position is unclear.

The most serious problem with the present accuracy assessment method is that it unnecessarily forces more subdivisions than are actually required. (This is partially related to the first point above.) In steeply sloping areas, the tool shadow

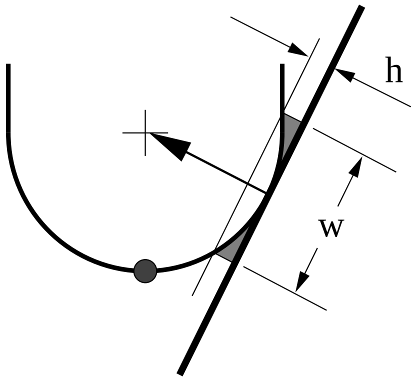


Figure 4.25: Extent of machined region in steep areas.

may not contain all of the Hilbert region, forcing further subdivision. In many cases, the subdivision is justified, since points must be closer together to maintain a specified tolerance. In a few cases, especially with more liberal tolerances, the Hilbert region will still be partially outside the tool shadow but the tool *will* cut the region accurately enough. This happens when the cylindrical portion of the tool (above the ball portion) makes a small angle with the steep area of the object and hence cuts the (small) amount of Hilbert region outside the tool shadow to within tolerance. Nonetheless, the implementation performs further subdivision until the region lies entirely within the tool shadow.

The problem stems from the fixed direction used for creating pixel models. As seen in Figure 4.25, the tool may cut the surface accurately, but the machined region does not always lie strictly *beneath* the tool. A solution to the problem is to use the surface normal vector to determine the plane on which to build the pixel model. Of course, this means the orientation for the tool may be different relative to each contact point (even though the tool axis direction remains fixed). Hence, a minimum of two pixel models will be required at each contact point, plus an accurate surface normal.

In fact, three pixel models will be required. The tool must still be positioned using the 3-axis, highest point method; to *position* the tool using a variable orientation pixel model means that undercuts and similar features may lead to gouging of the work above the tool tip. Hence, only the *assessment* pixel models should vary in orientation.

Using the proposed method for tool positioning has another benefit: it will allow other tool shapes, in particular flat endmills, to be incorporated into the hybrid scheme. Flat endmills are an important consideration for practical systems and using the modified assessment strategy would allow the contact-oriented tool positioning method to remain unchanged.

Chapter 5

Conclusions and Future Work

The results of implementing the hybrid method suggest that the method does meet its intended goals: pixel model storage is reduced and several pixel model advantages are retained. It is unclear if load estimation or cut volume can be incorporated, but it seems possible that cleanup cuts could be added by adapting them to the Hilbert tree data structure. The implementation suffered because of the method used to find the surface normal vector and from an inadequate accuracy assessment method. Nonetheless, the region and profile paradigm, introduced to improve the edge handling properties of the hybrid method, showed promise as a way of partitioning the geometry to be machined and for modelling relations between the partitions.

The remainder of this chapter offers more thoughts on the method and suggests areas for future work.

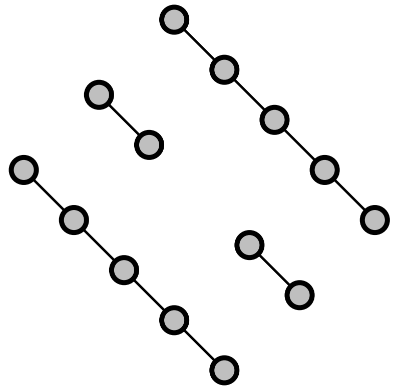


Figure 5.1: Problem with global point reordering. It is not possible to connect the passes in a zigzag topology. The best approximation to a zigzag path is also unclear.

5.1 Conclusions

An interesting outcome of the hybrid method is that it supports the creation of point-based roughing paths. This is not surprising given that the Hilbert sampling is a form of point-wise area fill and that the profile handling removes positions not inside the active region(s). It is not clear how such paths compare to traditional level-by-level roughing paths in terms of run time and cutting performance. (Certainly a semi-roughing operation would be necessary to smooth out the excess material before finish machining.)

For surface machining paths, the zigzag simplification step is not always adequate. Too much Hilbert character remains in the paths for many shapes. It seems reasonable that larger sections of the path could be reordered than at present. It also seems likely there would be a penalty in doing so as the zigzag passes would probably not connect directly to the unrefined sections,

In the most general case, the whole collection of points could be reordered.

An obvious way is to sort the points by x then y coordinates, create subpasses having constant x or y value, then link them together. Unfortunately, there is no guarantee that a zigzag tour of the points is possible. Consider the simple example in Figure 5.1. No direct zigzag tour is possible and it is unclear what constitutes a best approximation of a zigzag tour. A similar reordering issue arises in point-based roughing paths if there are missing points in the middle of the collection.

An advantage of the hybrid approach over Griffiths' method is that both coarse- and fine-grained parallelism exists. The coarse parallelism comes from the Hilbert subdivision. Independent sections could be handled by different processors since there are no data dependencies between sections. Fine-grained parallelism exists for the tool positioning and accuracy assessment algorithms; each processor could handle one pixel. Merging the sections of path is trivial as well: just retain the Hilbert ordering on the subpaths and use the Hilbert connectivity.

An unclear result is the cost of the change proposed for the accuracy assessment. Two or possibly three pixels images would be required for each tool position, and the images would be larger. How much larger each would be (or how big they *could* be) is not clear. A possible advantage of the change is that general tool orientations could be accommodated, as well as general shapes. This means 5-axis positioning may be possible.

A limitation of the point-wise accuracy assessment is that it is not adequate for surface machining paths as it leads to more points than are optimally required. This is true even of the proposed method. Consider the case of a flat endmill cutting a tilted surface, as shown in Figure 5.2. The machined region that is within

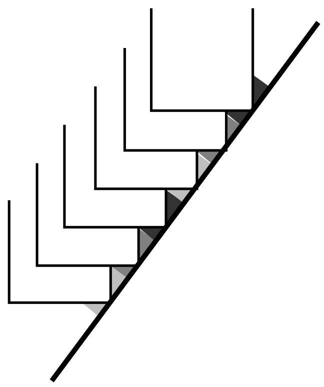


Figure 5.2: Problems with pointwise tool positioning. Many static positions are required to remove the same material as one tool movement.

tolerance is quite small and a number of positions are required to cover the surface in a pointwise manner. However, a sweep of the tool defined by just two points would remove the same material.

The extra points are a problem because tool positioning and assessment are relatively expensive operations. Invoking them more often than necessary is costly in terms of overall processing time. The extra points are also expensive in storage terms too, especially if the path was created off-line on a CAD/CAM system then moved to the machine tool later. If tool positioning were cheap, the method might be of use where the points are not stored, but sent directly to the machine tool.

To create paths with an optimal number of points, tool movements (not path points) must be used. This means that pairs of points must be used (as in traditional step along computations). The shape swept by the tool is to be compared to the surface and the maximum deviation used to decide if the proposed positions are acceptable.[1]

[1]The simulation and verification literature contains work related to computing and manipu-

In general, a tool movement must be positioned so the portion of the swept tool shape that machines the object within tolerance overlaps neighbouring movements. If this condition is satisfied, the object will be machined within tolerance. The hybrid method attempts to approximate that condition by ensuring that static tool positions cut a collection of square regions within tolerance. The problem is that there is much overlap of the in-tolerance portion of the static machined region (especially as implemented) and no account is made of the trough carved from position to position.

Using swept shapes for accuracy assessment is not compatible with the hybrid approach. Reordering of the points means that tool movements change compared to what was present at point generation time. Hilbert subdivision needs to know at point generation time whether a proposed position is acceptable. However, not all of the neighbouring points may exist. Even if all neighbours exist, searching for them (by scanning the Hilbert curve or traversing the Hilbert tree) is not always easy. Thus, the role of point-wise constructed paths seems best suited to point-based and level-by-level roughing path creation.

Concerning the role of pixel models in path generation, it seems they are well suited for supporting tool positioning and related operations but they are not a good choice for path planning. The need to position cutters tangent to boundaries and to determine which region, in a collection, a cutter is in makes the pure pixel model approach unattractive.

---

lating swept tool shapes.

