'88 KACC 1988. 10. 21~22

NC TOOL PATH GENERATION OF ARBITRARILY SHAPED POCKETS

* Yong Seok Suh and Kunwoo Lee

Department of Mechanical Design and Production Engineering

Seoul National University, Seoul, Korea

ABSTRACT

In machining die cavities or mechanical parts, we often encounter the needs to remove a material within a given boundary. Even though this pocket cutting capability has been implemented in many NC packages, most of them can handle convex shaped pockets bounded by curves of limited types and numbers.

In this work, a procedure has been developed to machine pockets of a free surface bounded by lines, circular arcs and free curves. Also, the cutter location data is computed directly without using iterative method for better computational efficiency.

1. INTRODUCTION

In machining die cavities or mechanical parts, we often encounter the needs to remove material within a given boundary. Therefore, a procedure is needed to cut away the bounded area, so called a pocket, with an NC machine efficiently and exactly within some given tolerances. This pocket cutting capability has been implemented in many NC packages[1,2]. Even though many efficient ways of NC pocketing have been published[3], most of them can handle the pockets of limited shapes, i.e., convex-shaped pockets and/or pockets bounded by curves of limited types and numbers. Also, only few publications with the detail description of the algorithm can be found. Among the published works, Persson's early work[3] seems to describe the efficient and brilliant idea, but it cannot handle the pockets bounded by free curves.

The objective of this work is to derive a procedure in which the cutter location data (CL data) are generated in the proper sequence to cut the pockets of arbitrary shape, which is modelled by a B-Rep[4] based modeler. Our method is somewhat traditional in that the offset procedure is used, but rather general in that the pocket may be bounded by many arbitrary free curves with allowing some areas that are not to be machined so called "islands". Additionally, the bottom of the pocket to be machined may be any sculptured surface.

The procedure starts from a three dimensional model generated by a geometric modeler developed in our laboratory, which is a non-uniform rational B-spline[16] based B-Rep modeler. Once the shape of the pocket is modelled, the offset curves and the associated CL data are generated at every offsetting step. To machine pockets precisely within given tolerances, reliable CL data should be computed through a fast algorithm. In this work, the CL data are computed from each offset curve without using numerical iterative method. Then the sequence among the generated CL data is determined to cut the pocket in the right sequence.

Basic approach of our method is discussed in the next section, followed by more details.

2. BASIC APPROACH

A pocket can be machined in zig-zag or spiral-like fashion as shown in Figure 2.1, and the tool path generation of the spiral-like machining is considered in this work. The basic idea adopted in this work to generate the spiral-like tool path is to shrink the boundary profile inwardly while the island profile is expanded outwardly by a given amount step by step as illustrated in Figure 2.2. Here the term 'boundary profile' means a closed loop composed of the upper edges of the wall faces of a pocket, and the 'island profile' is a group of edges bounding the island which is the region not to be machined.

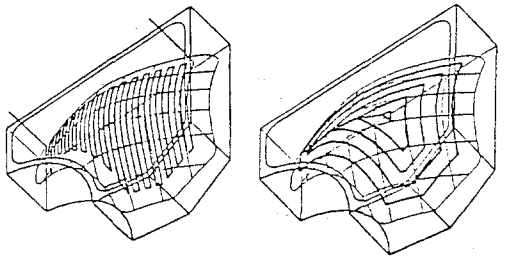


Figure 2.1 Tool path planning for pocket

The boundary profile is parametrized to have a clockwise direction when viewed above, and a counterclockwise direction for the island profile(s) as in Figure 2.2(a). The offset value is determined by the machining tolerances and geometrical considerations on the radius of curvature of the part surface at each step. As this offsetting process illustrated in Figure 2.2 goes on, three extraordinary cases may happen. The first case is that the boundary profile collapses and forms some loops by itself as in Figure 2.3(a). The same case can occur on the island profile with a concave portion. The loops should be processed such that the resulting profile is obtained as in Figure 2.3(b). The detail

