# Pocket Milling with Tool Engagement Detection

**Thomas R. Kramer**

**April 4, 1991**

ABSTRACT

This paper presents an algorithm for generating a tool path for cutting a pocket with islands, which includes detecting when the tool is certain to be making a minimal engagement cut. Minimal engagement is defined as the amount of engagement present in peripheral milling of a flat side face of a workpiece in a straight line with constant stepover. Feed rates and spindle speeds are reset by the algorithm when machining conditions change from a minimal engagement cut to any other type of cut or vice versa.

KEYWORDS

milling, tool path, pocket, algorithm, automated manufacturing, machining, NC-program, numerical control

Funding for the work described in this paper was provided to Catholic University by the National Institute of Standards and Technology under cooperative agreement Number 70NANB9H0923.

## 1 Background

Research in the Factory Automation Systems Division at the National Institute of Standards and Technology has included the development of an Off-Line Programming System (OLPS) for automatically generating NC-programs for a vertical machining center [Kramer2]. Investigating existing machining techniques and developing new techniques have been elements of this work.

### 1.1 Pocket

One of the most common operations in machining metal parts is pocket milling: removing all the material inside some arbitrary closed boundary on a flat surface of a workpiece to a fixed depth. Such a shape is frequently called a generalized pocket, or (more simply) a *pocket*. We will use *pocket* in this paper. We make the further limitation that the boundary be composed solely of arcs of circles and straight line segments joined together in a continuous chain. Any boundary can be approximated arbitrarily closely by such a chain without introducing any new sharp corners. Most discussions of pockets and systems for cutting them make this limitation. Often a pocket with islands in it (material surrounded by the same type of boundary which is not to be removed) will be specified in a design. We will assume in this paper that a pocket may have islands in it. A block-shaped workpiece with a pocket containing two islands is shown in Figure 1. Because a pocket may be characterized by a depth and set of closed boundaries on a plane, most other figures in this paper will be two-dimensional. We will assume that the bottom of a pocket is flat and is perpendicular to the side walls of the pocket.

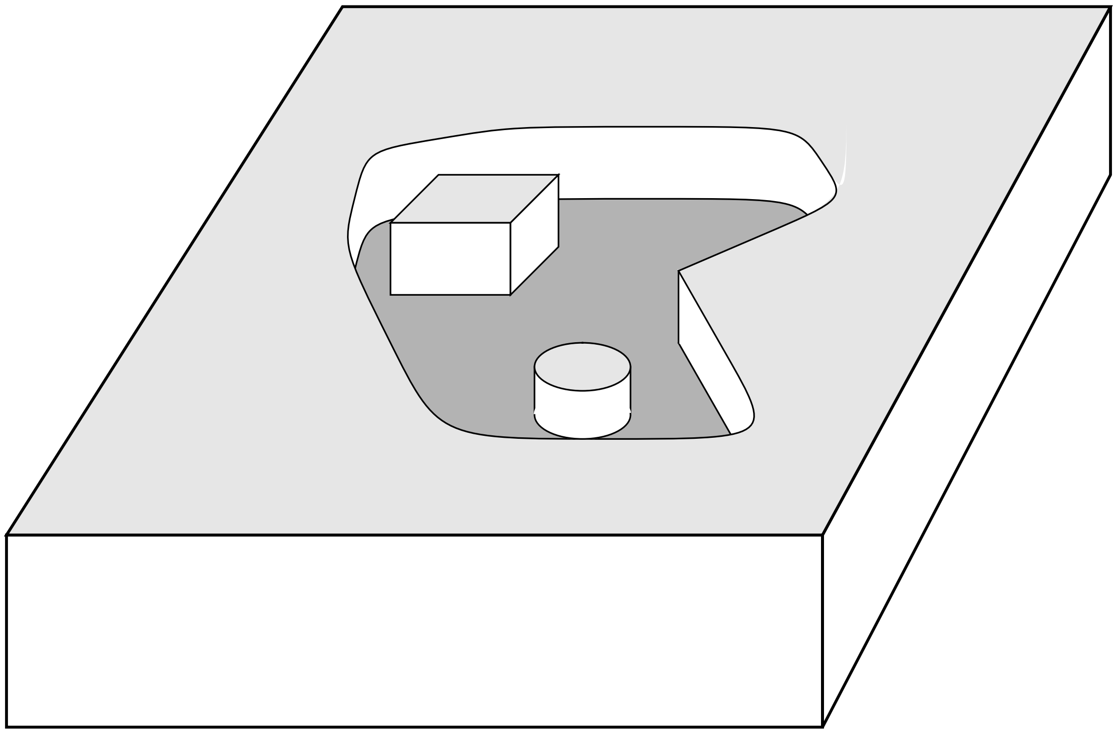


**Figure 1. Pocket With Two Islands**

## 1.2 Milling a Pocket

### 1.2.1 Overview

It is common machining practice to form a pocket by milling it on a milling machine or machining center using one or more flat-bottom end mills.

If the area inside the boundary of a pocket is at all large in comparison to the cross section of the end mills which are suitable for cutting it, it is common practice to make a "roughing cut" to remove the bulk of the material by making a slightly smaller pocket, leaving a thin (0.0254 centimeter or 0.01 inch is typical) layer of material which is later removed by a finish end mill. We are not concerned in this paper with how the pocket is finished, but only with the removal of the bulk of the material. Bulk removal may be accomplished with the same mill used for finishing the pocket, but will more commonly be done with a roughing end mill for efficiency. The algorithm described here is better suited to a roughing end mill than a finish end mill, for reasons given below.

It is rarely feasible to push an end mill straight down into material more than about the thickness of the flutes on the end mill (about a millimeter), so pocket milling is usually initiated by making a starter hole. This may be done with a different tool or (if the end mill is a center cutting end mill - meaning it has cutting surfaces that extend into the center of the end) by ramping back and forth or cutting along a helical path into the material.

Once a starter hole is made, the end mill is inserted in the hole to a fixed depth and then driven parallel to the surface of the workpiece in some path to cut away the bulk of the material in the pocket. If the pocket is deep (more than about one tool diameter), it is often not feasible to cut to the full depth immediately. Then, the entire pattern of the tool path is repeated at increasing depths until the full depth is reached, cutting out a layer at a time.

If a pocket has constricted regions, such as the middle of an hourglass, into which only a very small diameter end mill will fit, the pocket is generally divided into subparts so that most of the material can be removed with a larger end mill. We will not deal here with pockets that must be subdivided.