5.2 Future Work

There are a number of possibilities for future work. The existing method could be implemented on a parallel computer to determine what speedup is possible. A limitation of doing so is that pixel grids may have to be computed without the use of graphics hardware (if synchronization between hardware and processes was difficult or impossible). If fine-grained parallelism was used, each processor could find one pixel height as well as compute one tool tip position.

The proposed tool positioning method could be investigated further to see if it would support 5-axis positioning. One concern would be that static tool positioning may not be adequate; the swept tool shape may be required to ensure the tool does not gouge the work in moving from position to position. An advantage of the hybrid approach is that tool positions will be close together, so the tool is not as likely to have drastic changes in orientation compared to more widely spaced locations.

Another extension of the hybrid scheme would be incorporating cleanup cut generation. The Hilbert regions having excess remaining could be marked by flags in the leaf nodes. A smaller tool would visit the region and move on a path created by further Hilbert refinement. Regions with excess left after cleanup would be flagged for the next smaller cutter in the sequence, and so on. The collection of disjoint subpaths for a given tool would have to be connected; a suggestion would be to use Griffiths' area skipping algorithm. Regions containing no cleanup paths would be skipped, and the subpaths would become a single path.

For point-based roughing, further work is necessary to even out the amount of material engaged at each position. Simple x and y sorting of path points, as proposed above for surface machining, would be one way. Finding the shortest path, however, would require a more sophisticated approach, especially if points were missing (due to forbidden regions).

A more general problem is that of determining the conditions under which a true zigzag tour is possible for the Hilbert path points. In most instances, this will not be possible. A related problem is therefore to find what approximations to a zigzag path are possible and what are the costs in terms of extra machine travel to pass through the points. A requirement would be that passes of the tool should be longer than they are in the Hilbert path (one movement) and ideally passes should be as long as possible, but still allow easy connection to adjacent passes.

Finally, further investigation into the profile and region paradigm is justified. It seems that region and profile abstractions might be sufficient to represent the elements implied in current machining practice, but that needs to be verified. The method must be tested on more complex examples. (Note that the Hilbert curve and pixel model path planning method are not necessary; other methods could be used.) Also, further study should be done to see what (other) types of regions there are and what operations they support. The results could be useful for designing a path generation programming language or engineering an object-oriented path creation framework.

Appendix A

Classified Bibliography of Literature on Path Generation

This appendix contains a classified bibliography of literature relating to tool path generation. The categories reflect approximately the clustering of topics that exist in the literature, with the exception of the categories *Point-based Roughing Paths, Cleanup Cut Tool Paths, Space-filling Curves,* and *Region Decomposition.* These I have specifically added since there is little material available in any of these areas.

The groupings are not exclusive; papers may appear in more than one category.

A.1 Surveys

[1] S. Marshall and J. G. Griffiths. A survey of cutter path construction techniques for milling machines. *International Journal of Production Research*, 32(12):2861–2877, 1994.

[2] J. Shah, P. Sreevalsan, and A. Mathew. Survey of CAD/feature-based process planning and NC programming techniques. *Computer-Aided Engineering Journal*, pages 25–33, February 1991.

[3] S. T. Tan, W. S. Sze, and W. Y. Wong. A survey and classification of cutter path generation for three-axis NC milling. *Advanced Manufacturing Engineering*, 2(2):103–13, April 1990.

A.2 Issues

There are two kinds of papers in this section, those dealing with specific concepts and those dealing with more general, philosophical issues.

[1] H. Araki, M. Kawai, Y. Amano, and M. Nonaka. Replacement of physical models by highly accurate numerical models for manufacturing stamping dies in the integrated CAD/CAM system. In J.P. Crestin and J.F. McWaters, editors, *Software for Discrete Manufacturing*, pages 47–60. IFIP, Elsevier, 1986.

[2] K. A. Awan and C. B. Besant. Design and manufacture of complex surfaces with 5-axis machining. In V.C. Venkatesh and J.A. McGeough, editors, *7th International Conference on Computer-Aided Production Engineering*, pages 449–59, Amsterdam, 1991. Elsevier.

[3] P. Broomhead and M. Edkins. Generating NC data at the machine tool for the manufacture of free-form surfaces. *International Journal of Production Research*, 24(1):1–14, 1986.

[4] B. K. Choi, Y. C. Chung, J. W. Park, and D. H. Kim. Unified CAM-system architecture for die and mould manufacturing. *Computer Aided Design*, 26(3):235–43, March 1994.

[5] Z. Dong, H. Li, and G. W. Vickers. Optimal rough machining of sculptured parts on a CNC milling machine. *Journal of Engineering for Industry*, 115(4):424–31, November 1993.

[6] M. Guyder. Automating the optimization of $2\frac{1}{2}$ axis milling. In *Proceedings, Computer Applications in Production & Engineering, CAPE '89.* North-Holland, October 1989.

[7] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[8] M. Held, G. Lukacs, and L. Andor. Pocket machining based on contour-parallel tool paths generated by means of proximity maps. *Computer Aided Design*, 26(3):189–203, March 1994.

[9] R. B. Jerard, J. M. Angleton, and R. L. Drysdale. Sculptured surface tool path generation with global interference checking. In *Design Productivity International Conference*, pages 737–742, February 1991.

[10] B. H. Kim and C. N. Chu. Effect of cutter mark on surface roughness and scallop height in sculptured surface machining. *Computer Aided Design*, 26(3):179–188, March 1994.

[11] K. Kim and J. E. Biegel. Path generation method for sculptured surface manufacture. *Computers & Industrial Engineering*, 14(2):95–101, 1988.

[12] R. Klass and P. Schramm. Numerically-controlled milling of CAD surface data. In H. Hagen and D. Roller, editors, *Geometric Modelling. Methods and Applications.*, pages 213–25, Germany, 1991. Springer-Verlag.

[13] D. Koncewicz. Numerical control programming system - NCS. *Prace Naukowe Instytutu Cybernetyki Technicznej, Politechniki Wroclawskiej*, 89(39):13–19, 1991.

[14] T. R. Kramer. Pocket milling with tool engagement detection. *Journal of Manufacturing Systems*, 11(2):114–23, 1992.

[15] H. Li, Z. Dong, and G. W. Vickers. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11):787–785, November 1994.

[16] S. X. Li and R. B. Jerard. 5-axis machining of sculptured surfaces with a flat-end cutter. *Computer Aided Design*, 26(3):165–178, March 1994.

[17] X. Li. *Automatic tool path generation for numerically controlled machining of sculptured surfaces.* PhD thesis, University of New Hampshire, 1993.

[18] K. Marciniak. Influence of surface shape on admissible tool positions in 5-axis face milling. *Computer Aided Design*, 19(5):233–36, June 1987.

[19] K. Marciniak. *Geometric Modelling for Numerically Controlled Machining.* Oxford Univ. Press, New York, 1991.

[20] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[21] S. Marshall and J. G. Griffiths. A survey of cutter path construction techniques for milling machines. *International Journal of Production Research*, 32(12):2861–2877, 1994.

[22] J. L. Murray, J. R. Corney, and J. Y. Alexander. Advancements in automatic machining of complex 2.5D components. In *Proceedings of the Institution of Mechanical Engineers. Effective CAD/CAM '91*, pages 59–65, Bury St. Edmonds, UK, November 1991. Institution of Mechanical Engineers, Mech. Eng. Publications.

[23] J. H. Oliver and E. D. Goodman. Direct dimensional NC verification. *Computer Aided Design*, 22(1):3–10, January-February 1990.

[24] P. V. Prabhu, A. K. Gramopadhye, and H. P. Wang. A general mathematical model for optimizing NC tool path for face milling of flat convex polygonal surfaces. *International Journal of Production Research*, 28(1):101–30, January 1990.

[25] K. Preiss. Automated mill pocketing computations. In F. L. Krause and H. Jansen, editors, *Advanced Geometric Modelling for Engineering Applications*, pages 261–273. Elsevir, Amsterdam, 1990.

[26] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–16, July 1991.

[27] J. Shah, P. Sreevalsan, and A. Mathew. Survey of CAD/feature-based process planning and NC programming techniques. *Computer-Aided Engineering Journal*, pages 25–33, February 1991.