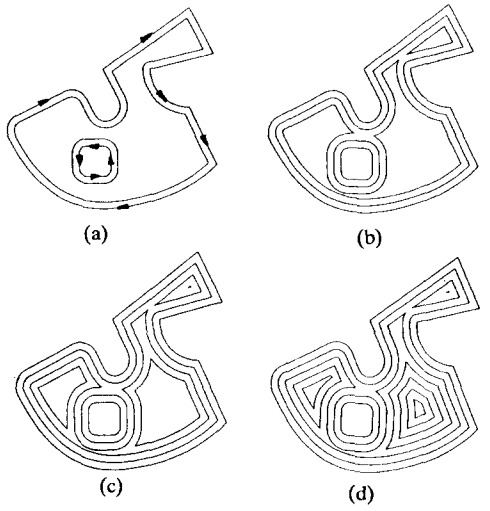


Figure 2.2 Offset procedure

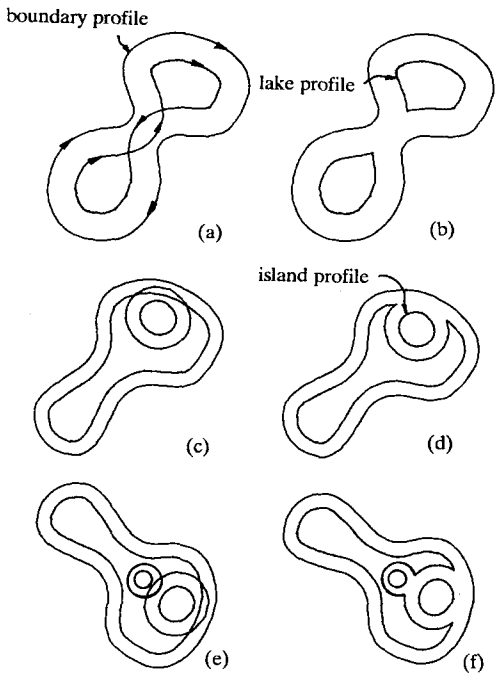


Figure 2.3

description of the processing the loops of self-intersections will be described later in section 2.1. The second case is shown in Figure 2.3(c), where the shrunk boundary profile intersects the expanded island profile(s) at that step. The resulting offset profile at this step has to be obtained as in Figure 2.3(d). The last case is that the expanded island profiles intersect each other as illustrated in Figure 2.3(e). This case should be corrected as in Figure 2.3(f) where the number of the islands is reduced. Note that, in the first and the second cases above, the resulting profile(s) is neither shrunk boundary nor the expanded island and this profile will be called "lake" profile from now on (see Figure 2. ( )). As can be seen easily, the lake profile can never exists with the boundary profile at the same time at each offsetting step.

In this work, the profile(s) corresponding to the previous offset step is overwritten by the current one(s) instead of storing the every offsetted profile to save the memory usage. So, the CL data corresponding to each step have to be calculated simultaneously during the offsetting procedure. In other words, Once the CL data are computed and stored for a step, the offset step procedure is performed and the corresponding CL data is calculated, stored again. The procedure is repeated until some terminal condition is satisfied. The offset procedure terminates when a profile is shrunk to a point by any one further step. When every profile stops offsetting, the overall offset procedure ends. The CL data are computed at each offsetting step as follows. After one offset step is performed, the offsetted profiles are swept down along -Z direction to yield the surfaces. Then, by intersecting the ruled surfaces and the bottom surface of the pocket, we can get the tool contact points(CC points). The CC data can be transformed easily to the CL data as described in section 2.3. Calculation of the CL data to machine a sculptured surface has been described in many publications [5,6,7,8,9], but they usually used numerical iterative method as in APT. The method has no guarantee on convergence and has difficulties in guessing the initial values. In this work, the CL data can be obtained directly as a by-product of the surface intersection between the ruled surfaces from the profile and the bottom surface. The detail will be explained in section 2.3. Note that at the first step of the offset procedure, the original boundary profile and the island profile(s) should be offsetted by the amount of the tool radius and the CC data is calculated by intersecting their ruled surfaces and the bottom surface which is offsetted by the tool radius in order to avoid the gouging problem that may happen at the boundary of two surfaces as in Figure 2.4[10].

The CL data calculated at each offset step is not arranged in the same sequence as the proper machining sequence but rugged and scattered here and there, as described previously. Thus the CL data should be sorted in a suitable order. The method of sorting will be described in later section. The resulting CL data can be transferred as a file or used for verifying the tool path.

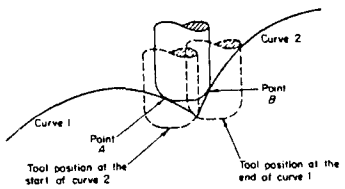


Figure 2.4 Gouging problem at surface boundary

2.1 Computation of offset curves

There are several articles on B-spline offset techniques[11,12]. For two dimensional offsetting, Tiller and Hanson's subdivision method[11] seems to be adequate, and thus the offset procedure in this work follows the Tiller's with some modifications. In this work, not only a curve but also a whole profile has to be offsetted. Here we use the term 'profile' to represent a closed two dimensional entity composed of many curves. Actually, a profile is composed of many kinds of curves and all the profiles used in this work are

closed. The profile of a pocket may be composed of arbitrary 3-D curves, but we don't need to consider the three dimensional offset problem because the eventual goal of our profile offset is obtaining the ruled surfaces by sweeping the profile in Z-direction. Tiller also gave a good algorithm to handle this profile offset problem. Only a brief discussion will be made about the procedure to offset a curve and a whole profile respectively. The full description can be found in the Tiller's.

Single curve offset

Since there are many literatures describing the B-spline curve[13,14, 15], only the expression of the nonrational and rational B-spline curves will be given here.[16]

$$\vec{C}(t) = \sum_{i=1}^{n} B_{i,k}(t)\,\vec{P}_i \qquad (1)$$

$$\vec{C}(t) = \frac{\sum_{i=1}^{n} B_{i,k}(t)\,h_i\,\vec{P}_i}{\sum_{i=1}^{n} B_{i,k}(t)\,h_i} \qquad (2)$$

where

P_i : *2-D or 3-D Coordinates of points called control points*
h_i : *Homogeneous coordinates*
t : *Parameter variable*
$B_{i,k}(t)$: *B-spline function in variable t of order k (degree k-1)*
n : *Number of the control points*

The B-spline function $B_{i,k}(t)$ is defined by the order k and a knot vector $\{t_j\}_{j=1}^{n+k}$. The offset of a curve $\vec{C}(t)$ can be obtained by

$$\vec{O}(t) = \vec{C}(t) + d\,\vec{N}(t) \qquad (3)$$

where d is the scalar value of the offset distance and $\vec{N}(t)$ is the unit normal vector of $\vec{C}(t)$ at parameter t. Generally, the offset of a B-spline curve cannot be expressed in a B-spline form[11], and thus it has to be approximated. Following the next steps will give a good approximation to the offset of a curve. The following step can be understood better with the help of Figure 2.5.