Some algorithms for milling pockets also allow concave corners with radii smaller than the tool radius, leaving material in such corners. These algorithms assume the user will notice such corners and remove the material with some other tool. We assume here that the tool will fit into all corners.

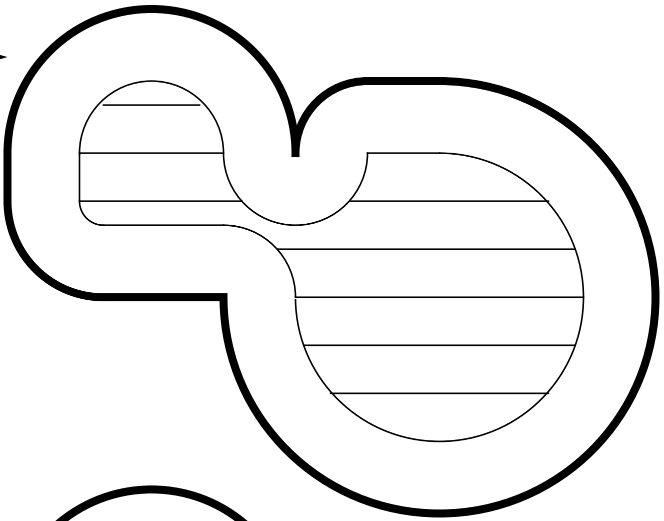
If part of the boundary of a pocket is accessible from the side for some reason (such as an adjoining pocket already having been machined), then the most efficient method of machining is probably to mill away at the exposed portion of the boundary, and the algorithm given here (or any other algorithm assuming no side access) will probably be significantly less efficient.

### 1.2.2 Pocket Milling Algorithms

Three common approaches are taken to pocket milling: *unidirectional*, *zig-zag*, and *arachnoid* (like a spider web). The three approaches are illustrated and described in Figure 2. The figure does not show any islands, but none of the algorithms changes qualitatively if islands are included. The boundary of any island is treated like more pocket boundary. The only major difference is that with islands there may be several loops of boundary, not just one. The distance between the boundary of the pocket (or of an island) and the tool path required to cut it is necessarily equal to the radius of the end mill being used. The distance between the parallel lines in the unidirectional and zig-zag algorithms or between loops in the arachnoid algorithm, is called *stepover*.

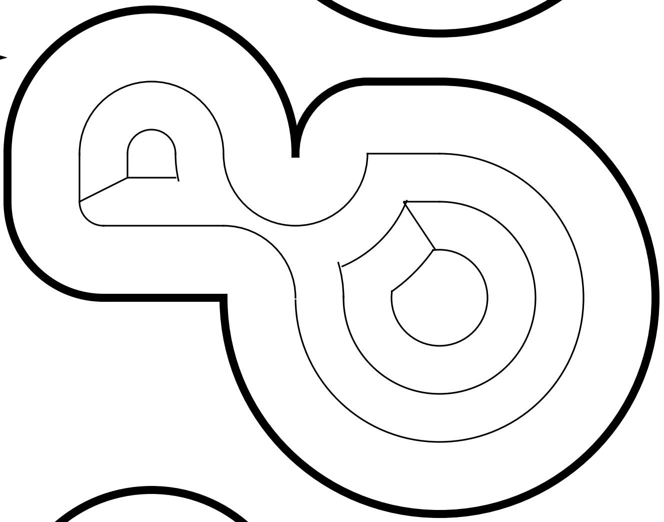
**Unidirectional**

The entire periphery is cut first. Then the horizontal lines are cut from top to bottom by inserting the tool at the right end of each line and cutting to the left end. The tool is withdrawn at the end of one line and moved to the beginning of the next before being inserted again. The number of starter cuts required is one more than the number of islands.



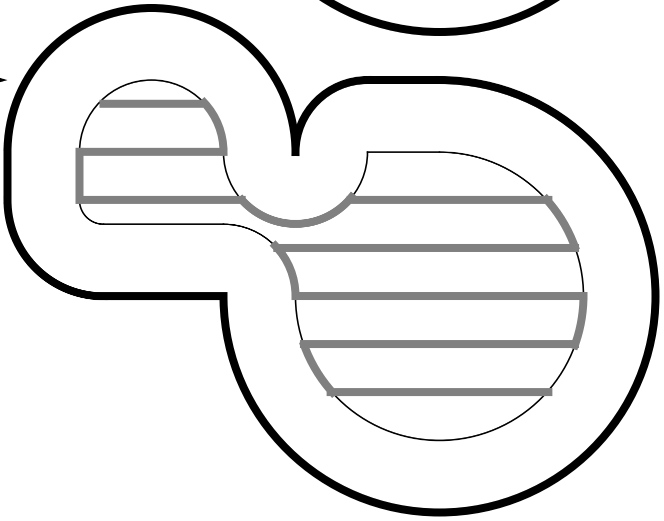
**Arachnoid**

The two inner loops at the right are cut first. The tool is withdrawn and reinserted to cut the loop on the left. Then the outer tool path is cut. The number of starter cuts required depends on the shape of the pocket and any islands. In the most common traversal of the loops, one starter cut is used for each loop having no other loop inside it. Alternative traversals of the path using fewer starter cuts (e.g. by slotting between loops) are feasible.



**Zig-Zag**

The zig-zag pattern (shown in heavy dotted lines) is cut first. Then the outer tool path (light solid line) is cut. Regardless of the shape of the pocket or number of islands, only one starter cut is required at the beginning of the zig-zag. The tool may or may not be withdrawn and moved between completing the zig-zag and starting the outer tool path.



## Figure 2. Pocket Milling Algorithms

This figure shows the three principal algorithms for milling a pocket. The same pocket is shown (heavy solid line) in each subfigure. Starter cuts and moves in air are not shown.

The value of the stepover depends on many factors, which we will not discuss further here. The stepover must be less than the diameter of the tool, of course. The unidirectional and zig-zag algorithms are not much affected by the size of the stepover. The arachnoid algorithm, however, is tuned to a stepover of one tool radius, since then each inner loop is exactly the outline of the island left by cutting the next outer loop (or a portion of it). If the stepover is more than a tool radius in the arachnoid algorithm, islands of material may be left. The algorithm can be modified to solve this problem, of course.

The parallel lines in the unidirectional and zig-zag algorithms will be called scan lines in this paper. They are shown as horizontal lines in Figure 2. With the conventional orientation of a two-dimensional coordinate system, the two ends of a horizontal line have the same y-value, which makes calculations easier. Some implementations of these two algorithms allow the scan lines to be at some other angle, although they must still be parallel to one another.