[28] C. J. Su and A. Mukerjee. Automated machinability checking for CAD/CAM. *IEEE Transactions on Robotics and Automation*, 7(5):691–9, 1991.

[29] S. T. Tan, W. S. Sze, and W. Y. Wong. A survey and classification of cutter path generation for three-axis NC milling. *Advanced Manufacturing Engineering*, 2(2):103–13, April 1990.

[30] J. S. M. Vergeest, J. J. Broek, B. B. Schierbeek, and J. W. H. Tangelder. Rapid prototyping of complex shapes for conceptual design. *International Journal of Systems Automation: Research and Applications*, 1(3):305–24, 1991.

[31] G. Vickers and K. Quan. Ball-mills versus end-mills for curved surface machining. *Transactions of the ASME, Journal of Engineering for Industry*, 111(22):22–26, February 1989.

[32] H. Wang, H. Chang, R. A. Wysk, and A. Chandawarkar. On the efficiency of NC tool path planning for face milling operations. *Journal of Engineering for Industry*, 109:370–376, November 1987.

[33] H. P. Wang, H. Chang, and R. A. Wysk. An analytical approach to optimize NC tool path planning for face milling flat convex polygonal surfaces. *IIE Transactions*, 20(3):325–32, September 1988.

[34] W. P. Wang. Three-dimensional collision avoidance in production automation. *Computers in Industry*, 15(3):169–74, November 1990.

A.3 Systems

Papers in this section tend to describe entire path generation systems. While papers in other categories are narrower in focus, similar information may also be found in: *Non-iso-parametric Paths, Iso-parametric Paths, Sculptured Surface Pocketing Paths* and *Roughing Paths.*

[1] M. Bala and T. C. Chang. Automatic cutter selection and optimal cutter path generation for prismatic parts. *International Journal of Production Research*, 29(11):2163–76, November 1991.

[2] P. Broomhead and M. Edkins. Generating NC data at the machine tool for the manufacture of free-form surfaces. *International Journal of Production Research*, 24(1):1–14, 1986.

[3] G. Catania. A computer-aided prototype system for NC rough milling of free-form shaped mechanical part-pieces. *Computers in Industry*, 20(2):275–93, October 1992.

[4] Y. D. Chen, J. Ni, and S. M. Su. Real-time CNC tool path generation for machining IGES surfaces. *Journal of Engineering for Industry*, 115(4):480–6, November 1993.

[5] Y. D. Chen, J. Ni, and S. M. Wu. Real-time CNC tool path generation for machining IGES surfaces. *Sensors, Controls, and Quality Issues in Manufacturing*, PED v.55:187–99, 1991.

[6] B. K. Choi, Y. C. Chung, J. W. Park, and D. H. Kim. Unified CAM-system architecture for die and mould manufacturing. *Computer Aided Design*, 26(3):235–43, March 1994.

[7] W. Chungwatana, A. C. Lin, and W. F. Lu. Automatic generation of machining data for NURBS-based sculptured surfaces. In *Proceedings of the 1992 Japan - USA Symposium on Flexible Automation, Part 1*, pages 401–8, New York, 1992. ASME.

[8] A. Haapaniemi, H. Nagase, M. Fujimoto, H. Hiraoka, F. Kimura, and T. Sata. Development of a real time numerical controller for machining of sculptured surfaces. In *Software for Discrete Manufacturing, Proceedings of the 6th International IFIP/IFAC Conference*, pages 205–214, Amsterdam, 1986. North-Holland.

[9] G. Hermann. Algorithms for real-time toolpath generation. In M. J. Wozney, H. W. McLaughlin, and J. L. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 295–305. Elsevir, Amsterdam, 1988.

[10] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[11] T. Kuragano. Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free toolpaths. *Computer Aided Design*, 24(11):573–81, November 1992.

[12] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In *Proceedings of the 2nd USA-Japan Symposium on Flexible Automation*, pages 931–938, New York, July 1988. ASME.

[13] T. Kuragano, N. Sasaki, and A. Suzuki. The FRESDAM/M system for generating tool paths for manufacturing aesthetically pleasing injection molded products. In F. Kimura and A. Rolstadas, editors, *Computer Applications in Production and Engineering*, pages 411–418, Amsterdam, 1989. IFIP, Elsevier.

[14] Y-S Lee and T-C Chang. CASCAM—an automated system for sculptured surface cavity machining. *Computers in Industry*, 16:321–342, 1991.

[15] G. C. Loney and T. M. Ozsoy. NC machining of free form surfaces. *Computer Aided Design*, 19(2):85–90, March 1987.

[16] G. E. Yang and Y. S. Kong. Study on the 3-dimensional NC machining of the non-uniform B-spline composite surface. *Concurrent Engineering*, PED v.59:115–26, 1992.

A.4 Iso-parametric Paths

Works in this section develop paths in the parametric domain. The focus is mostly on surface machining.

[1] P. Broomhead and M. Edkins. Generating NC data at the machine tool for the manufacture of free-form surfaces. *International Journal of Production Research*, 24(1):1–14, 1986.

[2] G. Catania. A computer-aided prototype system for NC rough milling of free-form shaped mechanical part-pieces. *Computers in Industry*, 20(2):275–93, October 1992.

[3] J. J. Chou, E. Cohen, and S. H. Drake. Automatic sculptured five-axis milling with check surfaces. Technical Report UUCS-89-010, University of Utah, Salt Lake City, Utah, April 1989.

[4] J. J. Cox, Y. Takezaki, H. R. P. Ferguson, K. E. Kohkonen, and E. L. Mulkay. Space-filling curves in tool-path applications. *Computer Aided Design*, 26(3):215–24, March 1994.

[5] G. Elber. *Free form surface analysis using a hybrid of symbolic and numeric computation*. PhD thesis, University of Utah, 1992.

[6] G. Elber and E. Cohen. Tool path generation for freeform surface models. In *2nd ACM Solid Modeling*, pages 419–428, New York, May 1993. ACM.

[7] G. Elber and Cohen E. Toolpath generation for freeform surface models. *Computer Aided Design*, 26(6):490–496, June 1994.

[8] A. C. Lee, D. P. Chen, and C. L. Lin. CAD/CAM system from 3D coordinate measuring data. *International Journal of Production Research*, 28(12):2353–71, December 1990.

[9] G. C. Loney and T. M. Ozsoy. NC machining of free form surfaces. *Computer Aided Design*, 19(2):85–90, March 1987.

[10] M. G. Milanova, I. A. Aleksandrov, and R. K. Kunchev. Recovering and CAD/CAM processing of complex surfaces. In *ECAI' 92. 10 European Conference on Artificial Intelligence Proceedings*, pages 824–5. Wiley, UK, August 1992.

[11] G. E. Yang and Y. S. Kong. Study on the 3-dimensional NC machining of the non-uniform B-spline composite surface. *Concurrent Engineering*, PED v.59:115–26, 1992.

[12] C. Zhu. Tool-path generation in manufacturing sculptured surfaces with a cylindrical end-milling cutter. *Computers in Industry*, 17:385–389, 1991.

A.5 Non-iso-parametric Paths

The papers in this section develop paths in Cartesian space. Some systems use plane-surface intersection curves to build the path, others use planar curves and selected points on them.

[1] J. E. Bobrow. NC machine tool path generation from CSG part representations. *Computer Aided Design*, 17(2):69–76, March 1985.

[2] Y. D. Chen, J. Ni, and S. M. Su. Real-time CNC tool path generation for machining IGES surfaces. *Journal of Engineering for Industry*, 115(4):480–6, November 1993.

[3] Y. D. Chen, J. Ni, and S. M. Wu. Real-time CNC tool path generation for machining IGES surfaces. *Sensors, Controls, and Quality Issues in Manufacturing*, PED v.55:187–99, 1991.

[4] B. K. Choi, C. S. Lee, J. S. Hwang, and C. S. Jun. Compound surface modelling and machining. *Computer Aided Design*, 20(3):127–136, April 1988.

[5] J. J. Chou. *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces.* PhD thesis, University of Utah, 1989.