1. Set i ← 1
2. Let C be the curve and d be the offset distance. And let $\{A\}^i$ be a set of the original control points and $\{K\}^i$ be the corresponding knot vector.
3. Offset each legs of the control polygon $\{A\}^i$ by the distance d and then calculate the intersection of the offset polygon legs pairwise. The resulting point set $\{B\}^i$ is a new control polygon set, which has the same number as $\{A\}^i$. With this $\{B\}^i$ and the $\{K\}^i$, the roughly approximated offset O^i is defined.
4. Check the deviation Δ of O^i from the true offset which is computed by Equation (3)
5. If the Δ is greater or equal to given tolerance, subdivide $\{A\}^i$ and $\{K\}^i$ to get $\{A\}^{i+1}$,$\{K\}^{i+1}$. Then set i ← i+1 and go to step 2. If the deviation is less than the tolerance, stop the procedure.

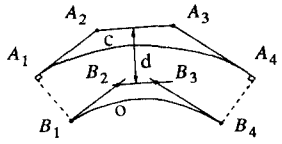


Figure 2.5 Polygon offset

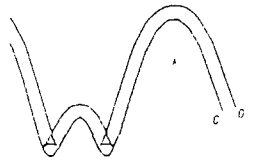


Figure 2.6 Self loop of a curve

There can be a self-loop problem as in Figure 2.6 when the radius of curvature at some point is less than the absolute value of the offset distance. This can be avoided simply as follows. First, calculate the two parametric values t_1 and t_2 corresponding to the self intersecting points. Then split the curve at t_1 and t_2 using Boehm algorithm[17] to divide the curve into three parts. Discarding the middle portion, the remaining two separated curves left take the place of the original one with the loop removed.

Profile offset

Let P = $\{C^i\}_{i=1}^{n}$ be a profile composed of n curves as shown in Figure 2.7. The following steps describe a procedure to offset a whole profile. Here $O_e^{\,i}$ and $O_s^{\,i+1}$ mean the end point of the offset of the i-th curve and the starting point of that of the (i+1)-th curve respectively, i.e. the superscript i stands for the i-th curve of the given profile.

1. Offset C^i and C^{i+1} to obtain O^i and O^{i+1}.
2. If $O_e^{\,i} = O_s^{\,i+1}$ Then
   go to END
3. Else
   begin
   Calculate tangent $\vec{T}_e^{\,i}$ and $\vec{T}_s^{\,i+1}$ at O_e^i and O_s^{i+1} respectively. Then, construct two lines $L^i = \vec{O}_e^{\,i} + \lambda^i \vec{T}_e^{\,i}$ and $L^{i+1} = \vec{O}_s^{\,i+1} + \lambda^{i+1}\vec{T}_s^{\,i+1}$ where λ^i and λ^{i+1} are parametric values. Intersecting the two lines, three cases arise.

   If (parallel or $\lambda^i < 0$ and $\lambda^{i+1} > 0$) Then
   Connect simply O_e^i and O_s^{i+1} by a line as in Figure 2.7(a).

   Else if ($\lambda^i\,\lambda^{i+1} > 0$) then
   Extend the two curves at both the points O_e^i and O_s^{i+1} by lines along their tangent direction respectively as in Figure 2.7(b).

   Else if ($\lambda^i > 0$ and $\lambda^{i+1} < 0$) then
   Connect the two points O_e^i and O_s^{i+1} by a circular arc as in Figure 2.7(c).

   End;

2.2 Processing loops of an offset profile

When a whole profile is offsetted, some loops may result as in the case of a single curve as illustrated in Figure 2.8. In this case, some loops have to be removed while others being alive. We will focus on the situation shown in Figure 2.8. Other situations shown in Figure 2.3(c) and (e) can be processed in the similar fashion. By giving a simple glimpse at Figure 2.8, we intuitively know that the loops L_2 and L_4 should be removed. But how can we let the computer perceive and