The tool path for cutting the boundary of the pocket and any islands will be called the *outer tool path*. This path is the same for any algorithm, as shown in Figure 2.

### 1.3 Previous Work

A description of an implementation of the unidirectional algorithm is given in [Kramer1].

Excluding minimal engagement detection, variants of the zig-zag algorithm described here for determining tool paths have been known and in use in commercial systems for some time (see [Kishi] or [Stoutenborough], for example), but details are never given by other authors. The more sophisticated systems generally implement the arachnoid algorithm. Some of these (as described in [Lallande], [Preiss] and [Held], for example) are said to adjust feeds and speeds according to cutting conditions. Others (those described in [Hansen] and [Guyder], for example), do not claim to. The minimal engagement detection algorithm described here has not, to the author's knowledge, been described elsewhere.

In the arachnoid algorithm, there are two distinct approaches to generating the nested loops which form the cutter path. In the first method, as implemented by [Hansen], for example, the loops are generated by starting at the outer tool path and working inwards iteratively, on each iteration creating a complete closed offset loop one stepover inside the previous one and then reducing or subdividing the offset loop by removing extraneous segments. Iteration stops when all reduced offset loops amount to nothing. In the second method, as implemented by [Held], for example, a Voronoi diagram for the pocket is generated first, and used on each iteration to help build the loop.

## 2 Cutting Conditions

Figure 3 illustrates a variety of cutting conditions. The figure shows the top view of a block of material being cut by five identical end mills, all turning clockwise (the normal way). The portion of the circumference of each mill which is engaged with material is shown with a heavy line. The straight arrows attached to end mills A, B, and C indicate the direction in which they are moving.

One useful distinction is between *slotting* and *peripheral milling*. Intuitively, slotting is cutting a slot whose width is the same as the diameter of the end mill and peripheral milling is cutting around the periphery of some material. End mill C in the figure is slotting, while A and B are peripheral milling.

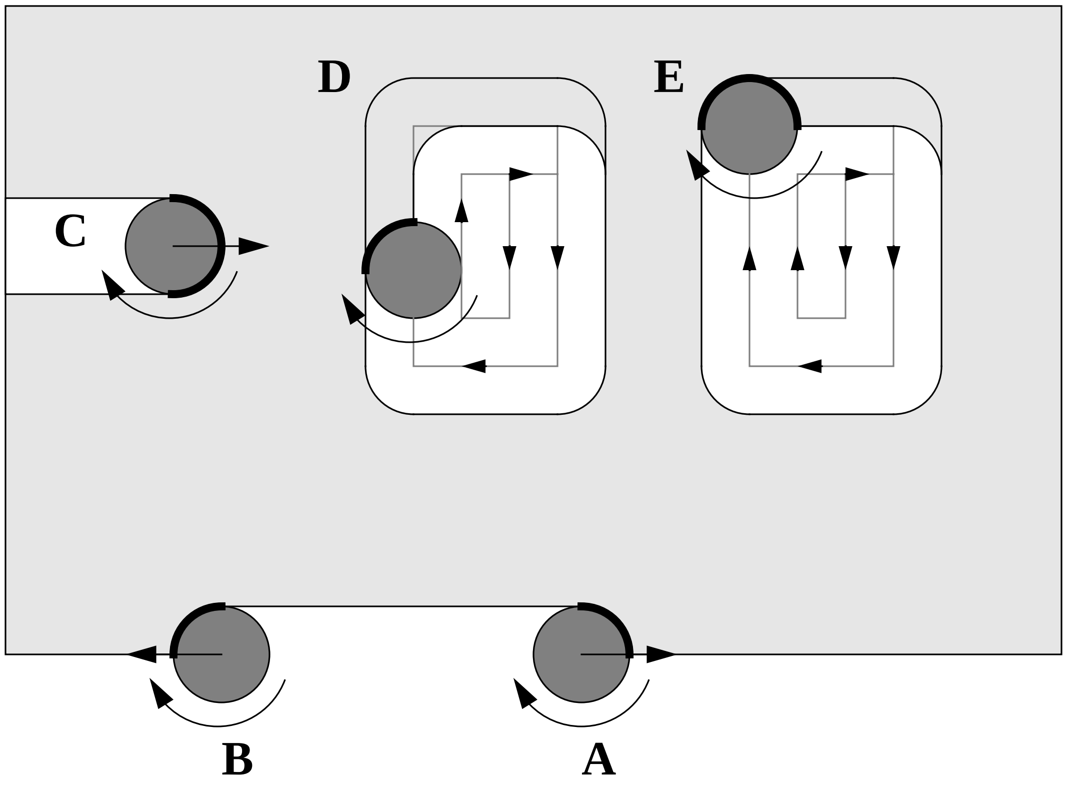


**Figure 3. Cutting Conditions**

In slotting, half the circumference of the end mill is engaged with the material of the workpiece. It is not possible to engage more than half the circumference since the half facing away from the direction of travel will not be engaged.

In peripheral milling, as shown in A and B, a smaller portion of the circumference of the end mill is engaged with the material. The amount engaged varies with the stepover. When the stepover is one tool radius, as shown in A and B, a quarter of the tool circumference is engaged. If the stepover is smaller, less circumference is engaged, and if it is larger, more circumference is engaged.

A second useful distinction is between *conventional cutting* and *climb cutting*, a distinction that occurs only during peripheral milling. In the figure, end mill A is doing conventional cutting while B is climb cutting. The difference is that in conventional cutting, the flutes or teeth of the mill first enter the material tangent to the surface being cut (and at the thin end of the chip being formed), while in climb cutting they enter perpendicular to the surface (and at the thick part of the chip). There is also a small difference in the speed at which the teeth pass through the material (usually called *surface speed*), because in climb cutting the teeth cut in the opposite direction from the feed, but in conventional cutting they cut in the same direction. This difference is almost always negligible, since the surface speed caused by tool rotation is typically more than a hundred times the feed rate. For example, a one centimeter end mill cutting aluminum might rotate at 4000 rpm, giving a speed of over 12000 centimeters per minute, while the feed rate is likely to be in the range of 30 centimeters per minute, one 400th the speed due to rotation.

Typically, the greatest rate at which material may be economically removed by peripheral milling is much greater than the rate possible with slotting, so peripheral milling is preferred. Thus, each of the three algorithms for cutting pockets is designed to have peripheral milling occur as much as possible, within the inherent limitations of the algorithm.

If a peripheral milling cut is being made in a straight line on a flat wall (as in A and B in Figure 3), the engagement of the tool with the material is clearly not changing, so the stepover, speed, and feed can be adjusted for optimum cutting.