[6] J. J. Chou and E. Cohen. Constant scallop height tool path generation. Technical Report UUCS-89-011, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[7] A. Haapaniemi, H. Nagase, M. Fujimoto, H. Hiraoka, F. Kimura, and T. Sata. Development of a real time numerical controller for machining of sculptured surfaces. In *Software for Discrete Manufacturing, Proceedings of the 6th International IFIP/IFAC Conference*, pages 205–214, Amsterdam, 1986. North-Holland.

[8] G. Hermann. Algorithms for real-time toolpath generation. In M. J. Wozney, H. W. McLaughlin, and J. L. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 295–305. Elsevir, Amsterdam, 1988.

[9] Y. Huang and J. H. Oliver. Non-constant parameter NC tool path generation on sculptured surfaces. *International Journal of Advanced Manufacturing Technology*, 9:281–290, 1994.

[10] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[11] R. B. Jerard, J. M. Angleton, and R. L. Drysdale. Sculptured surface tool path generation with global interference checking. In *Design Productivity International Conference*, pages 737–742, February 1991.

[12] T. Kuragano. Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free toolpaths. *Computer Aided Design*, 24(11):573–81, November 1992.

[13] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In *Proceedings of the 2nd USA-Japan Symposium on Flexible Automation*, pages 931–938, New York, July 1988. ASME.

[14] T. Kuragano, N. Sasaki, and A. Suzuki. The FRESDAM/M system for generating tool paths for manufacturing aesthetically pleasing injection molded

products. In F. Kimura and A. Rolstadas, editors, *Computer Applications in Production and Engineering*, pages 411–418, Amsterdam, 1989. IFIP, Elsevier.

[15] J-Y Lai and D-J Wang. A strategy for finish cutting path generation of compound surfaces. *Computers in Industry*, 25:189–209, 1994.

[16] X. Li. *Automatic tool path generation for numerically controlled machining of sculptured surfaces*. PhD thesis, University of New Hampshire, 1993.

[17] R. C. Luo and Y. Ma. Free form surface representation and machining for complex parts. In *Proceedings, 1994 IEEE Conference on Robotics and Automation*, pages 2897–2902, New York, 1994. IEEE.

[18] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[19] T. Sakuta, M. Kawai, and Y. Amano. Development of an NC machining system for stamping dies by offset surface method. In *Proceedings, Autofact '87*, pages 2–13–2–27. American Society of Manufacturing Engineers, November 1987.

A.6 Sculptured Surface Pocketing Paths

The approaches in this section focus on the case of pockets defined by sculptured surfaces. See also: *Planar Pocketing Paths,* and *Roughing Paths.*

[1] Y. Chen and B. Ravani. Offset surface generation and contouring in computer-aided design. *Journal of Mechanisms, Transmissions, and Automation in Design*, 109(1):133–42, March 1987.

[2] Y-S Lee and T-C Chang. CASCAM—an automated system for sculptured surface cavity machining. *Computers in Industry*, 16:321–342, 1991.

[3] Y. S. Lee and T. C. Chang. A contour growth method from intersections of hunting-planes to evaluate machining information. *Journal of Design and Manufacturing*, 2(3):119–33, September 1992.

[4] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[5] Y. S. Suh and K. Lee. NC milling tool path generation for arbitrary pockets defined by sculptured surfaces. *Computer Aided Design*, 22(5):273–84, June 1990.

A.7 Planar Pocketing Paths

This section concerns 2D pocketing paths, both contour-parallel and zigzag. See also: *Sculptured Surface Pocketing Paths* and *Roughing Paths.*

[1] M. Bala and T. C. Chang. Automatic cutter selection and optimal cutter path generation for prismatic parts. *International Journal of Production Research*, 29(11):2163–76, November 1991.

[2] J. J. Chou. *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces.* PhD thesis, University of Utah, 1989.

[3] J. J. Chou and E. Cohen. Computing offsets and tool paths with voronoi diagrams. Technical Report UUCS-89-017, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[4] Y. S. Chou, T. C. Woo, L. L. Chen, K. Tang, and S. Y. Shin. Scallop hull and its offset. *Computer Aided Design*, 26(7):537–542, July 1994.

[5] W. Chungwatana, A. C. Lin, and W. F. Lu. Automatic generation of machining data for NURBS-based sculptured surfaces. In *Proceedings of the 1992 Japan - USA Symposium on Flexible Automation, Part 1*, pages 401–8, New York, 1992. ASME.

[6] I. G. Gosling. Tool-path algorithm exhibiting improved collision behaviour. *Computer-Aided Engineering Journal*, 7(5):135–40, October 1990.

[7] I. G. Gosling. Generation of high speed tool paths with collision avoidance. In B.S. Lim, editor, *Proceedings of the International Conference on Computer Integrated Manufacturing, ICCIM '91. Manufacturing Enterprises of the 21st Century.*, pages 245–8, Singapore, 1991. World Scientific.

[8] M. Guyder. Automating the optimization of $2\frac{1}{2}$ axis milling. In *Proceedings, Computer Applications in Production & Engineering, CAPE '89.* North-Holland, October 1989.

[9] A. Hansen and F. Arbab. An algorithm for generating NC tool paths for arbitrarily shaped pockets with islands. *ACM Transactions on Graphics*, 11(2):152–82, April 1992.

[10] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[11] M. Held. Computational geometry for pocket machining. In S.M. Slaby and H. Stachel, editors, *Proceedings of the Third International Conference on Engineering Graphics and Descriptive Geometry*, pages 224–31 vol.1, Vienna, 1988. Tech. Univ. Vienna.

[12] M. Held. Geopocket-a sophisticated computational geometry solution of geometrical and technological problems arising from pocket machining. In F. Kimura and A. Rolstadas, editors, *Computer Applications in Production and Engineering. Proceedings of the Third International IFIP Conference. CAPE '89*, pages 283–93, Amsterdam, 1989. North-Holland.

[13] M. Held. ZigPocket: on the computational geometry of zigzag pocket machining. In T.S. Chua and T.L. Kunii, editors, *CG International '90. Computer Graphics Around the World.*, pages 281–96, Berlin, 1990. Springer-Verlag.

[14] M. Held. A geometry-based investigation of the tool path generation for zigzag pocket machining. *Visual Computer*, 7(5-6):296–308, 1991.

[15] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1991.

[16] M. Held, G. Lukacs, and L. Andor. Pocket machining based on contour-parallel tool paths generated by means of proximity maps. *Computer Aided Design*, 26(3):189–203, March 1994.

[17] W. Ho, J. H. Davis, and W. A. Adcock. Milling machining automation scheme for B-REP-2.5D solid objects. *Advances in Design Automation. 18th Annual ASME Design Automation Conference*, DE v.44 pt.1:151–154, 1992.

[18] A. Kaul. Minkowski sums. A simulation tool for CAD/CAM. In *Computers in Engineering - 1992. Proceedings of the 1992 ASME International Computers in Engineering and Exposition*, pages 447–456, New York, 1992. ASME.

[19] K. Kim and J. Jeong. Tool path generation for machining free-form pockets with islands. *Computers & Industrial Engineering*, 28(2):399–407, 1995.

[20] T. R. Kramer. Pocket milling with tool engagement detection. *Journal of Manufacturing Systems*, 11(2):114–23, 1992.

[21] Y-S Lee and T-C Chang. Application of computational geometry in optimizing 2.5D and 3D NC surface machining. *Computers in Industry*, 26:41–59, 1995.

[22] H. Li, Z. Dong, and G. W. Vickers. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11):787–785, November 1994.

[23] H. Persson. NC machining of arbitrarily shaped pockets. *Computer Aided Design*, 10(3):169–174, May 1978.

[24] K. Preiss. Automated mill pocketing computations. In F. L. Krause and H. Jansen, editors, *Advanced Geometric Modelling for Engineering Applications*, pages 261–273. Elsevir, Amsterdam, 1990.

A.8 Roughing Paths

Work in this section concerns more general aspects of level by level roughing.

[1] P. Broomhead and M. Edkins. Generating NC data at the machine tool for the manufacture of free-form surfaces. *International Journal of Production Research*, 24(1):1–14, 1986.