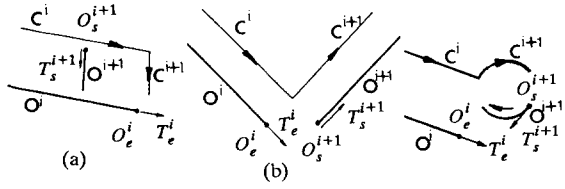


Figure 2.7

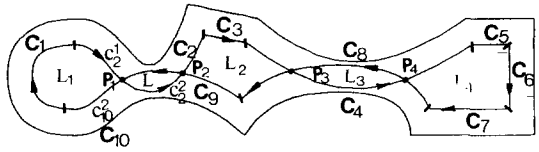


Figure 2.8 Self loops of a profile

process them automatically ? Tiller[11] gave several solutions to that question, but in this work, we invented a very simple and neat method suitable especially for this pocket profile offsetting. Remember that directions are already assigned to every profile from the beginning before the offset procedure as in Figure 2.2(a), and the direction does not change during the offset procedure. This means that if we start traveling from any one point on the profile and keep traveling in the direction of increasing parameter of each curve, then we can go through all the curves of the profile and return to the start point. Any existence of intersection point between the curves of a profile indicates the profile must have some self-loops as in Figure 2.8. For the example of Figure 2.8, the intersection between the curves will result points P_1,P_2,P_3 and P_4. Then, split the intersecting curves at those points. For example, splitting the curve C2 results in the curves C_2^1 and C_2^2. The split curves have the same directions as the original curve. Then the curves are grouped into several loops as follows. Start from any arbitrary curve (for example, curve C1 of Figure 2.8) and travel along the profile in its direction it, until any intersection point is encountered(point P_1 in this example). At this instant, store C_2^2 as the next starting curve of the next procedure, and transfer to the curve C_{10}^2 which intersects the current curve. Following C_{10}^2 in its direction, we return to the first starting curve, which forms the closed loop L_1. The above procedure is repeated for the next starting curve (C_2^2) to derive the loop L_2. The procedure is repeated until every intersection points P_i are visited twice. After this grouping procedure is performed, every loop is treated as a separated profile. In this example, loops L_2 and L_4 have counterclockwise direction when viewed above, while others do clockwise. Therefore, it can be concluded that the loops which is to be removed have a counterclockwise direction in this application. This method is based upon the fact that the boundary profile is directed in the clockwise direction and the island profile in the counterclockwise originally before the offset procedure start. Of course, if the direction is assigned in the other way from the beginning, the direction criterion for the removal must be reversed. The direction checking for a loop can be made easily by using the control points defining the loop. Eq. (4) gives the area(area vector) surrounded by a polygon in Figure 2.9 where $\hat{k}$ is the unit vector in Z-direction.

$$A\hat{k} = \frac{1}{2}\sum_{i=1}^{n} \vec{r}_i \times \vec{r}_{i+1} \qquad (4)$$

The sign of A depends on the direction of the vertices P_i. Suppose the $\{P_i\}$ be the control points of a loop. From the basic properties of the B-spline curve[27], the control points are ordered in the same direction as the loop and are quite similar in shape to the loop because they are subdivided during the curve offset procedure. If the value of A of a loop is greater than or equal to zero, the loop must have a counterclockwise direction and it has to be removed. Note that the island profiles are parametrized or directed reversely compared to the boundary or lake profiles in this work. So, the loops that have negative value of A should be removed for the island profiles. There is an exceptional case where this removal method cannot be applied. If the loops are nested so that one small loop is contained in the other bigger one as in Figure 2.10. Both the loops have the same directions, in which case the decision cannot be made. This nesting problem occurs at the concave portion of a profile when the radius of curvature reduces to the offsetting distance. For the boundary or lake profiles, the radius of curvature of the concave portion increases as they are offsetted and the nesting problem does not occur. For an island profile containing a concave portion, however, can be nested when offsetted. In this case, the only one loop with the longest arc length should be kept by removing all the others. The arc length can be approximated easily by the perimeter of the control polygon.

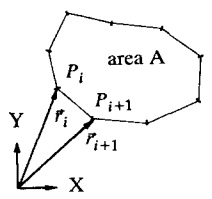


Figure 2.9 Vector area

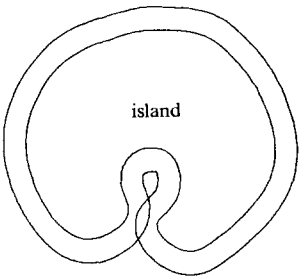


Figure 2.10 Nesting problem

2.3 Calculation of the CL data

The CL data are the 3-D Cartesian points through which the tool tip is to trace successively. For three axis NC milling machines, only the X,Y and Z coordinates will be sufficient to define a tool position. In this work, only the three-axis machining will be considered. The CL data can be calculated easily by Eq.(5) for a known tool contact point $\vec{r}$ when a ball-end mill is used.

$$\vec{r}_{CL}(u,v) = \vec{r}(u,v) + R\,(\,\hat{n}(u,v) - \hat{a}\,) \qquad (5)$$

where

$\vec{r}_{CL}(u,v)$: *CL data point*
$\vec{r}(u,v)$: *tool contact point on the part surface*
R : *radius of the ball-end mill*
$\hat{n}(u,v)$: *unit normal vector of the part surface at* u and v
$\hat{a}$: *tool axis vector*
for *three−axis machine*, $\hat{a} = [\,0\,0\,1\,]^T$

The tool contact points can be obtained by intersecting the bottom surface of the pocket and the ruled surfaces which are generated by the sweeping of the offsetted profiles along -Z direction (See Figure 2.11). Thus, the tool contact points(CC points) lie on the arbitrary 3-D intersection curve. Here we have to distinguish between the CC point and the CL point. The former represents the point lying on the part surface to be machined, where the tool is in contact, and the latter is the position of the tool tip which is normally off the part surface.