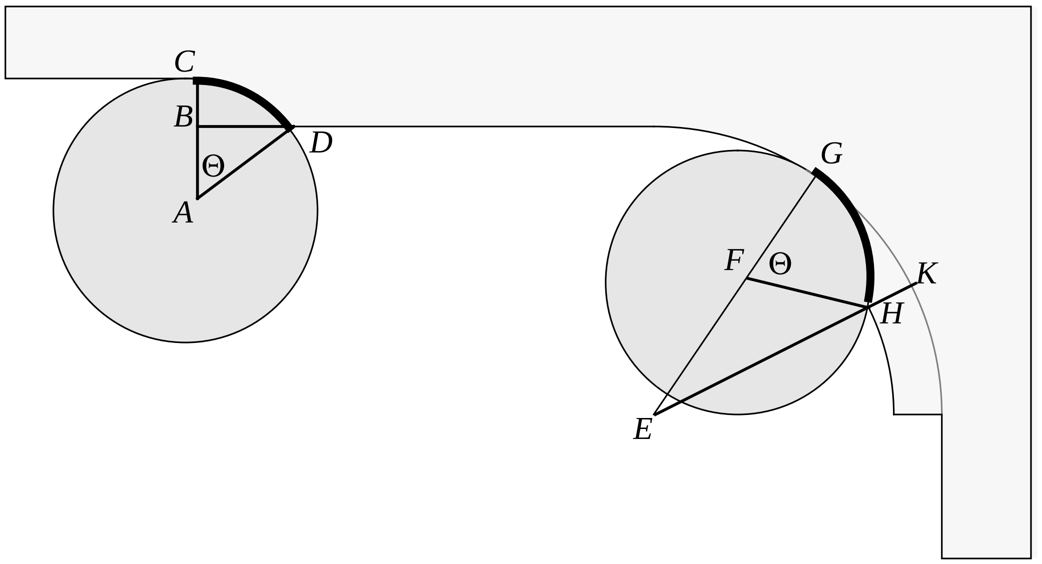
Cutting a pocket, however, involves dealing with complex shapes and tool paths. Maintaining constant tool engagement with the workpiece is not achieved by any known implementation of any of the three algorithms. Moreover, the geometry of tool engagement is trickier than one might expect. An example is shown in D and E of Figure 3. These subfigures show tools in the process of cutting identical pockets by the arachnoid algorithm, the tool path for which is shown with a dotted line. Notice that the width of the stepover is even all the way around. The portion of the circumference of the tools engaged with material, however, differs by a factor of two. Tool D has a quarter of its circumference engaged, while E has half its circumference engaged. This is a significant and very real difference. When cutting of this sort is done on a machining center, the load indicator moves up and the sound deepens each time the tool approaches a corner of the tool path, changing back as soon as the corner is turned.

Equations for the angle of engaged circumference may be derived as shown in Figure 4. On the left of the figure, the calculation is for cutting in a straight line with a constant stepover. On the right, the calculation is for cutting in a circular arc with a constant stepover. If the stepover equals the tool radius, the equation on the right simplifies to $\Theta = \Pi - \mathrm{acos}\,(r/[2(R - r)])$. When the radius of the arc being cut is not several times the tool radius, the engaged angle becomes significantly larger than the engaged angle for the same tool and stepover when cutting in a straight line. For example, if $R = 2r$, and $r = s$, the engaged angle on a straight line is 90˚, while the engaged angle is 120˚ on the arc, an increase of 33 1/3 percent.

Note that the point $H$ must exist for the derivation on the right to apply. A typical limiting case is when the stepover equals the tool radius, the point $H$ does not exist if $R$ is less than 1.5 times $r$.

It is also interesting to consider the difference in material removal rates between straight line cutting and cutting along an arc. The volume removal rate, $V$, for cutting in a straight line is the depth of cut, $d$, times the stepover, $s$, times the feed rate, $f$. In other words, $V = d\ s\ f$.

For cutting in an arc, with reference to Figure 4, the volume removal rate, $V$, is the depth of cut, $d$, times the stepover, $s$, times the rate of motion of the midpoint of $HK$, (since the midpoint of $HK$ moves at the average speed of the points on $HK$).



On the left, an end mill is cutting in a straight line. We let
*s* = stepover = *BC*
*r* = tool radius = *AC* = *AD*
Θ = engaged angle = angle *DAC*
From line *AC*, *AB* = *AC* - *BC* = *r* - *s*
Then since *ABD* is a right triangle:

$$\cos\Theta = \frac{AB}{AD}$$

$$\cos\Theta = \frac{r-s}{r}$$

$$\Theta = \operatorname{acos}\left(\frac{r-s}{r}\right)$$

On the right an end mill is cutting in a circular arc, removing a uniform thickness of material. We let
*s* = stepover = *HK*
*r* = tool radius = *FG* = *FH*
Θ = engaged angle = angle *GFH*
*R* = radius of arc being cut = *EG* = *EK*
From line *EG*, *EF* = *EG* - *FG* = *R* - *r*
From line *EK*, *EH* = *EK* - *HK* = *R* - *s*
Angle *EFH* = Π - Θ, since *EFG* is straight.
But we may find angle *EFH* by using the law of cosines on triangle *EFH*:

$$\angle EFH = \operatorname{acos}\left(\frac{(EF)^2 + (FH)^2 - (EH)^2}{2\,(FH)\,(EF)}\right)$$

$$\Pi - \Theta = \operatorname{acos}\left(\frac{(R-r)^2 + r^2 - (R-s)^2}{2r\,(R-r)}\right)$$

$$\Theta = \Pi - \operatorname{acos}\left(\frac{2r^2 + 2R\,(s-r) - s^2}{2r\,(R-r)}\right)$$

**Figure 4. Angle of Engaged Circumference**

The feed rate $f$ applies to the center of the tool, so the feed rate at the midpoint of $HK$ will be in the proportion of (i) the lever arm from $E$ to the center of $HK$ to (ii) the lever arm $EF$ from $E$ to the center of the tool. This is $(R - s/2)/(R - r)$, so $V = d\ s\ f\ (R - s/2)/(R - r)$.

A more rigorous (but less intuitively clear) way to get this result is to observe that the removal rate for cutting on an arc equals the volume removed on one full circle times the rate of making full circles. The former is the depth of cut times the area $\Pi(R^2 - (R - s)^2)$ of the annulus made by the stepover, while the latter is the feed rate divided by the circumference of a full circle $f / 2\Pi(R - r)$. Thus the volume removal rate is given by $V = d\ f\ \Pi(R^2 - (R - s)^2) / 2\Pi(R - r)$, which reduces to the same thing as above.