[2] G. Catania. A computer-aided prototype system for NC rough milling of free-form shaped mechanical part-pieces. *Computers in Industry*, 20(2):275–93, October 1992.

[3] W. Chungwatana, A. C. Lin, and W. F. Lu. Automatic generation of machining data for NURBS-based sculptured surfaces. In *Proceedings of the 1992 Japan - USA Symposium on Flexible Automation, Part 1*, pages 401–8, New York, 1992. ASME.

[4] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[5] W. Ho, J. H. Davis, and W. A. Adcock. Milling machining automation scheme for B-REP-2.5D solid objects. *Advances in Design Automation. 18th Annual ASME Design Automation Conference*, DE v.44 pt.1:151–154, 1992.

[6] T. Kuragano. Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free toolpaths. *Computer Aided Design*, 24(11):573–81, November 1992.

[7] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In *Proceedings of the 2nd USA-Japan Symposium on Flexible Automation*, pages 931–938, New York, July 1988. ASME.

[8] T. Kuragano, N. Sasaki, and A. Suzuki. The FRESDAM/M system for generating tool paths for manufacturing aesthetically pleasing injection molded products. In F. Kimura and A. Rolstadas, editors, *Computer Applications in Production and Engineering*, pages 411–418, Amsterdam, 1989. IFIP, Elsevier.

[9] K. Lee, T. J. Kim, and S. E. Hong. Generation of toolpath with selection of proper tools for rough cutting process. *Computer Aided Design*, 26(11):822–831, November 1994.

[10] Y-S Lee and T-C Chang. CASCAM—an automated system for sculptured surface cavity machining. *Computers in Industry*, 16:321–342, 1991.

[11] H. Li, Z. Dong, and G. W. Vickers. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11):787–785, November 1994.

[12] G. C. Loney and T. M. Ozsoy. NC machining of free form surfaces. *Computer Aided Design*, 19(2):85–90, March 1987.

[13] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[14] S. T. Tan, M. F. Yuen, W. S. Sze, and W. Y. Wong. NC machining algorithms for CSG solids. In *Advances in Design Automation - 1987. Volume One: Design Methods, Computer Graphics and Expert Systems*, pages 355–63, New York, 1987. IEEE.

[15] S. T. Tan, M. F. Yuen, W. S. Sze, and W. Y. Wong. An NC machining package for solid objects. In *Fourth International Conference on Computer-Aided Production Engineering*, pages 81–90, Bury St. Edmonds, UK, 1988. Mech. Eng. Publishers.

[16] M. F. Yuen, S. T. Tan, W. S. Sze, and W. Y. Wong. Octree approach to rough machining. *Management and Engineering Manufacture. Proceedings of the Institution of Mechanical Engineers, B.*, 201(B3):157–63, 1987.

A.9 Point-based Roughing Paths

The paper by Kuragano is the only substantial reference on point-based roughing. The others mention it, but do not go into detail.

[1] J. P. Duncan and K. K. Law. *Computer-aided Sculpture.* Cambridge University Press, New York, 1989.

[2] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In *Proceedings of the 2nd USA-Japan Symposium on Flexible Automation*, pages 931–938, New York, July 1988. ASME.

[3] Y. Takeuchi, M. Sakamoto, Y. Abe, and R. Orita. Development of a personal CAD/CAM system for mold manufacture based on solid modeling techniques. *Annals of the CIRP*, 38(1):429–432, 1989.

A.10 Cleanup Cut Tool Paths

Papers in this section mention the construction of paths that remove only excess material in specific regions of the model. Li's thesis goes into the most detail; others remain more conceptual.

[1] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer Aided Design*, 21(6):371–8, July-August 1989.

[2] J. P. Duncan and K. K. Law. *Computer-aided Sculpture*. Cambridge University Press, New York, 1989.

[3] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[4] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[5] K. Lee, T. J. Kim, and S. E. Hong. Generation of toolpath with selection of proper tools for rough cutting process. *Computer Aided Design*, 26(11):822–831, November 1994.

[6] X. Li. *Automatic tool path generation for numerically controlled machining of sculptured surfaces*. PhD thesis, University of New Hampshire, 1993.

[7] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–16, July 1991.

[8] S. W. Thomas. Scanline rendering for 3-axis NC toolpath generation, simulation, and verification. Technical Report CSE-TR-43-90, University of Michigan, Ann Arbor, Michigan, January 1990.

A.11 Tool Positioning

The following contain more detailed treatments of tool positioning. See also *Offset Surface Method,* and *Systems*. Papers on surface machining generally mention tool positioning, though not necessarily in detail.

[1] I. Ali. Parallel algorithms to generate interference-free tool paths for manufacturing B-spline surfaces. Master's thesis, University of Waterloo, 1990.

[2] I. Ali and S. Bedi. NC controller for surface machining. *International Journal of Advanced Manufacturing*, 7(3):150–8, 1992.

[3] H. D. Cho, Y. T. Jun, and M. Y. Yang. Five-axis CNC milling for effective machining of sculptured surfaces. *International Journal of Production Research*, 31(11):2559–73, November 1993.

[4] B. K. Choi, C. S. Lee, J. S. Hwang, and C. S. Jun. Compound surface modelling and machining. *Computer Aided Design*, 20(3):127–136, April 1988.

[5] A. Hansen and F. Arbab. Fixed-axis tool positioning with built-in global interference checking for NC path generation. *IEEE Journal of Robotics and Automation*, 4(6):610–21, December 1988.

[6] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[7] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[8] T. Kanda. A cutter path determined by a given numerical model in NC mold processing. In *Proceedings IECON '91. 1991 International Conference on Industrial Electronics, Control and Instrumentation*, pages 1221–5, New York, 1991. IEEE.

A.12 Offset Surface Method

The papers here use offset surfaces or offset surface approximations. Approximations may take various forms, including meshes, pixels, or CSG primitives.

[1] E. L. Brechner. *Envelopes and tool paths for three-axis end milling.* PhD thesis, Rensselaer Polytechnic Institute, 1990.

[2] E. L. Brechner. General tool offset curves and surfaces. *Advances in Design Automation*, DE v.23 pt.1:273–82, 1990.

[3] Y. Chen and B. Ravani. Offset surface generation and contouring in computer-aided design. *Journal of Mechanisms, Transmissions, and Automation in Design*, 109(1):133–42, March 1987.

[4] Y. D. Chen, J. Ni, and S. M. Su. Real-time CNC tool path generation for machining IGES surfaces. *Journal of Engineering for Industry*, 115(4):480–6, November 1993.

[5] Y. D. Chen, J. Ni, and S. M. Wu. Real-time CNC tool path generation for machining IGES surfaces. *Sensors, Controls, and Quality Issues in Manufacturing*, PED v.55:187–99, 1991.

[6] B. K. Choi, Y. C. Chung, J. W. Park, and D. H. Kim. Unified CAM-system architecture for die and mould manufacturing. *Computer Aided Design*, 26(3):235–43, March 1994.

[7] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[8] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[9] A. Kaul. Minkowski sums. A simulation tool for CAD/CAM. In *Computers in Engineering - 1992. Proceedings of the 1992 ASME International Computers in Engineering and Exposition*, pages 447–456, New York, 1992. ASME.

[10] T. Kishinami, T. Kondo, and K. Saito. Inverse offset method for cutter path generation. In *Proceedings, 6th International Conference on Production Engineering*, pages 807–812, Osaka, Japan, November 1987.

[11] J-Y Lai and D-J Wang. A strategy for finish cutting path generation of compound surfaces. *Computers in Industry*, 25:189–209, 1994.

[12] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[13] T. Sakuta, M. Kawai, and Y. Amano. Development of an NC machining system for stamping dies by offset surface method. In *Proceedings, Autofact '87*, pages 2–13–2–27. American Society of Manufacturing Engineers, November 1987.

[14] H. Suzuki, Y. Kuroda, M. Sakamoto, S. Haramaki, and H. Van Brussel. Development of the CAD/CAM system based on parallel processing and the inverse offset method. In P. Welch, D. Siles, T.L. Kunii, and A. Bakkers, editors, *Transputing '91. Proceedings of the World Transputer User Group (WOTUG) Conference*, pages 184–98, Amsterdam, 1991. IOS.