There are several ways to plan the tool path. Three types of tool path planning were presented by Choi [18]. The first is the so-called drive surface method used in APT III, and the second is to machine along the iso-parameteric curves of the part surface. The last is to plan the tool path in the Cartesian coordinates. The last method, so-called Cartesian machining, can be divided into two groups, i.e. CC-Cartesian and CL-Cartesian methods. The former method obtains the CC data by cutting the part surface with a vertical plane and then calculates the CL data from the CC data , and in the latter method, the CL data are obtained directly by the intersection of a plane and the offset surface of the part surface. Neither parametric machining nor the Cartesian machining can be used for the spiral-like tool path generation, and the APT method is not stable and time consuming as depicted in [18]. Therefore, a new method has been developed to generate the CL data by calculating the intersection between the bottom surface of a pocket and the ruled surfaces generated from the profiles. Here we now describe the method to calculate 'step lengths' and 'path intervals' with the comparison to the conventional method

Step Length

In the ideal case, the tool have to move in contact with the part surface continuously. But, in general, a surface intersection curve cannot be expressed in an analytic form and the NC machines use linear interpolation between the CL data points with satisfying some tolerances which limit the cutting error due to the linear interpolation as shown in Figure 2.12

Before the method developed in this work is presented, the conventional methods to calculate CL data and step lengths will be discussed. Faux and Pratt[19] describes the so-called minimum distance iteration method used in APT III, which is time consuming and may even fail to converge for an irregularly curved surfaces. In this method, the part surface is intersected by a plane called a drive surface and the tool moves in contact with both the drive surface and the part surface. The step length is defined by Eq.(6).

$$L = 2\sqrt{2\rho\delta - \delta^2} \qquad (6)$$

In this equation, the intersection curve between the two neighboring CC points was approximated to be a circle with radius equal to the radius of curvature of the part surface as in Figure 2.13. The distance D between the current CC point and the check surface as shown in Figure 2.13 and 2.14 is calculated by Eq. (7). Here the check surface is a plane normal to the tangent vector at the current CC point and passing through the next CC point to be obtained as shown in Figure 2.14.

$$D = L\left(1 - \frac{\delta}{\rho}\right) \qquad (7)$$

Once the value of D is computed by EQ.(7), the following equations are solved simultaneously to get the parametric value of u' and v' correspondingly to the next CC point shown in Figure 2.14.

$$\begin{aligned} [\,\vec{r}(u,v) - \vec{r}(u',v')\,] \cdot \hat{N}_D &= 0 \\ [\,\vec{r}(u,v) - \vec{r}(u',v')\,] \cdot \hat{N}_C &= D \end{aligned} \qquad (8)$$

where

$\vec{r}(u,v)$: *previous CC point*
$\vec{r}(u',v')$: *next CC point to be determined*
$\hat{N}_D$: *unit normal vector of the drive surface*
$\hat{N}_C$: *unit normal vector of the check surface*

Solving the two nonlinear equations for u' and v' using an iteration method is not a simple work, because there is no criterion to guess authentic initial values u',v'. In the above method, the next CC point, $\vec{r}(u',v')$ was computed by assuming that the segment between $\vec{r}(u,v)$ and $\vec{r}(u',v')$ is a circular arc. Thus the distance between the two CC points is not necessarily equal to L. If this distance is bigger than L, the value of L is reduced by half and the same procedure starting from Eq.(7) is repeated. In another Choi's work[18], the tool path is planned on x-y plane, where the pre-defined distance d between the sampling points is fixed(see Figure 2.15). This method cannot get economic CL data because for even a nearly flat part surface, too many CL data are to be calculated where only two CL points can cover.

In this work, the CL data is calculated by intersecting the bottom surface and the ruled surfaces generated from the profile. Two tolerances are defined in this work and they determine the positions and the numbers of the CL data points for a given tool path. The first is the allowable deviation, ϵ, between the interpolated line and the true profile when viewed above and the second, δ, is for the