Using the same example as earlier ($R = 2r$ and $r = s$), the volume removal rate on the arc is

$V = 1.5\ d\ s\ f$, an increase of fifty percent from the rate on a straight line.

The derivations for a cutting on an arc apply to concave cutting. If a convex arc is being cut, there are decreases of comparable size in tool engagement and cutting rates.

In order to cut a pocket safely, an algorithm that does not distinguish between when it is slotting and when it is peripheral milling must set its feeds and speeds for the worst case, which is slotting. This will yield cutting times much higher than an algorithm which distinguishes slotting from peripheral milling and changes speeds and feeds appropriately. As we have just seen, however, even if peripheral milling is recognized, if we do not detect when the tool is minimally engaged, it will still be necessary to reduce feeds and speeds somewhat to allow for the deeper cuts and increased volume removal rates that may occur during peripheral milling.

# 3 The Algorithm

## 3.1 Introduction

The zig-zag algorithm with minimal engagement detection has been implemented by the author in an automatic NC-code generation system called the Off-Line Programming System (OLPS), written in LISP [Kramer2]. In OLPS, the algorithm is used to generate code for rough milling pockets. It is intended that rough-cutting end mills, which behave about the same in conventional cutting and climb cutting, be used to do the cutting, because the algorithm alternates between conventional cutting on the zig and climb cutting on the zag. The algorithm generates both a tool path and instructions for changing spindle speeds and feed rates along the path. We will describe path generation first and then feed and speed control.

The algorithm described here allows an entry hole to be made at any point in the pocket, and only one entry hole is needed. All other entries are provided by the material cut away by the operation of the algorithm. The OLPS implementation offers a choice of methods for making an entry hole, but that is not significant in the remainder of the procedure.

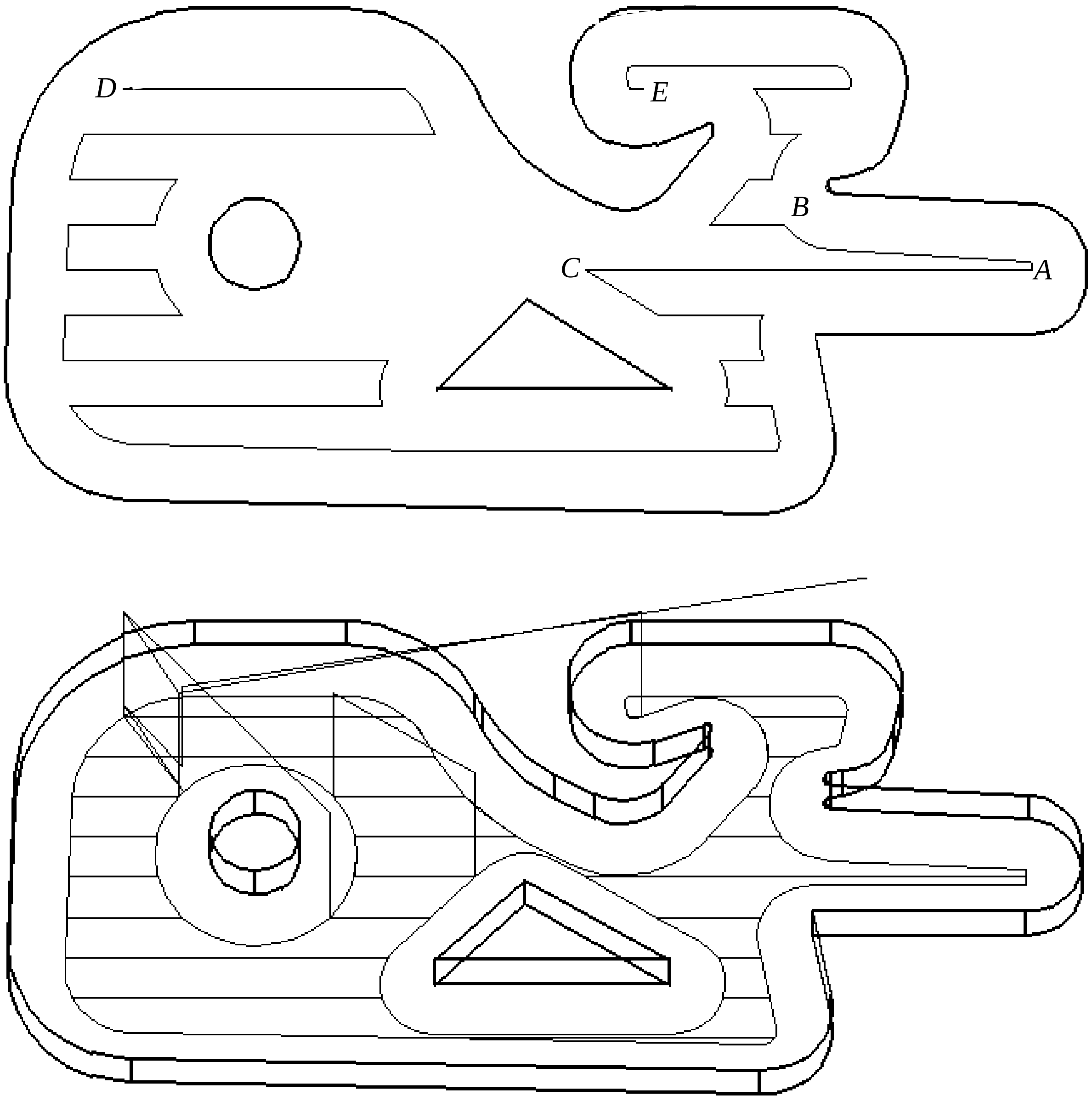


**Figure 5. Pocket Milling Example**

## 3.2 Tool Path Generation

Figure 5 shows the tool path generated by the algorithm for a moderately complex pocket with two islands. The figure was drawn by the OLPS implementation. The upper half of the figure shows the top view of the first zig-zag generated (what gets generated before the first time step 3C*2* is used as described in the next section). This first zig-zag starts at *D* and ends at *E*. The bottom half of the figure shows a 3-dimensional view of the entire tool path, including the retractions and reinsertions of the tool.