[15] Y. Takeuchi, M. Sakamoto, Y. Abe, and R. Orita. Development of a personal CAD/CAM system for mold manufacture based on solid modeling techniques. *Annals of the CIRP*, 38(1):429–432, 1989.

A.13 Mesh Models

The papers below deal with triangular and quad meshes as a basis for path generation.

[1] I. Ali. Parallel algorithms to generate interference-free tool paths for manufacturing B-spline surfaces. Master's thesis, University of Waterloo, 1990.

[2] I. Ali and S. Bedi. A multiprocessor-based surface machining system with interference checking. In *Second International Conference on Advanced Manufacturing Systems and Technologies*, June 1990.

[3] K. A. Awan and C. B. Besant. Design and manufacture of complex surfaces with 5-axis machining. In V.C. Venkatesh and J.A. McGeough, editors, *7th International Conference on Computer-Aided Production Engineering*, pages 449–59, Amsterdam, 1991. Elsevier.

[4] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer Aided Design*, 21(6):371–8, July-August 1989.

[5] J. J. Chou. *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces.* PhD thesis, University of Utah, 1989.

[6] J. J. Chou and E. Cohen. Constant scallop height tool path generation. Technical Report UUCS-89-011, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[7] J. P. Duncan and K. K. Law. *Computer-aided Sculpture.* Cambridge University Press, New York, 1989.

[8] A. Hansen and F. Arbab. Fixed-axis tool positioning with built-in global interference checking for NC path generation. *IEEE Journal of Robotics and Automation*, 4(6):610–21, December 1988.

[9] A. B. Hansen. *Tool positioning and path generation algorithms for computer-aided manufacturing.* PhD thesis, University of Southern California, 1989.

[10] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[11] T. Kanda. A cutter path determined by a given numerical model in NC mold processing. In *Proceedings IECON '91. 1991 International Conference on Industrial Electronics, Control and Instrumentation*, pages 1221–5, New York, 1991. IEEE.

[12] A. Kaul. Minkowski sums. A simulation tool for CAD/CAM. In *Computers in Engineering - 1992. Proceedings of the 1992 ASME International Computers in Engineering and Exposition*, pages 447–456, New York, 1992. ASME.

[13] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[14] T. Kuragano. Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free toolpaths. *Computer Aided Design*, 24(11):573–81, November 1992.

[15] T. Kuragano, N. Sasaki, and A. Kikuchi. The FRESDAM system for designing and manufacturing freeform objects. In *Proceedings of the 2nd USA-Japan Symposium on Flexible Automation*, pages 931–938, New York, July 1988. ASME.

[16] T. Kuragano, N. Sasaki, and A. Suzuki. The FRESDAM/M system for generating tool paths for manufacturing aesthetically pleasing injection molded products. In F. Kimura and A. Rolstadas, editors, *Computer Applications in Production and Engineering*, pages 411–418, Amsterdam, 1989. IFIP, Elsevier.

[17] J-Y Lai and D-J Wang. A strategy for finish cutting path generation of compound surfaces. *Computers in Industry*, 25:189–209, 1994.

[18] Y-S Lee and T-C Chang. CASCAM—an automated system for sculptured surface cavity machining. *Computers in Industry*, 16:321–342, 1991.

[19] S. X. Li and R. B. Jerard. 5-axis machining of sculptured surfaces with a flat-end cutter. *Computer Aided Design*, 26(3):165–178, March 1994.

[20] X. Li. *Automatic tool path generation for numerically controlled machining of sculptured surfaces*. PhD thesis, University of New Hampshire, 1993.

[21] T. Sakuta, M. Kawai, and Y. Amano. Development of an NC machining system for stamping dies by offset surface method. In *Proceedings, Autofact '87*, pages 2–13–2–27. American Society of Manufacturing Engineers, November 1987.

[22] H. Suzuki, Y. Kuroda, M. Sakamoto, S. Haramaki, and H. Van Brussel. Development of the CAD/CAM system based on parallel processing and the inverse offset method. In P. Welch, D. Siles, T.L. Kunii, and A. Bakkers, editors, *Transputing '91. Proceedings of the World Transputer User Group (WOTUG) Conference*, pages 184–98, Amsterdam, 1991. IOS.

A.14 Pixel and Point Models

The papers in this section deal with pixel or image data as well as non-uniform point sample data sets.

[1] B. K. Choi, Y. C. Chung, J. W. Park, and D. H. Kim. Unified CAM-system architecture for die and mould manufacturing. *Computer Aided Design*, 26(3):235–43, March 1994.

[2] J. P. Duncan and K. K. Law. *Computer-aided Sculpture.* Cambridge University Press, New York, 1989.

[3] R. B. Jerard, J. M. Angleton, and R. L. Drysdale. Sculptured surface tool path generation with global interference checking. In *Design Productivity International Conference*, pages 737–742, February 1991.

[4] C. B. Kim, M. Y. Yang, and S. Park. Tool path verification and NC program editing under a multi-window environment. In *Proceedings of the 1992 Japan - USA Symposium on Flexible Automation, Part 1*, pages 409–14, New York, 1992. ASME.

[5] T. Kishinami, T. Kondo, and K. Saito. Inverse offset method for cutter path generation. In *Proceedings, 6th International Conference on Production Engineering*, pages 807–812, Osaka, Japan, November 1987.

[6] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[7] S. Park, M. Y. Yang, and Lee C. W. Simulation of NC machining using a ball end mill. *Computer Modeling and Simulation of Manufacturing Processes*, MD v.20:67–76, 1990.

[8] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–16, July 1991.

[9] H. Suzuki, Y. Kuroda, M. Sakamoto, S. Haramaki, and H. Van Brussel. Development of the CAD/CAM system based on parallel processing and the inverse offset method. In P. Welch, D. Siles, T.L. Kunii, and A. Bakkers, editors, *Transputing '91. Proceedings of the World Transputer User Group (WOTUG) Conference*, pages 184–98, Amsterdam, 1991. IOS.

[10] Y. Takeuchi, M. Sakamoto, Y. Abe, and R. Orita. Development of a personal CAD/CAM system for mold manufacture based on solid modeling techniques. *Annals of the CIRP*, 38(1):429–432, 1989.

[11] S. W. Thomas. Scanline rendering for 3-axis NC toolpath generation, simulation, and verification. Technical Report CSE-TR-43-90, University of Michigan, Ann Arbor, Michigan, January 1990.

[12] A. M. Vepsäläinen. An application of morphological filters to NC-programming. *Proceedings of the SPIE - The International Society for Optical Engineering*, 1350:177–83, 1990.

[13] J. S. M. Vergeest, J. J. Broek, B. B. Schierbeek, and J. W. H. Tangelder. Rapid prototyping of complex shapes for conceptual design. *International Journal of Systems Automation: Research and Applications*, 1(3):305–24, 1991.

[14] D. Zhang and A. Bowyer. CSG set-theoretic solid modelling and NC machining of blend surfaces. In *Proceedings, 2nd Annual ACM Conference on Computational Geometry*, pages 236–245, Yorktown Heights, New York, June 1986. IBM.

A.15 Space-filling Curve-based Tool Paths

There are only two primary references on the use of space-filling curves in tool path generation. Marshall and Griffiths provide additional comments on Griffiths' method in their survey paper.

[1] J. J. Cox, Y. Takezaki, H. R. P. Ferguson, K. E. Kohkonen, and E. L. Mulkay. Space-filling curves in tool-path applications. *Computer Aided Design*, 26(3):215–24, March 1994.

[2] J. G. Griffiths. Toolpath based on Hilbert's curve. *Computer Aided Design*, 26(11):839–844, November 1994.

[3] S. Marshall and J. G. Griffiths. A survey of cutter path construction techniques for milling machines. *International Journal of Production Research*, 32(12):2861–2877, 1994.

A.16 Region Decomposition

Papers in this section decompose the geometry using a number of different techniques and criteria, including Voronoi diagrams, CSG primitives, and feature-based geometry.

[1] M. Bala and T. C. Chang. Automatic cutter selection and optimal cutter path generation for prismatic parts. *International Journal of Production Research*, 29(11):2163–76, November 1991.