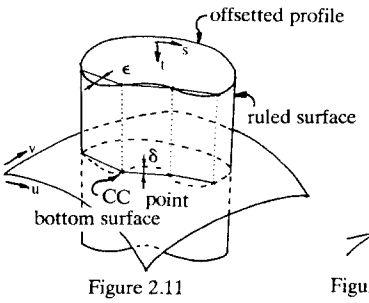


Figure 2.11

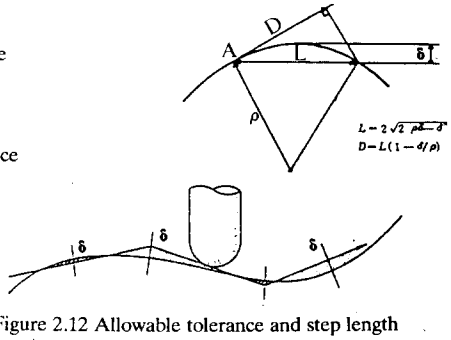


Figure 2.13 Calculation of step length

Figure 2.12 Allowable tolerance and step length

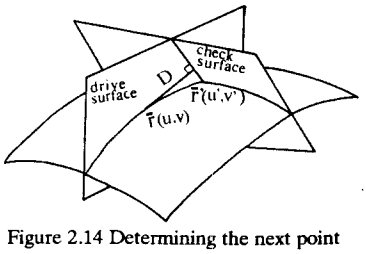


Figure 2.14 Determining the next point

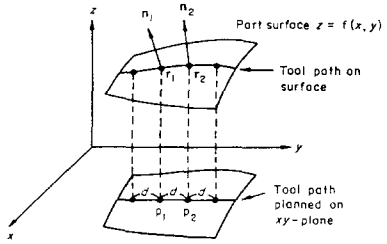


Figure 2.15

bottom surface as illustrated in Figure 2.11. Using these two tolerances ϵ and δ, we have to calculate the 'best' CL data (i.e. the least number of CL data points satisfying the tolerances). In this method, the subdivision method [20,21,22,23] is used, i.e., both the patches are subdivided to be approximated as a small planar quadrilaterals and the intersection points are obtained by intersecting the quadrilaterals as shown in Figure 2.16. Actually, these points obtained by the subdivision method can be used as a good initial values for a Newton-Raphson method for a better accuracy by using the following three equations (9) and one of (10).[23]

$$\vec{F}(u,v) = \vec{R}(s,t) \tag{9}$$

$$\begin{cases} v = constant \\ v = constant \\ s = constant \\ t = constant \end{cases} \tag{10}$$

where

$\vec{F}(u,v)$: *the bottom surface of a pocket*

$\vec{R}(s,t)$: *the ruled surface*

The initial values are good enough that the convergency is generally guaranteed with only a few iterations. The basic idea to calculate the CL data are based on the subdivision surface intersection algorithm in which the patches are approximated as planar quadrilaterals during the process. There must be some criterion values determining whether a subdivided patch can be approximated to a plane or not for each surface in that routine. If the step length tolerances, i.e. δ for the bottom surface and ϵ for the ruled surfaces are substituted for the criterion value, then it may be possible that the resulting intersection points are the CC points satisfying the step length tolerances.

The above procedure can generate economic CL data adapting to the curvature of the bottom surface of the pocket automatically, without calculating the actual curvature which requires the calculation of the first and second differentiation of the surface. These method can also be applied to the Cartesian machining(Figure 2.17), where the vertical plane intersects the part surface. Through this method, we can get reliable and near optimal CL data with little computation time.

Path Interval

The distance between the tool paths are termed as a path interval. If the bottom surface of the pocket is flat, the path interval is constant(tool radius R, for example) for all paths. But, for a free-formed bottom surface, the path interval may be different path to path, depending on its radius of curvature and the given tolerance of scallop height (or cusp height) shown in the Figure 2.18. Calculating the path intervals for the parametric or Cartesian machining has been dealt in many publications[8,18,10]. In this work, the path intervals are calculated in the similar way for the spiral-like tool path generation. The procedure are described step by step for a given cutter contact point.

1. Calculate the normal vector $\hat{B}$ and the tangent vector $\hat{T}$ of the iso-parametric curve in s of the ruled surface at the CC point. (See Figure 2.19)
2. The radius of curvature used for the path interval is that of the normal curvature of the intersection curve between the bottom surface and a plane which contains the vectors, $\hat{B}$ and $\hat{B}\times\hat{T}$. If we denote the intersection curve $\vec{r}(t)$, then from the relation,

$$\dot{\vec{r}}\cdot\hat{T} = (\vec{r}_u\dot{u} + \vec{r}_v\dot{v})\cdot\hat{T} = (\vec{r}_u\cdot\hat{T})\dot{u} + (\vec{r}_v\cdot\hat{T})\dot{v} = 0$$

we can get the ratio $\dot{u}$ and $\dot{v}$ by (11)

$$\dot{u} : \dot{v} = \vec{r}_v\cdot\hat{T} : -(\vec{r}_u\cdot\hat{T}) \tag{12}$$

3. Then we can calculate $\dot{\vec{r}}$ using equation (11)

$$\dot{\vec{r}} = \vec{r}_u\dot{u} + \vec{r}_v\dot{v}$$

4. The curvature of the intersection curve can be evaluated as follows.

$$\rho = \frac{\dot{u}^T [D]\,\dot{u}}{\dot{u}^T [G]\,\dot{u}} \tag{13}$$

$$[D] = \begin{bmatrix} \hat{n}\cdot\vec{r}_{uu} & \hat{n}\cdot\vec{r}_{uv} \\ \hat{n}\cdot\vec{r}_{uv} & \hat{n}\cdot\vec{r}_{vv} \end{bmatrix} \tag{14}$$

$$[G] = \begin{bmatrix} \vec{r}_u\cdot\vec{r}_u & \vec{r}_u\cdot\vec{r}_v \\ \vec{r}_u\cdot\vec{r}_v & \vec{r}_v\cdot\vec{r}_v \end{bmatrix} \tag{15}$$