### 3.2.1 Summary

In outline form, the algorithm for generating the tool path is:

1. Generate the outer tool path. As defined earlier, this is the set of one or more continuous closed tool paths needed for cutting the final boundary of the pocket and any islands. Do not write NC-code for cutting the outer path yet.
2. Put a set of horizontal lines (called *scan lines*) across the pocket and find all intersection points where the scan lines cross the outer tool path. The portions of the scan lines lying inside the pocket between intersection points will be called *scan segments*.
3. Traverse the scan segments and portions of the outer tool path as follows to clear the pocket. Generate NC-code for the traversal during the traversal. Any intersection point encountered during the traversal is said to have been "visited".

    A. Choose an intersection point at which to start, and cut to it from the entry point.

    B. Follow the scan segment through the point to the intersection point at other end of the segment.

    C. Continue the zig-zag by:

    *1*. If at least one of the neighboring intersection points along the outer tool path of the point just reached has not been visited, follow the outer tool path one way or the other to such a point. Then go back to step 3B.

    *2*. If there is no unvisited neighboring intersection point along the outer tool path,

    *a*. If there is at least one unvisited intersection point, find an unvisited intersection point, P, one of whose two neighbors along the outer tool path has already been visited. Jump over to the neighbor and cut along the outer tool path to P. Then go to step 3B.

    *b*. If there is no unvisited intersection point, the zig-zag is complete. Go to step 4.

4. Generate NC-code for cutting along the outer tool path determined in step 1 for the pocket and any islands.

### 3.2.2 Details

***step* 1** Each continuous piece of the outer tool path is generated by offsetting by one tool radius from the boundary of the pocket or one of its islands.

***step* 2** The location of the scan lines in the y-direction is determined by setting the y-value of one scan line. This may be done by the user or automatically; OLPS provides an option. If the outer tool path could fit in between two scan lines, care must be taken to ensure that at least one scan line intersects it.

The x-values for the ends of the scan lines may be taken as the maximum and minimum values of x reached on the pocket itself, since all tool paths lie entirely within the pocket.

In finding the intersections of the scan lines with a closed loop in the outer tool path, two methods are feasible: (i) checking for intersections of every scan line with every straight line segment and arc in the loop, or (ii) following around the loop and, whenever an intersection point is found, checking the scan line on which the point lies, plus the two scan lines above and below for the next intersection point. The second method is much more efficient and has been implemented. In the OLPS implementation, generating the path for getting from each intersection point to its neighbor(s) is done at the same time the outer tool path is being traversed to find the intersections. Each such path is a portion of the outer tool path.

***step* 3A** The first intersection point and the entry point must be selected so that the tool stays within the pocket when cutting on a straight line between the two.

***step* 3B** The point at the other end of the scan segment cannot have been visited yet, since if it were, the point at this end of the scan segment would also have been visited.

***step* 3C*1*** In most cases, when the end of a scan segment is reached, one of its neighbors along the outer tool path will have been visited at the beginning of the previous scan segment. Thus, if the zig-zag pattern starts going down (towards negative y), it will continue going down, and if it starts up, it will continue up. When there is a choice of two neighbors, the choice may be made arbitrarily. In the OLPS implementation, a note is made of whether the zig-zag is currently going up or down, and when there is a choice, the zig-zag is continued in the same direction. When a section of outer tool path is cut, a note is made in the data for each end point, indicating that the portion of outer tool path going up or down from the point (as the case may be) has been cut, so that this information is available in step 4.

***step* 3C*2a*** There must be at least one point, P, of the sort being sought. It is approached along the outer tool path from a previously visited neighbor because that route is known to be possible and safe. In fact, this is one reason for following the outer tool path throughout step 3. Cutting might be more efficient in a straight line, but if a straight line were tried, a check would need to be made whether the cut would gouge the part, and some other path would have to be generated if so, complications which are avoided by using approach along the outer tool path.

***step* 3C*2b*** If there is no unvisited intersection point, all the scan segments must have been cut. This is because every point visited is visited at the beginning or end of cutting the scan segment on which the point lies.

***step* 4** Much of the outer tool path will already have been traversed in step 3, but there will be some uncut portions. An efficient way of traversing the uncut portions of the outer tool path is to cut around the entire path. In order to minimize the time required for cutting around the entire path, portions which have already been cut are cut at a high feed rate (127 centimeters or 50 inches per minute in the OLPS implementation).

## 3.3 Minimal Engagement Detection

Minimal engagement is defined as the amount of engagement present in peripheral milling of a flat side face of a workpiece in a straight line with constant stepover (such as at A and B in Figure 3 or on the left in Figure 4).

Minimal engagement detection is performed by the algorithm for cuts along scan segments in the zig-zag traversal described in step 3. This is performed concurrently with generating the traversal. Sometimes minimal engagement milling will be taking place on the scan segments and sometimes non-minimal. The feed rate and spindle speed are changed appropriately. Minimal engagement detection for the parts of the zig-zag that lie on the outer tool path is not performed. All those parts are cut as though slotting were always taking place. It would be feasible to implement minimal engagement detection on these parts also, but this has not been done.

Figure 6 shows the same pocket as Figure 2, with additional graphics regarding minimal engagement detection.