[2] J. E. Bobrow. NC machine tool path generation from CSG part representations. *Computer Aided Design*, 17(2):69–76, March 1985.

[3] B. K. Choi, C. S. Lee, J. S. Hwang, and C. S. Jun. Compound surface modelling and machining. *Computer Aided Design*, 20(3):127–136, April 1988.

[4] J. J. Chou. *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces.* PhD thesis, University of Utah, 1989.

[5] J. J. Chou and E. Cohen. Computing offsets and tool paths with voronoi diagrams. Technical Report UUCS-89-017, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[6] J. J. Chou and E. Cohen. Constant scallop height tool path generation. Technical Report UUCS-89-011, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[7] J. J. Chou, E. Cohen, and S. H. Drake. Automatic sculptured five-axis milling with check surfaces. Technical Report UUCS-89-010, University of Utah, Salt Lake City, Utah, April 1989.

[8] G. Elber. *Free form surface analysis using a hybrid of symbolic and numeric computation.* PhD thesis, University of Utah, 1992.

[9] G. Elber and Cohen E. Toolpath generation for freeform surface models. *Computer Aided Design*, 26(6):490–496, June 1994.

[10] M. C. Fields and D. C. Anderson. Fast feature extraction for machining applications. *Computer Aided Design*, 26(11):803–813, November 1994.

[11] J. G. Griffiths. Toolpath based on Hilbert's curve. *Computer Aided Design*, 26(11):839–844, November 1994.

[12] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1991.

[13] M. Held, G. Lukacs, and L. Andor. Pocket machining based on contour-parallel tool paths generated by means of proximity maps. *Computer Aided Design*, 26(3):189–203, March 1994.

[14] W. Ho, J. H. Davis, and W. A. Adcock. Milling machining automation scheme for B-REP-2.5D solid objects. *Advances in Design Automation. 18th Annual ASME Design Automation Conference*, DE v.44 pt.1:151–154, 1992.

[15] C. G. Jensen and D. C. Anderson. Accurate tool placement and orientation for finish surface machining. *Concurrent Engineering*, PED v.59:127–45, 1992.

[16] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[17] T. R. Kramer. Pocket milling with tool engagement detection. *Journal of Manufacturing Systems*, 11(2):114–23, 1992.

[18] A. C. Lee, D. P. Chen, and C. L. Lin. CAD/CAM system from 3D coordinate measuring data. *International Journal of Production Research*, 28(12):2353–71, December 1990.

[19] Y-S Lee and T-C Chang. Application of computational geometry in optimizing 2.5D and 3D NC surface machining. *Computers in Industry*, 26:41–59, 1995.

[20] H. Li, Z. Dong, and G. W. Vickers. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11):787–785, November 1994.

[21] Y. B. Li and S. K. Ghosh. NC programming and machining of sculptured surfaces. *Journal of Materials Processing Technology*, 24:105–14, December 1990.

[22] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[23] M. Matsuda and F. Kimura. Extraction of machining features for milling data generation. In G. Doumeingts, J. Browne, and M. Tomljanovich, editors, *Computer Applications in Production and Engineering. Proceedings of the Fourth International IFIP TC5 Conference on Computer Applications in Production and Engineering, CAPE '91.*, pages 353–60, Amsterdam, 1991. North-Holland.

[24] H. Persson. NC machining of arbitrarily shaped pockets. *Computer Aided Design*, 10(3):169–174, May 1978.

[25] M. Shpitalni and A. Fischer. Separation of disconnected machining regions on the basis of a CSG model. *Computer Aided Design*, 24(1):46–58, January 1994.

[26] Y. J. Tseng and S. B. Joshi. Recognizing multiple interpretations in $2frac$12D machining of pockets. *International Journal of Production Research*, 32(5):1063–1086, 1994.

[27] W. P. Wang. Three-dimensional collision avoidance in production automation. *Computers in Industry*, 15(3):169–74, November 1990.

[28] D. Y. Yu, Z. C. Duan, H. D. Sun, P. G. Li, H. W. Ha, and X. P. Li. Automatic generation of the cutting-tool path for free-form surfaces. *Journal of Materials Processing Technology*, 36(4):415–25, April 1993.

A.17 Five Axis Methods

The works in this section pertain to various aspects of 5-axis path creation.

[1] K. A. Awan and C. B. Besant. Design and manufacture of complex surfaces with 5-axis machining. In V.C. Venkatesh and J.A. McGeough, editors, *7th International Conference on Computer-Aided Production Engineering*, pages 449–59, Amsterdam, 1991. Elsevier.

[2] B. K. Choi, J. W. Park, and C. S. Jun. Cutter-location data optimization in 5-axis surface machining. *Computer Aided Design*, 25(6):377–386, June 1993.

[3] J. J. Chou. *Numerical control milling machine toolpath generation for regions bounded by free form curves and surfaces.* PhD thesis, University of Utah, 1989.

[4] J. J. Chou, E. Cohen, and S. H. Drake. Automatic sculptured five-axis milling with check surfaces. Technical Report UUCS-89-010, University of Utah, Salt Lake City, Utah, April 1989.

[5] G. Elber. Accessibility in 5-axis milling environment. *Computer Aided Design*, 26(11):796–802, November 1994.

[6] C. G. Jensen and D. C. Anderson. Accurate tool placement and orientation for finish surface machining. *Concurrent Engineering*, PED v.59:127–45, 1992.

[7] H Ko, M-S Kim, H-G Park, and S-W Kim. Face sculpturing robot with recognition capability. *Computer Aided Design*, 26(11):814–821, November 1994.

[8] S. X. Li and R. B. Jerard. 5-axis machining of sculptured surfaces with a flat-end cutter. *Computer Aided Design*, 26(3):165–178, March 1994.

[9] X. Li. *Automatic tool path generation for numerically controlled machining of sculptured surfaces*. PhD thesis, University of New Hampshire, 1993.

[10] G. Vickers and K. Quan. Ball-mills versus end-mills for curved surface machining. *Transactions of the ASME, Journal of Engineering for Industry*, 111(22):22–26, February 1989.

[11] W. P. Wang. Three-dimensional collision avoidance in production automation. *Computers in Industry*, 15(3):169–74, November 1990.

[12] D. Y. Yu, Z. C. Duan, H. D. Sun, P. G. Li, H. W. Ha, and X. P. Li. Automatic generation of the cutting-tool path for free-form surfaces. *Journal of Materials Processing Technology*, 36(4):415–25, April 1993.

A.18 Simulation and Verification

This section gives a sampling of some of the simulation and verification literature. Considerably more is available, as the field contains a large body of published material.

[1] Y. Huang and J. H. Oliver. NC milling error assessment and tool path correction. In *Computer Graphics, Proceedings SIGGRAPH '94*, pages 287–294, New York, July 1994. ACM, ACM.

[2] R. B. Jerard and R. L. Drysdale. Methods for geometric modeling, simulation and spatial verification of NC machining programs. In J. Turner, J. Pegna, and M. Wozny, editors, *Product Modeling for Computer-Aided Design and Manufacture. Selected and Expanded Papers from the IFIP TC5/WG5.2 Working Conference on Geometric Modeling for Product Engineering*, pages 39–52, Amsterdam, 1991. North-Holland.

[3] C. B. Kim, M. Y. Yang, and S. Park. Tool path verification and NC program editing under a multi-window environment. In *Proceedings of the 1992 Japan - USA Symposium on Flexible Automation, Part 1*, pages 409–14, New York, 1992. ASME.

[4] H. Mayr and S. Stifter. Off-line generation of error-free robotic/NC code using simulation and automatic programming techniques. In G. Halevi, R. Weill, and I. Yudilevich, editors, *Robotic Systems and AMT. Proceedings of the IFIP TC5/WG 5.3 International Conference*, pages 127–36, Amsterdam, 1989. North-Holland.

[5] J. P. Menon and H. B. Voelcker. Toward a comprehensive formulation of NC verification as a mathematical and computational problem. *Concurrent Engineering*, PED v.59:147–64, 1992.

[6] J. H. Oliver and E. D. Goodman. Direct dimensional NC verification. *Computer Aided Design*, 22(1):3–10, January-February 1990.