ρ : *radius of curvature of the curve*

$\dot{u} = [u\,v]^T$

5. Now a distance l (see Figure 2.19) can be calculated using the equations (16).

$$l = \begin{cases} \dfrac{2\sqrt{2Rh - h^2}}{(R+\rho)(h+\rho)} & \rho = \infty \\ \dfrac{\rho\sqrt{4(R+\rho)^2(h+\rho)^2 - \{\rho^2 + 2R\rho + (h+\rho)^2\}^2}}{(R+\rho)(h+\rho)} & \rho \neq \infty \end{cases} \tag{16}$$

R : *radius of the tool*

h : *allowable scallop height*

ρ : *radius of curvature*

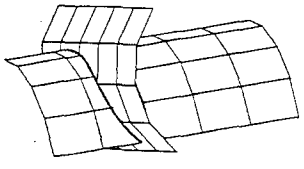


Figure 2.16 intersection using subdivision method

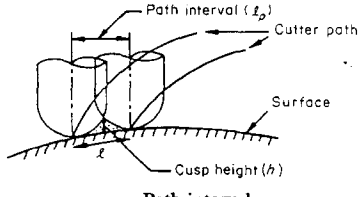


Figure 2.18 Path interval

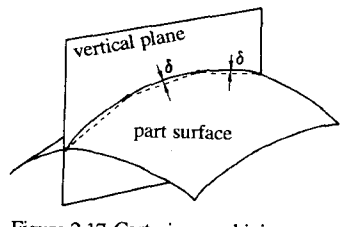


Figure 2.17 Cartesian machining

Then the tangential direction d can be calculated by

$$d = \begin{cases} l\sqrt{4\rho^2 - l^2} \,/\, 2\rho & \rho \neq \infty \\ l & \rho = \infty \end{cases} \quad (17)$$

6. Then the path interval W can be calculated.

$$W = d\,|\hat{B}\cdot\hat{r}| \quad (18)$$

The above calculation is performed for every tool contact point of a tool path and the smallest W is chosen as the path interval, i.e., the offset distance for generating the next profile.

2.3 Determination of Cutting order

After the offset procedure is completed, the calculated CL data must be sorted in the machining sequence. Let's make it a rule that the machining starts at the inner-most point of the pocket, and expands its move outward (i.e., toward the boundary of the pocket). The offsetted profiles and their corresponding CL data can be expressed as a tree structure as shown in the Figure 2.20, where each node denotes a profile and the branches represents the causality between the two profiles, that is, the lower node(s) is(are) offsetted from the upper node(profile) connected by the branch. Therefore, each depth of the tree represents each offset step respectively. For example, at the fifth offset step, 6 and 7,8,9 profiles exist that are generated from the previous profiles 4 and 5, respectively. The above example in Figure 2.20 terminated its offset procedure at its tenth step. Note that the CL data for the island profile were not taken into account because the offsetting procedure of the island profile corresponds to the machining sequence and no sorting is required. The CL data of the island profiles are stored in an another separate array CLISL (Figure 2.22), and these CL data must be used in machining prior to the others. The tree structure is stored in an array ICLPTR as in Figure 2.21. The CL data corresponding to the boundary or lake profiles are stored in CLDAT array (Figure 2.22). Now we describe the steps to get ordered CL data as follows.

1. The pocket starts its machining along the island profile CL data stored in the CLISL array.
2. The deepest node of the tree (for example profile 23 in Figure 2.21) is machined first, and its father (previous profile) is machined next. This tree-climbing-up continues until a node with (a) brother node(s) is met. (node 9 in this case) The brother node stands for the profile which has a common father profile. For example, node 9 have two brother profiles 7 and 8.
3. If a node has its brother profile and the brother profile is not yet machined, the next profile to be machined is the deepest node of the brother (node 17 or 16 in this case)
4. The similar procedure as in step 2. and 3. are repeated until no father is detected.

Following the previous steps results the machining sequence for the example in Figure 2.21 as below :

23 → 22 → 21 → 18 → 13 → 9 → 17 → 12 → 8 → 16 → 11 → 7 → 5 → 20 → 15 → 19 → 14 → 10 → 6 → 4 → 3 → 2 → 1

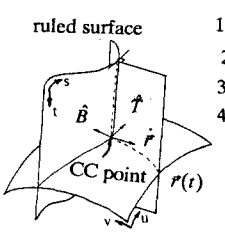


Figure 2.19

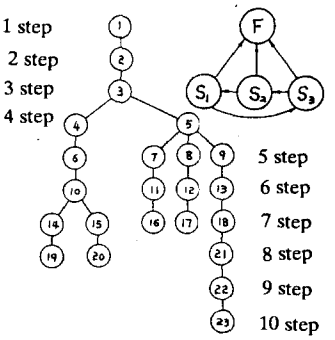


Figure 2.20 ordering tree

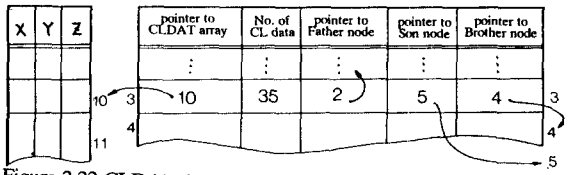


Figure 2.22 CLDAT (3,*) or CLISL (3,*) array

Figure 2.21 ICLPTR (5,*) array

3. CASE STUDY

A pocket of Figure 3.1 has twelve boundary curves and an island which is composed of eight curves. The bottom surface of the pocket is a smooth convex cubic Bezier patch. The dimension of the pocket is about 200 × 200 (mm).