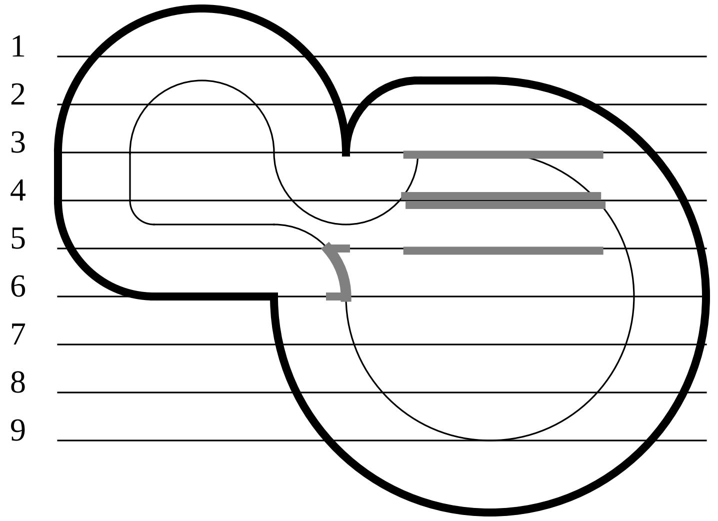


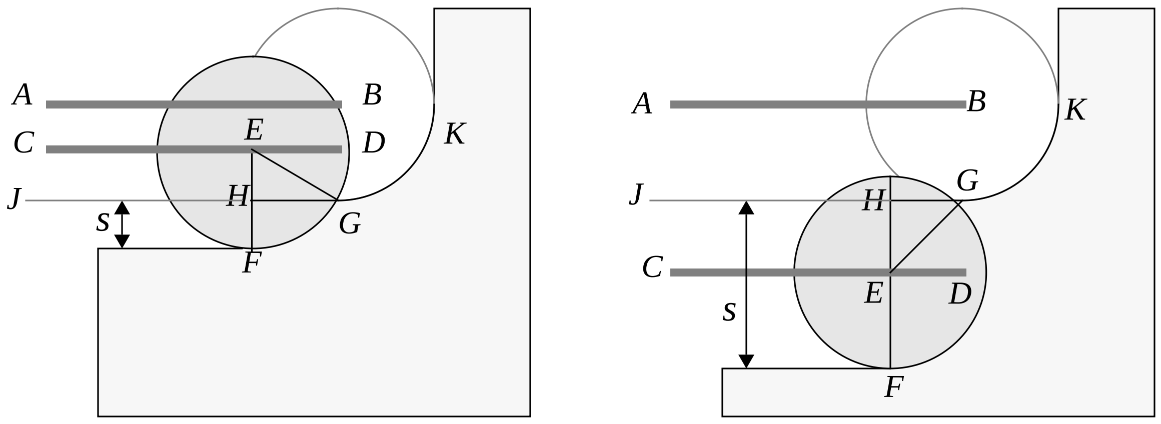
**Figure 6. Scan-clear Lines**

The scan-clear lines are the thin horizontal lines numbered 1 to 9. When a cut is made along a heavy dotted line ▬, parts of the adjacent scan-clear lines are marked clear: ▬

To detect minimal engagement on scan segments, a set of data is maintained which keeps track of where cuts will be minimal when made. We will describe a conceptual view of the data here and omit details of the OLPS implementation. The data represents another set of scan lines, s*can-clear lines,* directly on top of the scan lines used to find intersections with the outer tool path. The scan-clear lines are initialized as shown by the thin horizontal lines in Figure 6. The horizontal extent of the scan-clear lines is the same as that of the pocket being cut. The topmost scan-clear line is one stepover above the topmost cutting scan line, and the lowest scan-clear line is one stepover below the lowest cutting scan line.

When a scan segment is cut, the portion of the scan-clear lines on, above, and below the one being cut are marked as being clear. If a section of outer tool path is cut, the scan-clear lines above and below the section of outer tool path being cut are marked as clear. This is illustrated in Figure 6. If the scan segment marked in a dotted line on scan line 4 were cut, the portions of scan-clear lines 3, 4, and 5 shown with the long grey bars would be marked as clear. If the part of the outer tool path between scan lines 5 and 6 marked with a dotted line were cut, the portions of scan-clear lines 5 and 6 marked with short grey bars would be marked as clear. If two clear portions of a scan-clear line overlap, they are combined into one.

Whenever a scan segment is being cut, that portion of it that is clear is cut at the minimal engagement feed rate and spindle speed, while the portion not marked clear is cut at the slotting feed rate and spindle speed. However, the clear portion of a scan line being cut is shortened at the end toward which the tool is moving by an amount $d$ given by the formula in Figure 7, where $r$ is the radius of the tool and $s$ is the stepover. The reason for this, as illustrated in Figure 7, is that the cut becomes heavier as the end of the clear line is approached. The cut is not heavier at the beginning of the clear line because the cutting is occurring on the other side of the tool.



Both diagrams above show material (light shading) being cut from left to right along scan line *CD* by an end mill (darker shading), after having been cut previously along the scan line from *B* to *A*. The center of the tool has reached a point *E* where the point, *G*, of first contact of the tool with the boundary of existing material has come to the end of a straight portion of boundary, *JG*, and started to lie on the arc *GK*. On the left, the stepover, $s$, is less than the tool radius, $r$. On the right, the stepover is larger than the tool radius.

The circle shown in a dotted line shows the outline of the tool when its center was at *B*, so it may be seen why arc *GK* exists.

Since scan line *AB* has already been cut, scan line *CD* is marked clear. However, it may be seen from the diagrams that as the center of the tool moves to the right of *E*, the amount of the circumference of the tool engaged with material, *FG*, will increase beyond minimal engagement as contact point *G* moves up the arc *GK*. We want to find the length, $d$, of *ED*, which is the amount by which the clear portion of scan line *CD* should be shortened. We observe first that *ED* = *HG*. We may find *HG* by using the Pythagorean Theorem on triangle *EHG*.

$$(HG)^2 = (EG)^2 - (EH)^2$$

On the left, *EH* = *EF* - *HF* = $r$ - $s$, while *EG* = $r$. thus,

$$d^2 = r^2 - (r-s)^2$$

$$d = \sqrt{2rs - s^2}$$

On the right, *EH* = *HF* - *EF* = $s$ -$r$,

$$d^2 = r^2 - (s-r)^2$$

$$d = \sqrt{2rs - s^2}$$

**Figure 7. Shortening Scan-clear Lines**

It is possible that a cut along the outer tool path will have the tool engaged less than in a minimal engagement cut, as defined here. The piece of outer tool path between points *A* and *B* on the right of the pocket in Figure 8 is an example. At the far right, because it is doubling back very close to previously cut scan segment *AC*, the tool is making a very light cut. However, as the tool nears point *B* and curves up into the material, the cut becomes heavier, so that at *B* the engagement of the tool with the material is much greater than minimal. Because of the complexity of monitoring the degree of engagement of a cut like this, the algorithm does not attempt to monitor tool engagement along the outer tool path, and all cuts along the outer tool path are at the spindle speed and feed rate for slotting.

A cut along a scan segment will be lighter than minimal also, if a piece of outer tool path has been cut previously close to it. Such cuts are treated like minimal engagement cuts.

Figure 8 shows the same pocket and zig-zag tool path as Figure 5 with the portions which are cut as minimal engagement cuts marked with grey bars. Data for the figure was generated by the OLPS implementation.