[7] S. Park, M. Y. Yang, and Lee C. W. Simulation of NC machining using a ball end mill. *Computer Modeling and Simulation of Manufacturing Processes*, MD v.20:67–76, 1990.

[8] H. Ristock. Distributed selective simulation in NC manufacturing. In D. Richter and H. Grabowski, editors, *Engineering Information in Data Bases and Knowledge Based Systems. Techno-Data '90. Proceedings of the International Conference Techno-Data '90.*, pages 350–365, Berlin, 1990. Akademie-Verlag.

[9] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–16, July 1991.

[10] X. Sheng and M. Blunck. Parallel NC simulation based on a discrete solid model. In T.S. Durrani and et. al., editors, *Applications of Transputers 3. Proceedings of the Third International Conference on Applications of Transputers.*, pages 798–803, Amsterdam, 1991. IOS.

[11] P. Su and S. Drysdale. Simulating numerically controlled machining in parallel. In J. Jaja, editor, *Third Symposium on the Frontiers of Massively Parallel*

Computation. Proceedings, pages 80–9, Los Alamitos, Cal., 1990. IEEE Computer Society Press.

[12] W. H. Walstra, W. F. Bronsvoort, and J. S. M. Vergeest. Interactive simulation of robot milling for rapid shape prototyping. *Computers and Graphics*, 18(6):861–871, 1994.

Appendix B

Sample Input File

```
# Sample input file for hybrid path generation
# General format: window specification, surfaces, profiles
#
# Window Data:
# --window spec.: x left, x right, y bottom, y top.
W -1.5 1.5 -1.5 1.5

# Surface Data:
# --number of surfaces included below
1
#
# --surface #1 header: degree u, v, # ctrl pts u, v
S 1 1 2 2
#
```

```
# --knot list (terminated by 'E')
# --u knot list
0 0 1 1
# --v knot list
0 0 1 1
E
#
# --point list (terminated by 'E')
p -3.0 -3.0 -0.75
p  3.0 -3.0 -0.75
p -3.0  3.0 -0.75
p  3.0  3.0 -0.75
E

# --other surface would follow here...


# Profile Data:
# --number of profiles (0, 1, or 2) followed by type of zone:
#      0 = exclusion zone; 1 = inclusion zone
1 0

#
```

```
# --data for first profile
# --each profile has a profile header & the given number of curves
# --header format: P <ncurves> <offdir> <closedflag> <always0>
#   where:
#      ncurves      - number of Bsplines below (must have common pt)
#      offdir= 1    - offset on right side of profile
#            = 0    - apply no offset
#            =-1    - offset on left side of profile
#      closedflag=0 - profile is open (must intersect window above)
#                =1 - profile is closed curve
#      always0      - must be 0

# --profile #1 header
P 3 0 1 0

# --curve 1
# --curve header: degree, # ctrl. pts.
C 1 2
# --knot vector (terminated by 'E')
0 0 1 1
E
# --control point list (terminated by 'E')
p  0.20 -0.20 1.0
```

```
p  0.0   0.20 1.0
E

# --curve 2
C 1 2
0 0 1 1
E
p -0.20 -0.20 1.0
p  0.20 -0.20 1.0
E

# --curve 3
C 1 2
0 0 1 1
E
p  0.0   0.20 1.0
p -0.20 -0.20 1.0
E

# --profile #2, if present, goes here

# --end of file
```

Bibliography

[Bre90a] E. L. Brechner. *Envelopes and tool paths for three-axis end milling.* PhD thesis, Rensselaer Polytechnic Institute, 1990.

[Bre90b] E. L. Brechner. General tool offset curves and surfaces. *Advances in Design Automation*, DE v.23 pt.1:273–82, 1990.

[CC89a] J. J. Chou and E. Cohen. Computing offsets and tool paths with voronoi diagrams. Technical Report UUCS-89-017, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[CC89b] J. J. Chou and E. Cohen. Constant scallop height tool path generation. Technical Report UUCS-89-011, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1989.

[CCPK94] B. K. Choi, Y. C. Chung, J. W. Park, and D. H. Kim. Unified CAM-system architecture for die and mould manufacturing. *Computer Aided Design*, 26(3):235–43, March 1994.

[CJ89] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in NC machining of sculptured surfaces. *Computer Aided Design*, 21(6):371–8, July-August 1989.

[CPJ93] B. K. Choi, J. W. Park, and C. S. Jun. Cutter-location data optimization in 5-axis surface machining. *Computer Aided Design*, 25(6):377–386, June 1993.

[EE94] G. Elber and Cohen E. Toolpath generation for freeform surface models. *Computer Aided Design*, 26(6):490–496, June 1994.

[Elb92] G. Elber. *Free form surface analysis using a hybrid of symbolic and numeric computation.* PhD thesis, University of Utah, 1992.

[Gri85] J. G. Griffiths. Table-driven algorithms for generating space-filling curves. *Computer Aided Design*, 17(1):37–41, January-February 1985.

[Gri94] J. G. Griffiths. Toolpath based on Hilbert's curve. *Computer Aided Design*, 26(11):839–844, November 1994.

[Guy89] M. Guyder. Automating the optimization of $2\frac{1}{2}$ axis milling. In *Proceedings, Computer Applications in Production & Engineering, CAPE '89.* North-Holland, October 1989.

[Hel91] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1991.

[HNF+86] A. Haapaniemi, H. Nagase, M. Fujimoto, H. Hiraoka, F. Kimura, and T. Sata. Development of a real time numerical controller for machining of sculptured surfaces. In *Software for Discrete Manufacturing, Proceedings of the 6th International IFIP/IFAC Conference*, pages 205–214, Amsterdam, 1986. North-Holland.

[Hwa92] J. S. Hwang. Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer Aided Design*, 24(12):667–76, December 1992.

[KKS87] T. Kishinami, T. Kondo, and K. Saito. Inverse offset method for cutter path generation. In *Proceedings, 6th International Conference on Production Engineering*, pages 807–812, Osaka, Japan, November 1987.

[LDV94] H. Li, Z. Dong, and G. W. Vickers. Optimal toolpath pattern identification for single island, sculptured part rough machining using fuzzy pattern analysis. *Computer Aided Design*, 26(11):787–785, November 1994.

[LJ94] S. X. Li and R. B. Jerard. 5-axis machining of sculptured surfaces with a flat-end cutter. *Computer Aided Design*, 26(3):165–178, March 1994.

[MG94] S. Marshall and J. G. Griffiths. A new cutter-path topology for milling machines. *Computer Aided Design*, 26(3):204–14, March 1994.

[NDW93] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1.* Addison-Wesley, Reading, Massachusetts, 1993.

[Ope92] OpenGL Architecture Review Board. *OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1.* Addison-Wesley, Reading, Massachusetts, 1992.

[Per78] H. Persson. NC machining of arbitrarily shaped pockets. *Computer Aided Design*, 10(3):169–174, May 1978.

[Sag94] H. Sagan. *Space-filling Curves.* Springer-Verlag, New York, 1994.

[SKS+91] H. Suzuki, Y. Kuroda, M. Sakamoto, S. Haramaki, and H. Van Brussel. Development of the CAD/CAM system based on parallel processing and the inverse offset method. In P. Welch, D. Siles, T.L. Kunii, and A. Bakkers, editors, *Transputing '91. Proceedings of the World Transputer User Group (WOTUG) Conference*, pages 184–98, Amsterdam, 1991. IOS.

[ST91] T. Saito and T. Takahashi. NC machining with G-buffer method. *Computer Graphics*, 25(4):207–16, July 1991.

[Tho90] S. W. Thomas. Scanline rendering for 3-axis NC toolpath generation, simulation, and verification. Technical Report CSE-TR-43-90, University of Michigan, Ann Arbor, Michigan, January 1990.

[TSAO89] Y. Takeuchi, M. Sakamoto, Y. Abe, and R. Orita. Development of a personal CAD/CAM system for mold manufacture based on solid modeling techniques. *Annals of the CIRP*, 38(1):429–432, 1989.

[Vep90] A. M. Vepsäläinen. An application of morphological filters to NC-programming. *Proceedings of the SPIE - The International Society for Optical Engineering*, 1350:177–83, 1990.