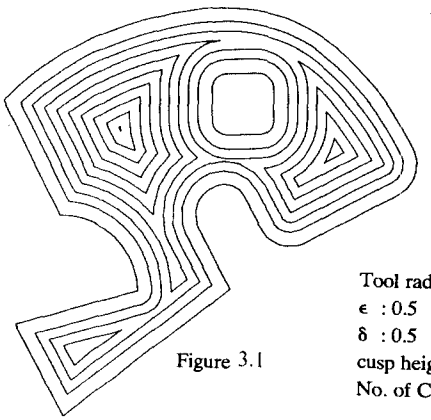
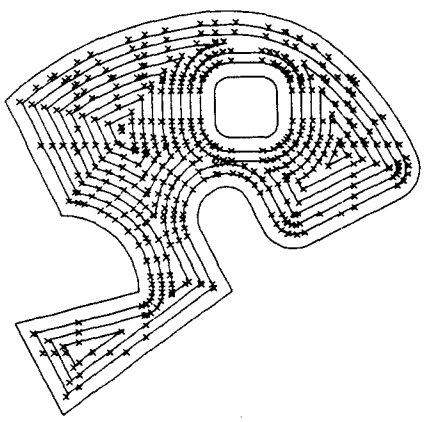


Figure 3.1

Tool radius : 6.0
ϵ : 0.5
δ : 0.5
cusp height : 0.4
No. of CL data : 633



REFERENCES

[1] CALMA NC Milling Applications User's Manual, Calma company

[2] CATIA User's Manual, IBM, 1984

[3] H. Persson, "NC machining of arbitrarily shaped pockets", Computer-Aided Design, Vol. 10, No. 3, 1978, pp. 169-174

[4] M. E. Mortenson, Geometric Modeling, John Wiley & Sons, 1985

[5] D H Kim, "Calculation of economic CL-data for sculptured surface machining", MS thesis, KAIST, 1984

[6] J H Lee, "Free-frmed surface design and NC machining using a cross-sectional method", MS thesis, KAIST, 1985

[7] C S Jun, "NC machining of sculptured surface from 3D measuring data", MS thesis, KAIST, 1985

[8] B K Choi, "Surface modeling and 3-D NC machining", Banghan pub. 1985

[9] G C Loney, T M Ozsoy, "NC machining of free form surfaces", Computer-Aided Design, Vol.19, No.2, 1987

[10] Lee, S C, "The tool path calculation for the NC machining of a bounded free-form surface", MS thesis, Seoul National University, Seoul, Korea, 1988

[11] W Tiller, E Hanson, "Offset of two-dimensional profiles", IEEE Comp. Graph. Appl. Vol. 4 No. 9 1984 pp. 36-46

[12] S Coquillart, "Computing offsets of B-spline curves", Computer-Aided Design, July, 1987

[13] C de Boor, "On calculating with B-spline", J Approximation Theory, Vol. 6, 1972, pp. 52-60

[14] C de Boor, "A practical Guide to splines", Springer-Verlag, 1978

[15] L L Schumaker, "Spline functions: basic theory", John Willey & Sons,1981

[16] W Tiller, "Rational B-spline curve and surface representation", IEEE Computer Graphics and Application, 1983, pp. 61-69

[17] W Boehm, "Inserting New knots into B-spline curves", Computer-Aided Design, Vol. 12, No. 4, 1980, pp. 199-201

[18] B K Choi, C S Lee, J S Hwang and C S Jun, "Compound surface modeling and and machining", Computer-Aided Design, Vol. 20, No. 3, 1988, pp. 127-136

[19] I D Faux and M I Pratt, "Computational geometry for design and manufacture", Ellis Horwood Ltd., Chichester, W. Sussex, UK, 1979

[20] E Coehn, T Lyche and R Riesenfeld, "Discrete B-splines and subdividsion techniques in computer aided geometric design and computer graphics", Computer Graphics and Image Processing, Vol. 14, 1980, pp. 87-101

[21] Q S Peng, "An algorithm for finding the intersection lines between two B-spline surfaces", Computer-Aided Design, Vol. 16, 1984, pp.191-196

[22] D Lasser, "Intersection of parametric surfaces in the Bernstein Bezier representation", Computer-Aided Design, Vol. 18, 1986, pp.186-192

[23] P A Koparkar and S P Mudur, "Generation of continuous smooth curves resulting from operations on parametric surface patches", Computer-Aided Design, Vol. 18, No. 4, 1986, pp. 193-206

[24] W Boehm, "Generating the Bezier points of B-spline curves and surfaces", Computer-Aided Design, Vol. 13, No. 6, 1981, pp. 365-366

[25] L Piegel, " Representation of rational Bezier curves and surfaces by recursive algorithms", Computer-Aided Design, Vol. 18, 1986, pp. 361-366

[26] J E Bobrow, "NC machine tool path generation from CSG part representations", Computer-Aided Design, Vol. 17, No. 2, 1985, pp. 69-75

[27] W Boehm, G Farin and J Kahmann, "A survey of curved surface methods in CAGD", Computer Aided Geometric Design, 1984,