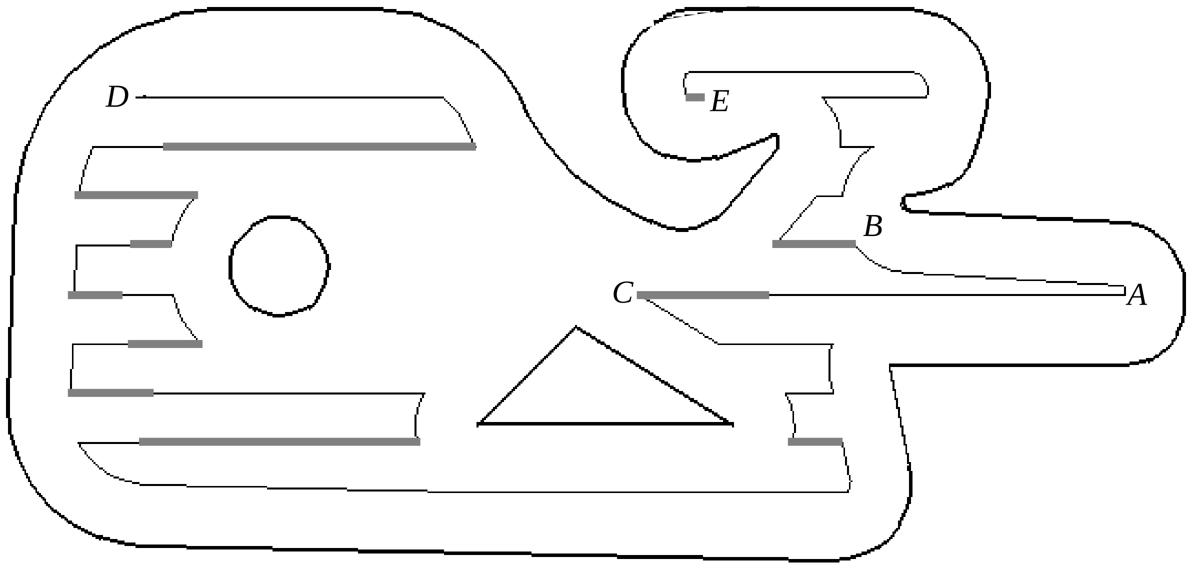


**Figure 8. Minimal Engagement Cuts**

## 3.4 Computational Hazards

If scan lines are simply intersected with the outer tool path, the number of intersections may not be predictable. For example, if two arcs meet at a y-maximum, and a scan line which is nominally at the same y-value is intersected, there may be zero to four intersections. This hazard is avoided in the OLPS implementation by following around the outer tool path.

In general, intersections of scan lines with the sections (arcs or lines) of the outer tool path near the end points of the sections should be avoided, since an intersection near one end point may or may not have a companion on the other section connecting to the point. This hazard is dealt with in the

OLPS implementation by moving the scan line vertically a small amount (much less than the least significant amount used by any machine tool controller) and recalculating the intersections, if an intersection is found near the end point of a section.

A scan line might also lie directly on top of a section of outer tool path. This is also handled by moving the segment vertically a small amount.

## 4 Concluding Remarks

### 4.1 Modeling Issues

Applying a typical commercial solid modeling system to the problem of detecting the degree of engagement of a tool with a workpiece or the problem of determining the instantaneous rate of material removal as a tool follows some path would be quite difficult because the shape of the part changes continuously with machining. Commercial solid modelers are generally set up to deal with static situations. The author is not aware of any solid modeler that was set up to deal with continuous changes of shape. Commercial solid modelers do not, for example, return the location of an intersection point as a function of the location of the items being intersected (such as the outline of a tool and the outline of a workpiece).

### 4.2 Further Work

It would be useful to have an algorithm for generating tool paths which put roughly constant magnitude forces on the tool. Such an algorithm might be based on maintaining constant engagement of the tool with the material, or on maintaining a constant rate of volume removal. Developing a practical algorithm of this sort may be expected to be wretchedly difficult. This is particularly so since machine tool controllers are not generally capable of controlling feeds and speeds through continuous changes. Programmed feeds and speeds must jump from one value to another. In the execution of the program, of course, acceleration and deceleration are required, as well, but the rates of acceleration and deceleration are not controllable by the user, or even revealed to the user. It seems likely that the problem of maintaining a constant load on the tool will be better solved by control and drive systems for machine tools which can sense the load and adjust feeds and speeds automatically to keep the load constant. Nevertheless, an algorithm which approximated a constant load with discrete changes of feed and speed would be very useful for existing machines and controllers.

The algorithm described here for minimal engagement detection falls well short of being optimum. It would be feasible to add some type minimal engagement detection for the outer tool path to the algorithm described here, and that would make a significant difference in cutting times. It would also be useful to add a third setting of feed and speed to be applied to cuts that were heavier than minimal but lighter than actual slotting.

It would be interesting to try to apply the techniques for feed and speed control described here to the arachnoid algorithm for tool path generation.

## References

[Guyder] Guyder, M. K., Automating the Optimization of 2 1/2 Axis Milling, source unknown

[Hansen] Hansen, A., and Arbab, F., An Algorithm for Generating NC Tool Paths for Arbitrarily Shaped Pockets with Islands, Technical Report CS 88-51, University of Southern California, Los Angeles, California, 1988

[Held] Held, M., GeoPocket - a Sophisticated Computational Geometry Solution of Geometrical and Technological Problems Arising From Pocket Machining, *Computer Applications in Production and Engineering*, F. Kimura and A. Rolstadas (Editors), Elsevier Science Publishers B. V., 1989, pp. 283 - 293

[Kishi] Kishi, H., Seki, M., and Tanaka, K., NC Automatic Programming in CAD/CAM Era, *“Advanced Manufacturing Technologies” Proceedings of the 21st Annual Meeting and Technical Conference,* Numerical Control Society, March 25 - 28, 1984, Long Beach, California, pp. 289 - 308

[Kramer1] Kramer, T. R., and Weaver, R. E., The Data Execution Module of the Vertical Workstation of the Automated Manufacturing Research Facility at the National Bureau of Standards, NBSIR 88-3704, National Bureau of Standards, Gaithersburg, MD, 1988

[Kramer2] Kramer, T. R., A STEP-Based Automatic NC-Programming System, to be submitted for publication as a NISTIR, National Institute of Standards and Technology, Gaithersburg, MD, 1991

[Lallande] Lallande, J. B. Jr., Purves, L., Walch, A. Jr., Premo, D. A., Super Pocket, *“Advanced Manufacturing Technologies” Proceedings of the 21st Annual Meeting and Technical Conference*, Numerical Control Society, March 25 - 28, 1984, Long Beach, California, pp. 18 - 29

[Preiss] Preiss, K., Automated Mill Pocketing Computations, *Proceedings of the International Symposium on Advanced Geometric Modelling for Engineering Applications*, November 8 - 10, 1989, Berlin, Germany, North Holland Publishers

[Stoutenborough] Stoutenborough, R., CAM Productivity with ANVIL-4000, *“Advanced Manufacturing Technologies” Proceedings of the 21st Annual Meeting and Technical Conference*, Numerical Control Society, March 25 - 28, 1984, Long Beach, California, pp. 112 - 121